Final Project Submission

- Name : Kagiri Malvis Karuma
- Data Science Full Time
- Technical Mentors: Mark Tiba
- blog url: https://medium.com/@malvis.kagiri/7de0c3a69dbe

Project Overview

For this project I will use exploratory data analysis to generate insights to a business stakeholder.

Business Problem

Microsoft sees all the big companies creating original video content and they want to get in on the fun. They have decided to create a new movie studio, but they decide to create a movie studio, but they don't know anything about creating movies.

Objective

- Explore the types of films in production and the different factors that aid in its production and success.
- Translate these findings into actionable insights

Datasets

1. tmdb.movies.csv
2. rt.movies_info.tsv
3. tn.movie_budgets.csv

1. Import Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
import seaborn as sns
```

1. Import Dataset

In [3]:
```
#tmd.movies.csv
reviews = pd.read_csv("tmdb.movies.csv")
#rt.movies_info.tsv
rt = pd.read_table("rt.movie_info.tsv")
#tn.movie_budgets
budgets = pd.read_csv("tn.movie_budgets.csv")
```

```
#imbd_top_1000
net = pd.read_csv('imdb_top_1000.csv')
```

1. Explore the Datasets

tmd.movies.csv

In [4]:
```
reviews.isnull().sum()
```

Out[4]:
```
Unnamed: 0          0
genre_ids           0
id                  0
original_language   0
original_title      0
popularity          0
release_date        0
title               0
vote_average        0
vote_count          0
dtype: int64
```

In [5]:
```
reviews.head(10)
```

Out[5]:

	Unnamed: 0	genre_ids	id	original_language	original_title	popularity	release_date	title	vote
0	0	[12, 14, 10751]	12444	en	Harry Potter and the Deathly Hallows: Part 1	33.533	2010-11-19	Harry Potter and the Deathly Hallows: Part 1	
1	1	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	
2	2	[12, 28, 878]	10138	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	
3	3	[16, 35, 10751]	862	en	Toy Story	28.005	1995-11-22	Toy Story	
4	4	[28, 878, 12]	27205	en	Inception	27.920	2010-07-16	Inception	
5	5	[12, 14, 10751]	32657	en	Percy Jackson & the Olympians: The Lightning T...	26.691	2010-02-11	Percy Jackson & the Olympians: The Lightning T...	
6	6	[28, 12, 14, 878]	19995	en	Avatar	26.526	2009-12-18	Avatar	
7	7	[16, 10751, 35]	10193	en	Toy Story 3	24.445	2010-06-17	Toy Story 3	
8	8	[16, 10751, 35]	20352	en	Despicable Me	23.673	2010-07-09	Despicable Me	

	Unnamed: 0	genre_ids	id	original_language	original_title	popularity	release_date	title	vote
9	9	[16, 28, 35, 10751, 878]	38055	en	Megamind	22.855	2010-11-04	Megamind	

In [6]:
```
reviews.tail(10)
```

Out[6]:

	Unnamed: 0	genre_ids	id	original_language	original_title	popularity	release_date	
26507	26507	[99]	545555	ar	Dreamaway	0.6	2018-10-14	Drea
26508	26508	[16]	514492	en	Jaws	0.6	2018-05-29	
26509	26509	[27]	502255	en	Closing Time	0.6	2018-02-24	Closi
26510	26510	[99]	495045	en	Fail State	0.6	2018-10-19	F
26511	26511	[99]	492837	en	Making Filmmakers	0.6	2018-04-07	Filn
26512	26512	[27, 18]	488143	en	Laboratory Conditions	0.6	2018-10-13	Lal Co
26513	26513	[18, 53]	485975	en	_EXHIBIT_84xxx_	0.6	2018-05-01	_EXHIBI
26514	26514	[14, 28, 12]	381231	en	The Last One	0.6	2018-10-01	The L
26515	26515	[10751, 12, 28]	366854	en	Trailer Made	0.6	2018-06-22	Trail
26516	26516	[53, 27]	309885	en	The Church	0.6	2018-10-05	The

In [7]:
```
reviews.shape
```

Out[7]:
```
(26517, 10)
```

In [8]:
```
reviews.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26517 entries, 0 to 26516
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Unnamed: 0        26517 non-null  int64  
 1   genre_ids         26517 non-null  object 
 2   id                26517 non-null  int64  
 3   original_language 26517 non-null  object 
 4   original_title    26517 non-null  object 
 5   popularity        26517 non-null  float64
 6   release_date      26517 non-null  object 
 7   title             26517 non-null  object 
 8   vote_average      26517 non-null  float64
 9   vote_count        26517 non-null  int64  
dtypes: float64(2), int64(3), object(5)
memory usage: 2.0+ MB
```

In [9]:
```
reviews.columns
```

Out[9]:
```
Index(['Unnamed: 0', 'genre_ids', 'id', 'original_language', 'original_title',
       'popularity', 'release_date', 'title', 'vote_average', 'vote_count'],
      dtype='object')
```

In [ ]:
```

```

rt.movies_info.tsv

In [10]:
```
rt.isnull().sum()
```

Out[10]:
```
id              0
synopsis       62
rating          3
genre           8
director      199
writer        449
theater_date  359
dvd_date      359
currency     1220
box_office   1220
runtime        30
studio       1066
dtype: int64
```

In [11]:
```
rt.head(10)
```

Out[11]:

	id	synopsis	rating	genre	director	writer	theater_date	dvd_dat
0	1	This gritty, fast-paced, and innovative police...	R	Action and Adventure|Classics|Drama	William Friedkin	Ernest Tidyman	Oct 9, 1971	Sep 2 200
1	3	New York City, not-too-distant-future: Eric Pa...	R	Drama|Science Fiction and Fantasy	David Cronenberg	David Cronenberg|Don DeLillo	Aug 17, 2012	Jan 201
2	5	Illeana Douglas delivers a superb performance ...	R	Drama|Musical and Performing Arts	Allison Anders	Allison Anders	Sep 13, 1996	Apr 1 200
3	6	Michael Douglas runs afoul of a treacherous su...	R	Drama|Mystery and Suspense	Barry Levinson	Paul Attanasio|Michael Crichton	Dec 9, 1994	Aug 2 199
4	7	NaN	NR	Drama|Romance	Rodney Bennett	Giles Cooper	NaN	Nal
5	8	The year is 1942. As the Allies unite overseas...	PG	Drama|Kids and Family	Jay Russell	Gail Gilchriest	Mar 3, 2000	Jul 1 200

	id	synopsis	rating	genre	director	writer	theater_date	dvd_dat
6	10	Some cast and crew from NBC's highly acclaimed...	PG-13	Comedy	Jake Kasdan	Mike White	Jan 11, 2002	Jun 18 200
7	13	Stewart Kane, an Irishman living in the Austra...	R	Drama	Ray Lawrence	Raymond Carver|Beatrix Christian	Apr 27, 2006	Oct 2 200
8	14	"Love Ranch" is a bittersweet love story that ...	R	Drama	Taylor Hackford	Mark Jacobson	Jun 30, 2010	Nov 9 201
9	15	When a diamond expedition in the Congo is lost...	PG-13	Action and Adventure|Mystery and Suspense|Scie...	Frank Marshall	John Patrick Shanley	Jun 9, 1995	Jul 27 199

In [12]:
```
rt.shape
```

Out[12]:
```
(1560, 12)
```

In [13]:
```
rt.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1560 entries, 0 to 1559
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   id            1560 non-null   int64 
 1   synopsis      1498 non-null   object
 2   rating        1557 non-null   object
 3   genre         1552 non-null   object
 4   director      1361 non-null   object
 5   writer        1111 non-null   object
 6   theater_date  1201 non-null   object
 7   dvd_date      1201 non-null   object
 8   currency      340 non-null    object
 9   box_office    340 non-null    object
 10  runtime       1530 non-null   object
 11  studio        494 non-null    object
dtypes: int64(1), object(11)
memory usage: 146.4+ KB
```

In [14]:
```
rt.columns
```

Out[14]:
```
Index(['id', 'synopsis', 'rating', 'genre', 'director', 'writer',
       'theater_date', 'dvd_date', 'currency', 'box_office', 'runtime',
       'studio'],
      dtype='object')
```

In [ ]:

tn.movie_budgets.csv

In [15]:
```
budgets.isnull().sum()
```

Out[15]:
```
id                   0
release_date         0
movie                0
production_budget    0
domestic_gross       0
worldwide_gross      0
dtype: int64
```

In [16]:
```
budgets.columns
```

Out[16]:
```
Index(['id', 'release_date', 'movie', 'production_budget', 'domestic_gross',
       'worldwide_gross'],
      dtype='object')
```

In [17]:
```
budgets.shape
```

Out[17]:
```
(5782, 6)
```

In [18]:
```
budgets.head(10)
```

Out[18]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
0	1	Dec 18, 2009	Avatar	$425,000,000	$760,507,625	$2,776,345,279
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	$410,600,000	$241,063,875	$1,045,663,875
2	3	Jun 7, 2019	Dark Phoenix	$350,000,000	$42,762,350	$149,762,350
3	4	May 1, 2015	Avengers: Age of Ultron	$330,600,000	$459,005,868	$1,403,013,963
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	$317,000,000	$620,181,382	$1,316,721,747
5	6	Dec 18, 2015	Star Wars Ep. VII: The Force Awakens	$306,000,000	$936,662,225	$2,053,311,220
6	7	Apr 27, 2018	Avengers: Infinity War	$300,000,000	$678,815,482	$2,048,134,200
7	8	May 24, 2007	Pirates of the Caribbean: At Worldâ s End	$300,000,000	$309,420,425	$963,420,425
8	9	Nov 17, 2017	Justice League	$300,000,000	$229,024,295	$655,945,209
9	10	Nov 6, 2015	Spectre	$300,000,000	$200,074,175	$879,620,923

imdb_top_1000.csv

In [19]:
```
net.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 16 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   Poster_Link    1000 non-null   object
 1   Series_Title   1000 non-null   object
 2   Released_Year  1000 non-null   object
```

```
 3   Certificate    899 non-null    object
 4   Runtime        1000 non-null   object
 5   Genre          1000 non-null   object
 6   IMDB_Rating    1000 non-null   float64
 7   Overview       1000 non-null   object
 8   Meta_score     843 non-null    float64
 9   Director       1000 non-null   object
 10  Star1          1000 non-null   object
 11  Star2          1000 non-null   object
 12  Star3          1000 non-null   object
 13  Star4          1000 non-null   object
 14  No_of_Votes    1000 non-null   int64
 15  Gross          831 non-null    object
dtypes: float64(2), int64(1), object(13)
memory usage: 125.1+ KB
```

In [20]:
```
net.describe()
```

Out[20]:

	IMDB_Rating	Meta_score	No_of_Votes
count	1000.000000	843.000000	1.000000e+03
mean	7.949300	77.971530	2.736929e+05
std	0.275491	12.376099	3.273727e+05
min	7.600000	28.000000	2.508800e+04
25%	7.700000	70.000000	5.552625e+04
50%	7.900000	79.000000	1.385485e+05
75%	8.100000	87.000000	3.741612e+05
max	9.300000	100.000000	2.343110e+06

In [21]:
```
net.columns
```

Out[21]:
```
Index(['Poster_Link', 'Series_Title', 'Released_Year', 'Certificate',
       'Runtime', 'Genre', 'IMDB_Rating', 'Overview', 'Meta_score', 'Director',
       'Star1', 'Star2', 'Star3', 'Star4', 'No_of_Votes', 'Gross'],
      dtype='object')
```

In [22]:
```
net.head(10)
```

Out[22]:

	Poster_Link	Series_Title	Released_Year	Certificate	Runtime	Genre
0	https://m.media-amazon.com/images/M/MV5BMDFkYT...	The Shawshank Redemption	1994	A	142 min	Drama
1	https://m.media-amazon.com/images/M/MV5BM2MyNj...	The Godfather	1972	A	175 min	Crime, Drama

	Poster_Link	Series_Title	Released_Year	Certificate	Runtime	Genre
2	https://m.media-amazon.com/images/M/MV5BMTMxNT...	The Dark Knight	2008	UA	152 min	Action, Crime, Drama
3	https://m.media-amazon.com/images/M/MV5BMWMwMG...	The Godfather: Part II	1974	A	202 min	Crime, Drama
4	https://m.media-amazon.com/images/M/MV5BMWU4N2...	12 Angry Men	1957	U	96 min	Crime, Drama
5	https://m.media-amazon.com/images/M/MV5BNzA5ZD...	The Lord of the Rings: The Return of the King	2003	U	201 min	Action, Adventure, Drama
6	https://m.media-amazon.com/images/M/MV5BNGNhMD...	Pulp Fiction	1994	A	154 min	Crime, Drama
7	https://m.media-amazon.com/images/M/MV5BNDE4OT...	Schindler's List	1993	A	195 min	Biography, Drama, History
8	https://m.media-amazon.com/images/M/MV5BMjAxMz...	Inception	2010	UA	148 min	Action, Adventure, Sci-Fi
9	https://m.media-amazon.com/images/M/MV5BMmEzNT...	Fight Club	1999	A	139 min	Drama

In [23]:
```
net.tail(10)
```

Out[23]:

	Poster_Link	Series_Title	Released_Year	Certificate	Runtime	Genre

	Poster_Link	Series_Title	Released_Year	Certificate	Runtime	Genre
990	https://m.media-amazon.com/images/M/MV5BYjRmY2...	Giù la testa	1971	PG	157 min	Drama, War, Western
991	https://m.media-amazon.com/images/M/MV5BMzAyND...	Kelly's Heroes	1970	GP	144 min	Adventure, Comedy, War
992	https://m.media-amazon.com/images/M/MV5BMjAwMT...	The Jungle Book	1967	U	78 min	Animation, Adventure, Family
993	https://m.media-amazon.com/images/M/MV5BYTE4YW...	Blowup	1966	A	111 min	Drama, Mystery, Thriller
994	https://m.media-amazon.com/images/M/MV5BZjQyMG...	A Hard Day's Night	1964	U	87 min	Comedy, Music, Musical
995	https://m.media-amazon.com/images/M/MV5BNGEwMT...	Breakfast at Tiffany's	1961	A	115 min	Comedy, Drama, Romance
996	https://m.media-amazon.com/images/M/MV5BODk3Yj...	Giant	1956	G	201 min	Drama, Western
997	https://m.media-amazon.com/images/M/MV5BM2U3Yz...	From Here to Eternity	1953	Passed	118 min	Drama, Romance, War
998	https://m.media-amazon.com/images/M/MV5BZTBmMj...	Lifeboat	1944	NaN	97 min	Drama, War
999	https://m.media-amazon.com/images/M/MV5BMTY5OD...	The 39 Steps	1935	NaN	86 min	Crime, Mystery, Thriller

Data Cleaning

- check for duplicates
- Drop columns
- Change column values to their correct format

In [24]:
```
##tmdb.movies.csv
# remove unwanted columns
reviews = reviews.drop(['Unnamed: 0', 'genre_ids', 'original_title', 'release_date', 'tit
```

In [25]:
```
reviews
```

Out[25]:

	id	original_language	popularity	vote_average	vote_count
0	12444	en	33.533	7.7	10788
1	10191	en	28.734	7.7	7610
2	10138	en	28.515	6.8	12368
3	862	en	28.005	7.9	10174
4	27205	en	27.920	8.3	22186
...	...	...	...	...	...
26512	488143	en	0.600	0.0	1
26513	485975	en	0.600	0.0	1
26514	381231	en	0.600	0.0	1
26515	366854	en	0.600	0.0	1
26516	309885	en	0.600	0.0	1

26517 rows × 5 columns

In [26]:
```
# rt.movie_info.tsv
import re
#remove unwanted columns
rt = rt.drop(['studio', 'id', 'currency', 'box_office', 'synopsis', 'director', 'writer']
# remove the word minutes
rt['runtime']=rt.runtime.str.replace('minutes','')

# converting it to a float
rt['runtime'] = pd.to_numeric(rt.runtime)
# replacing NaN with G 'general viewing'
rt.rating.fillna("G")
# remove every thing form dvd_date except year
rt.dvd_date = (rt.dvd_date.str.replace(r'^[^,]*,\s*', ''))
```

In [27]:
```
rt.rating.dropna()
```

Out[27]:
```
0          R
1          R
2          R
3          R
4         NR
         ..
1555       R
1556      PG
1557       G
1558      PG
1559       R
Name: rating, Length: 1557, dtype: object
```

In [28]:
```
rt.rating.isnull().sum()
```

Out[28]: 3

In [29]:
```
# tn.movie_budgets.csv
# removing everything before the comma
budgets.release_date = (budgets.release_date.str.replace(r'^[^,]*,\s*', ''))
# removing the dollar sign
colstocheck = budgets.columns
budgets[colstocheck] = budgets[colstocheck].replace({r'\$':''}, regex = True)
# removing the commas separating the digits
budgets['production_budget']=budgets.production_budget.str.replace(',','')
budgets['worldwide_gross']=budgets.worldwide_gross.str.replace(',','')
budgets['domestic_gross']=budgets.worldwide_gross.str.replace(',','')
# convert the string values to integers
budgets['production_budget'] = pd.to_numeric(budgets.production_budget)
budgets['worldwide_gross'] = pd.to_numeric(budgets.worldwide_gross)
budgets['domestic_gross'] = pd.to_numeric(budgets.worldwide_gross)
```

In [30]:
```
budgets.columns
```

Out[30]:
```
Index(['id', 'release_date', 'movie', 'production_budget', 'domestic_gross',
       'worldwide_gross'],
      dtype='object')
```

In [31]:
```
budgets.describe()
```

Out[31]:

	id	production_budget	domestic_gross	worldwide_gross
count	5782.000000	5.782000e+03	5.782000e+03	5.782000e+03
mean	50.372363	3.158776e+07	9.148746e+07	9.148746e+07
std	28.821076	4.181208e+07	1.747200e+08	1.747200e+08
min	1.000000	1.100000e+03	0.000000e+00	0.000000e+00
25%	25.000000	5.000000e+06	4.125415e+06	4.125415e+06
50%	50.000000	1.700000e+07	2.798445e+07	2.798445e+07
75%	75.000000	4.000000e+07	9.764584e+07	9.764584e+07
max	100.000000	4.250000e+08	2.776345e+09	2.776345e+09

In [32]:
```
##imdb_top_1000.csv
net = net.drop(['Poster_Link', 'Series_Title', 'Certificate', 'Overview', 'Director', 'St
net['Gross']=net.Gross.astype(str).str.replace(',','')
net['Gross']=net.Gross.astype(str).str.replace('nan','0')
net['Runtime']=net.Runtime.str.replace(' min','')
net.Meta_score = net.Meta_score.fillna(0)

# net.Gross = net.Gross.fillna(0)
net['Gross'] = pd.to_numeric(net.Gross)
net['Runtime'] = pd.to_numeric(net.Runtime)
```

Explore the overall statistics

In [33]:
```
# rt.movie_info()
Q1 = rt.runtime.quantile(0.25)
Q3 = rt.runtime.quantile(0.75)
IQR = Q3 - Q1
```

```
print('IQR =', IQR)
lower_limit = Q1 - 1.5 * IQR
upper_limit = Q3 + 1.5 * IQR
print('upper_limt =', upper_limit)
print('lower_limit =', lower_limit)
```

```
IQR = 23.0
upper_limt = 148.5
lower_limit = 56.5
```

In [34]:
```
outliers_lower = rt.runtime < lower_limit
outliers_upper = rt.runtime > upper_limit
```

In [35]:
```
outliers = rt[outliers_lower | outliers_upper].index
rt.drop(outliers, inplace = True)
```

In [36]:
```
rt.describe()
```

Out[36]:

	runtime
count	1455.000000
mean	101.670790
std	15.849258
min	57.000000
25%	91.000000
50%	100.000000
75%	112.000000
max	148.000000

In [ ]:
```

```

Analysis & Visualization

In [37]:
```
# Extract a random sample form the dataset
rev = reviews.sample(n = 300)
rev.isnull().sum()
gip = pd.to_numeric(budgets.release_date.unique())
y_axis = round(((budgets.worldwide_gross) / 100000000))
l_axis = round(((budgets.production_budget) / 100000000))
bip = pd.to_numeric(budgets.release_date)
```

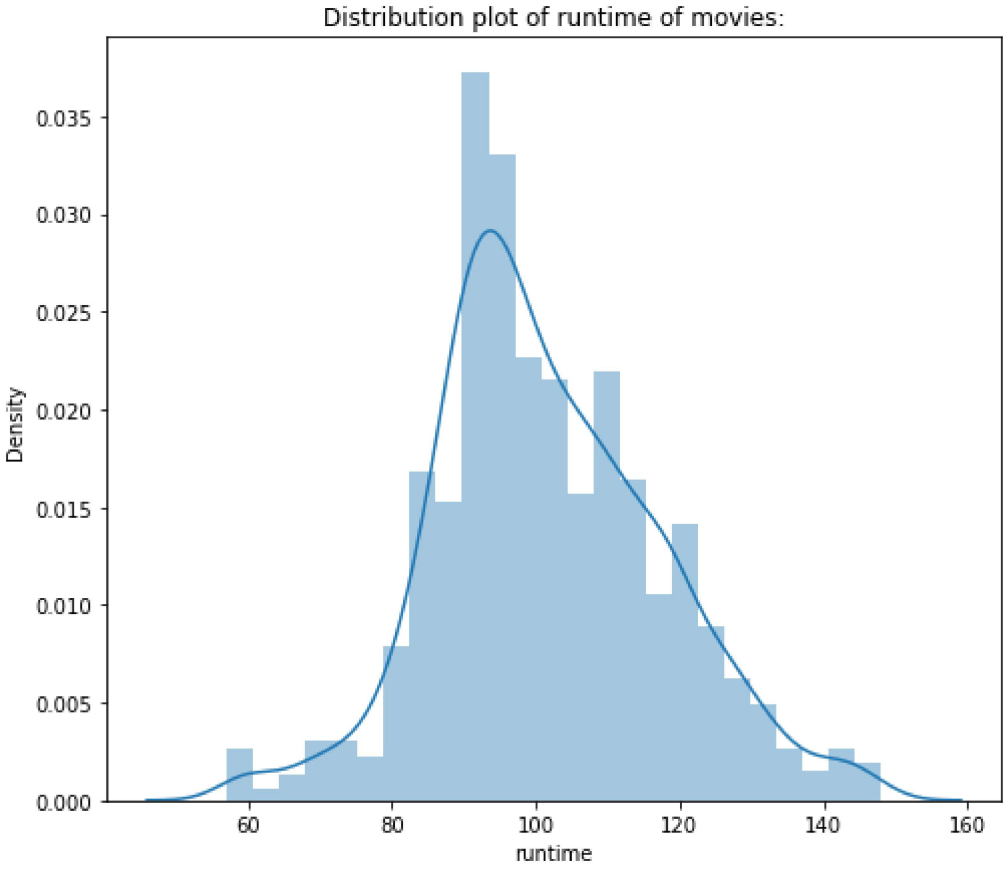
In [38]:
```
y_axis
```

Out[38]:
```
0       28.0
1       10.0
2        1.0
3       14.0
4       13.0
       ...
5777     0.0
5778     0.0
5779     0.0
5780     0.0
```

```
5781      0.0
Name: worldwide_gross, Length: 5782, dtype: float64
```

In [39]:
```
# Distibution of runtime data
import seaborn as sns
plt.figure(figsize=(8,7))
sns.distplot(rt['runtime']);
plt.title("Distribution plot of runtime of movies:");
# plt.savefig('rt.runtime distribution.jpg')
plt.show()
```

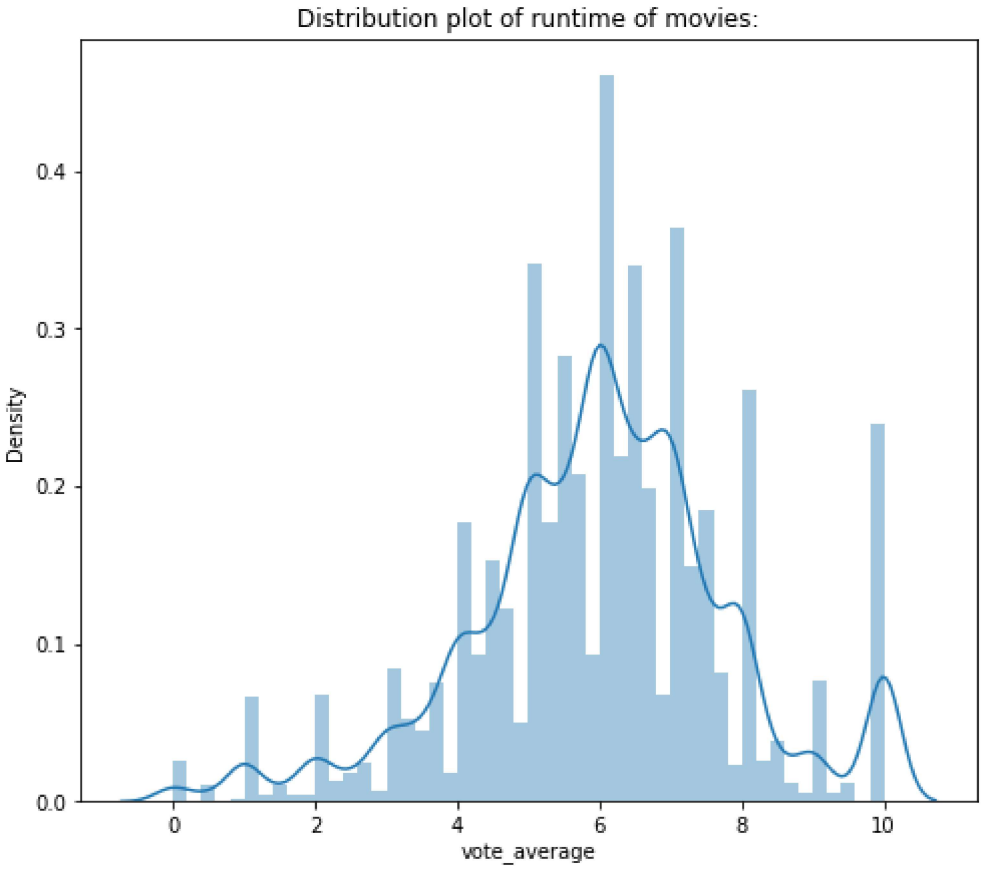
```
C:\Users\kagir\anaconda3\envs\learn-env\lib\site-packages\seaborn\distributions.py:2551:
FutureWarning: `distplot` is a deprecated function and will be removed in a future versio
n. Please adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [ ]:
```
```

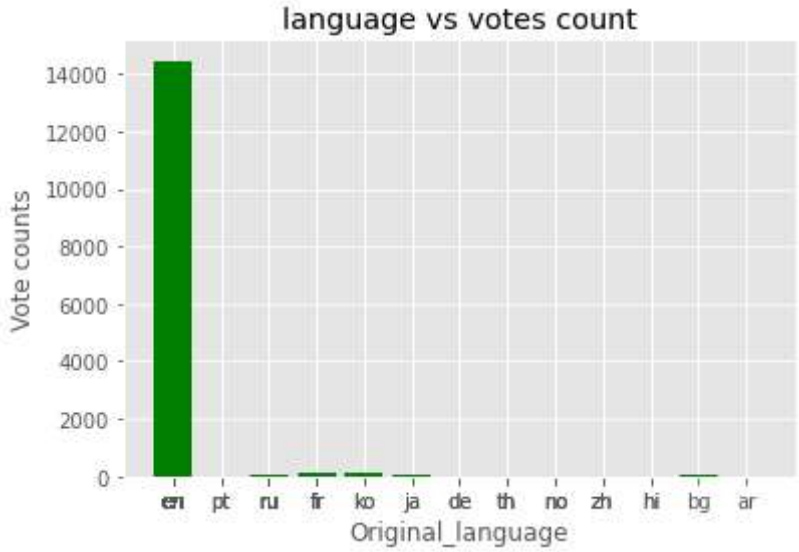
In [40]:
```
# Distribution of Average votes
import seaborn as sns
plt.figure(figsize=(8,7))
sns.distplot(reviews['vote_average']);
plt.title("Distribution plot of runtime of movies:");
# plt.savefig('reviews.vote average distrbution.jpg')
plt.show()
```

```
C:\Users\kagir\anaconda3\envs\learn-env\lib\site-packages\seaborn\distributions.py:2551:
FutureWarning: `distplot` is a deprecated function and will be removed in a future versio
n. Please adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



Analysis based on Original language of the movie

In [41]:
```
lang = (rev.original_language)
# vote = rev.vote_average
count = (reviews.vote_count.sample(300))
# fig, ax1 = plt.subplots(figsize=(8, 8))
# ax2 = ax1.twinx()

# import matplotlib.pyplot as plt
# %matplotlib inline
plt.style.use('ggplot')

x = lang


x_pos = [i for i, _ in (x)]

plt.bar(x_pos, count, color='green')
plt.xlabel("Original_language")
plt.ylabel("Vote counts")
plt.title("language vs votes count")

plt.xticks(x_pos, x)
# plt.savefig('language vs votecount.jpg', dpi = 100)
plt.show()
```



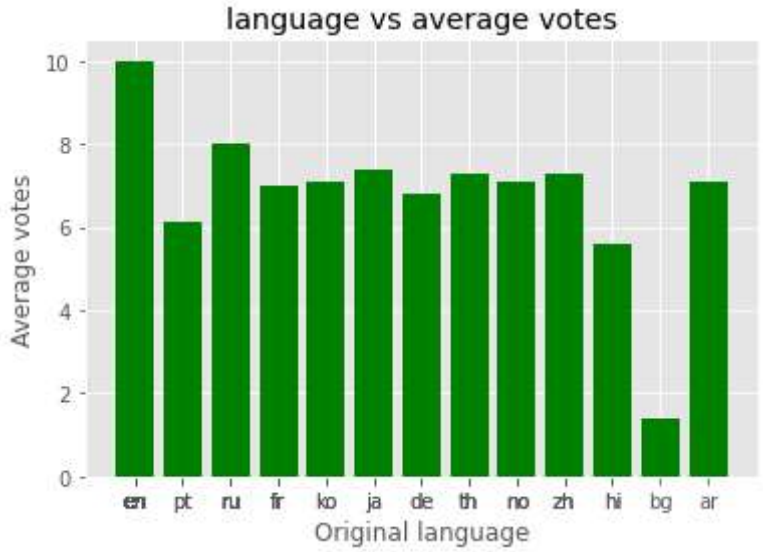
'''

Here we can see that movies made in English are more appealing to the target audience compared to other languages.

'''

In [ ]:

In [42]:
```
lang = (rev.original_language)
vote = rev.vote_average
count = reviews.vote_count
fig, ax1 = plt.subplots(figsize=(8, 8))
ax2 = ax1.twinx()

import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('ggplot')

x = lang


x_pos = [i for i, _ in (x)]

plt.bar(x_pos, vote, color='green')
plt.xlabel("Original language")
plt.ylabel("Average votes")
plt.title("language vs average votes")

plt.xticks(x_pos, x)
# plt.savefig('language vs average vote.png')
plt.show()
```



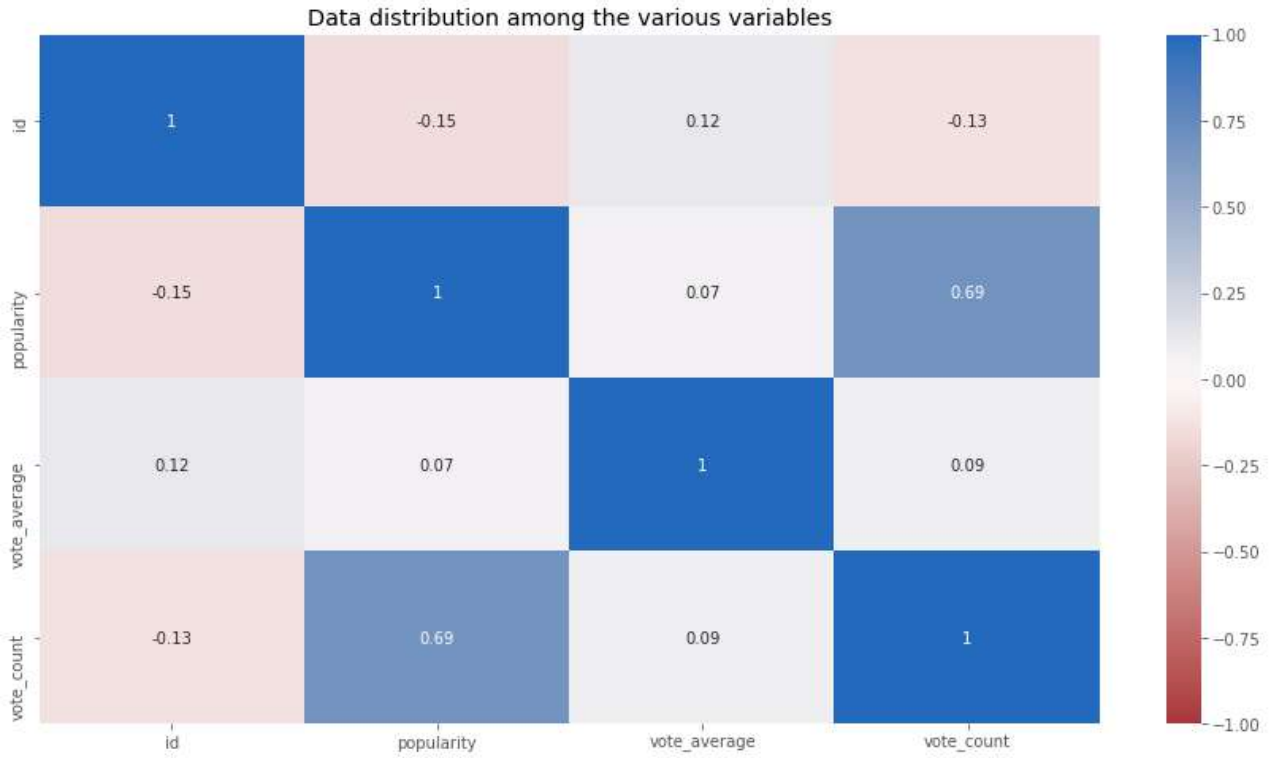
In [43]:
```
'''
Based on the above analysis we see that the average votes conducted
are more favourable to the English language hence movies produced in English
have a high approval among the target audience.
'''
```

Out[43]:
```
'\nBased on the above analysis we see that the average votes conducted\nare more favourab
le to the English language hence movies produced in English \nhave a high approval among
the target audience.\n'
```

In [ ]:

Correlation of the data

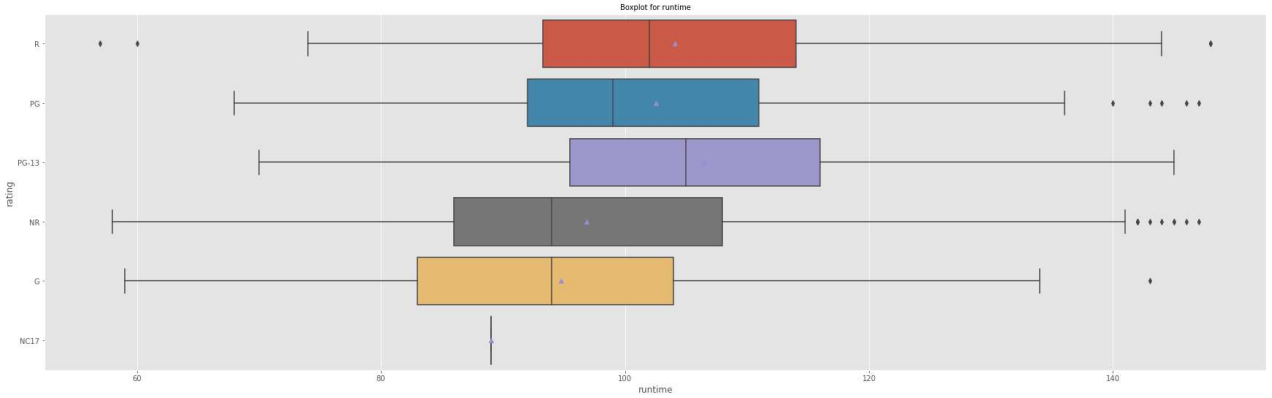
In [44]:
```
correlation = reviews.corr().round(2) # creating a 2-D Matrix with correlation plots
correlation
plt.figure(figsize = (15,8))
plt.title("Data distribution among the various variables")
sns.heatmap(correlation, annot=True, cmap='vlag_r', vmin=-1, vmax=1);
plt.savefig('correlation.jpg')

#color palette
```



Analysis based on Runtime

In [ ]:

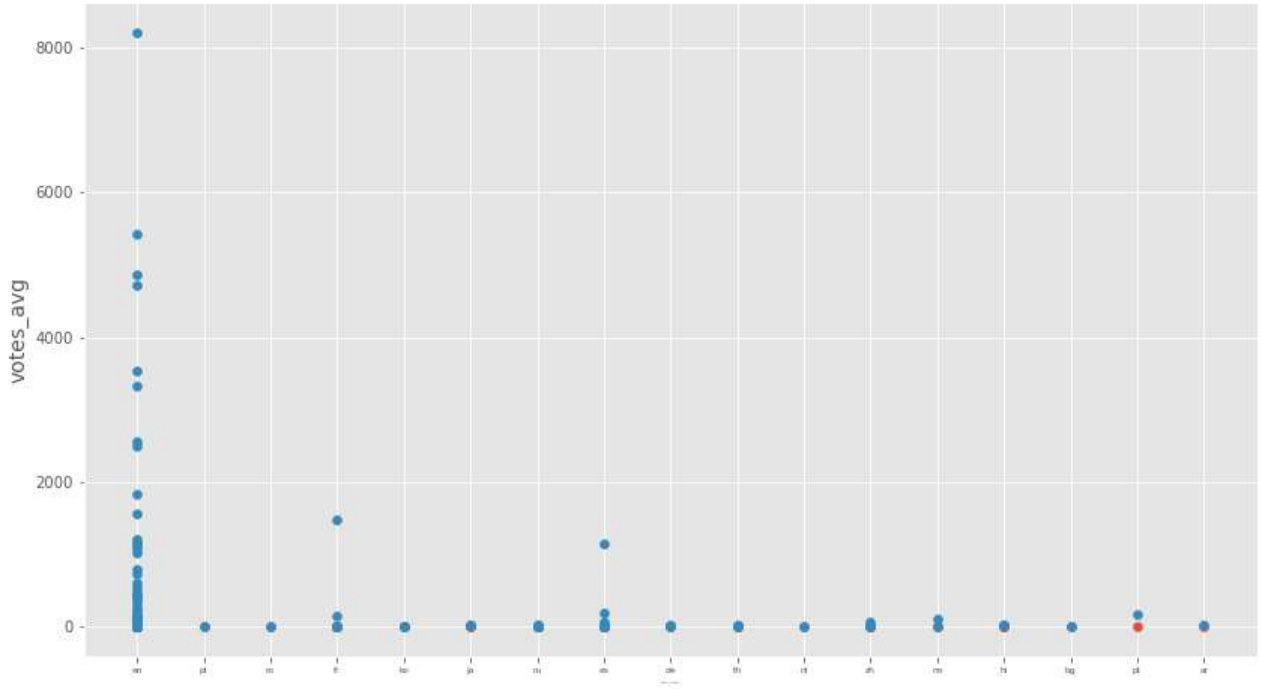
In [45]:
```
boxprops = dict(linestyle='--', linewidth=3, color='darkgoldenrod')
flierprops = dict(marker='o', markerfacecolor='green', markersize=12,
                  markeredgecolor='none')
medianprops = dict(linestyle='-.', linewidth=2.5, color='firebrick')
meanpointprops = dict(marker='D', markeredgecolor='black',
                      markerfacecolor='firebrick')
meanlineprops = dict(linestyle='--', linewidth=2.5, color='purple')
fs = 10
plt.figure(figsize=(30,9))
sns.boxplot(y = 'rating', x="runtime", data = rt,meanprops=meanlineprops,meanline=False,
plt.title("Boxplot for runtime", fontsize=fs)
# plt.savefig('rating vs runtime.jpg')
plt.show()
```



In [ ]:

Analysis based on Vote Count and Vote Average

In [46]:
```
import matplotlib.cm as cm
votes = rev.vote_average
counts = rev.vote_count
plt.figure(figsize=(14, 8))
plt.scatter(lang, vote, cmap = cm.brg)
plt.scatter(lang, counts, cmap = cm.brg)
plt.xlabel("votes_counts", size=2)
plt.ylabel("votes_avg", size=14)
plt.tick_params(axis='x', which='major', labelsize=6)
plt.show()
```

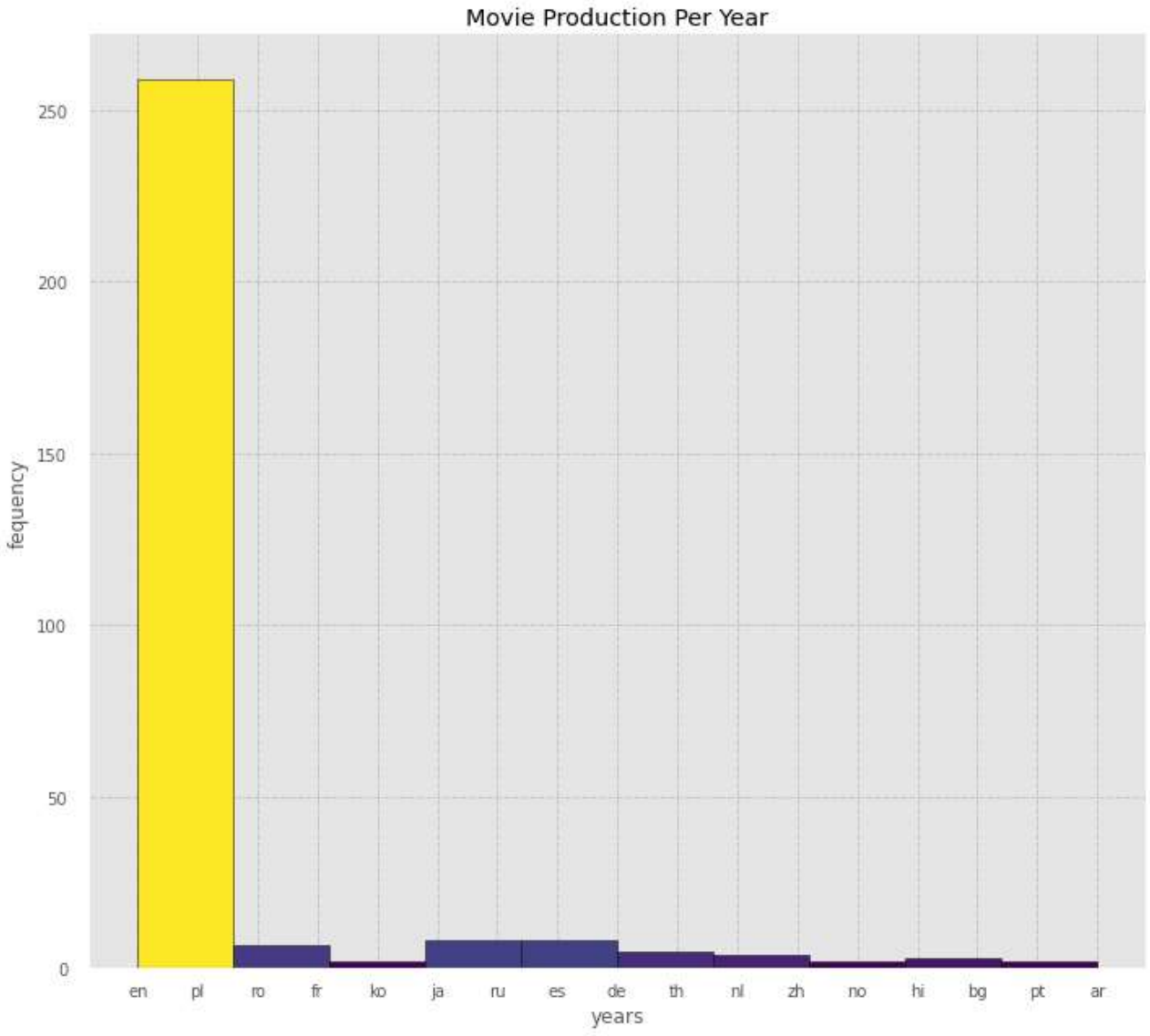


Analysis based on Time(years)

In [47]:
```
from matplotlib import colors
from matplotlib.ticker import PercentFormatter

fig, axs = plt.subplots(1,1,figsize=(10,9),tight_layout=True)
# Remove axes splines
for s in ['top', 'bottom', 'left', 'right']:
    axs.spines[s].set_visible(False)
 # Remove x, y ticks
axs.xaxis.set_ticks_position('none')
axs.yaxis.set_ticks_position('none')

# Add padding between axes and labels
axs.xaxis.set_tick_params(pad = 5)
axs.yaxis.set_tick_params(pad = 10)

# Add x, y gridlines
axs.grid(b = True, color ='grey',
        linestyle ='-.', linewidth = 0.5,
        alpha = 0.6)

N, bins, patches = axs.hist(x, edgecolor='black')
```

```
# Setting color
fracs = ((N**(1 / 5)) / N.max())
norm = colors.Normalize(fracs.min(), fracs.max())

for thisfrac, thispatch in zip(fracs, patches):
    color = plt.cm.viridis(norm(thisfrac))
    thispatch.set_facecolor(color)
axs.set_title('Movie Production Per Year')
axs.set_xlabel('years')
axs.set_ylabel('fequency')
# plt.savefig('movie production in different years.jpg')
```

Out[47]: Text(0, 0.5, 'fequency')

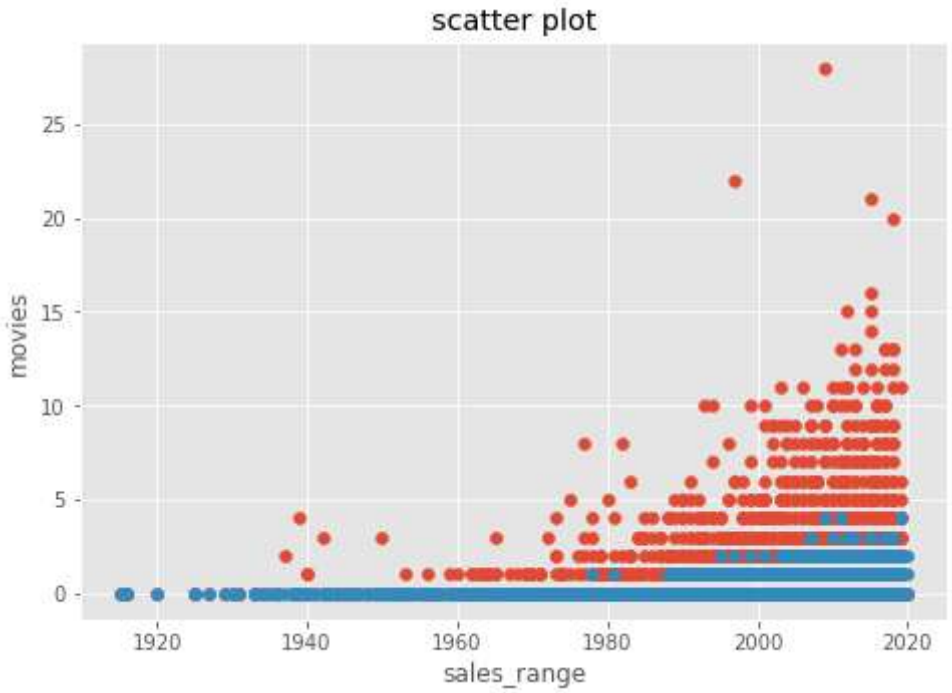


Analysis based on budget and sales

- showing the relationship between time in years and earnings(worldwide_gross) and budget(production_budget)

In [48]:
```
import matplotlib.cm as cm
y_axis = round(((budgets.worldwide_gross) / 100000000))
l_axis = round(((budgets.production_budget) / 100000000))
fig=plt.figure()
```

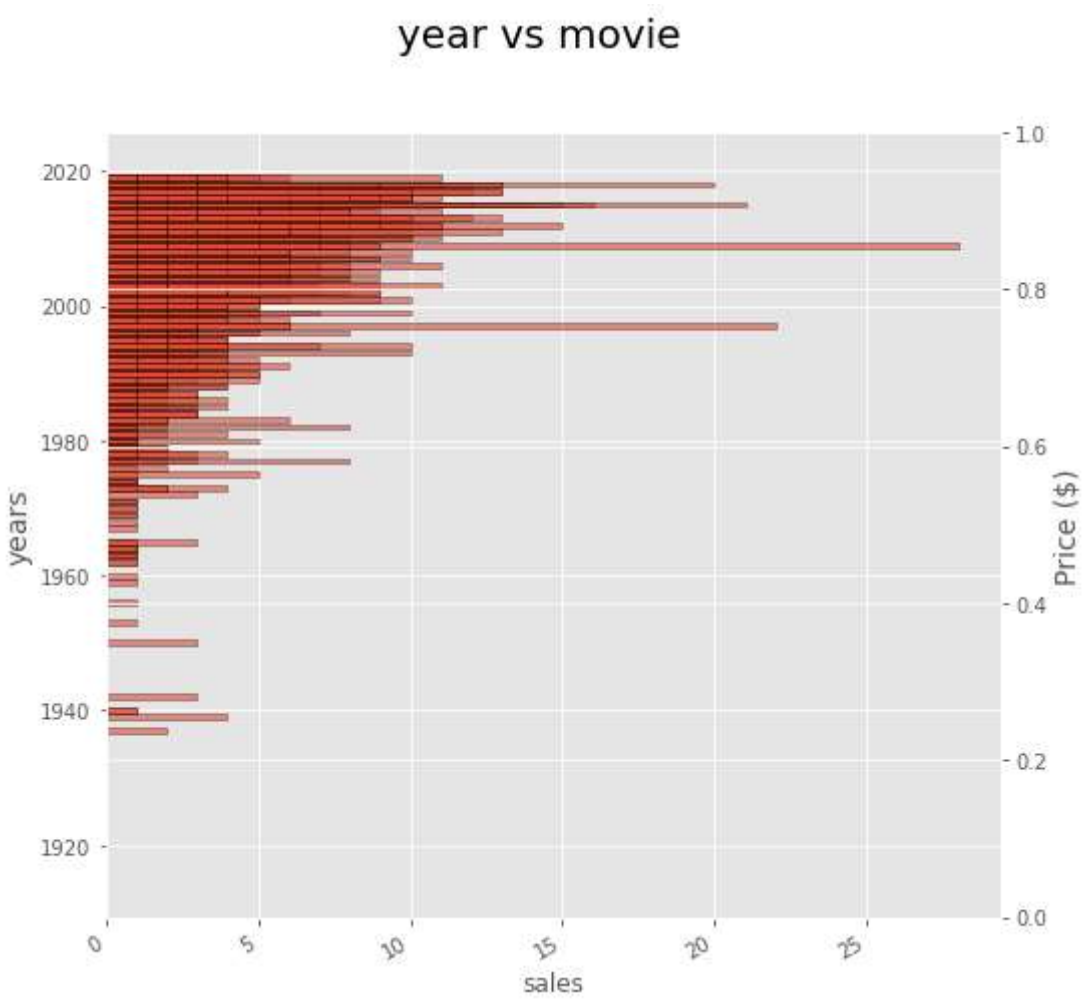
```
ax=fig.add_axes([0,0,1,1])
ax.scatter(bip, y_axis, cmap = cm.brg)
ax.scatter(bip, l_axis, cmap = cm.brg)
ax.set_xlabel('sales_range')
ax.set_ylabel('movies')
ax.set_title('scatter plot')

plt.show()
```



Analysis based on the Released Date("Year")

In [49]:

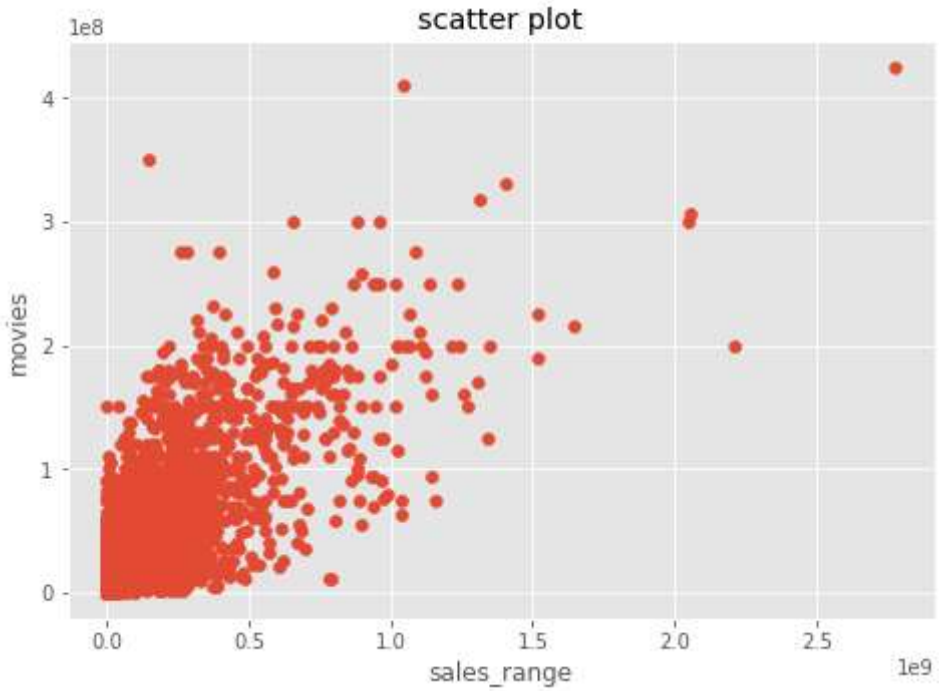
```
fig, ax1 = plt.subplots(figsize=(8, 8))
ax2 = ax1.twinx()

ax1.barh(bip, y_axis, edgecolor="black", alpha=0.6)


ax1.set_xlabel("sales")
ax1.set_ylabel("years", fontsize=14)
ax1.tick_params(axis="y")

ax2.set_ylabel("Price ($)", fontsize=14)
ax2.tick_params(axis="y")

fig.autofmt_xdate()
fig.suptitle("year vs movie", fontsize=20);
```



Distribution of worldwide gross revenue and production budget

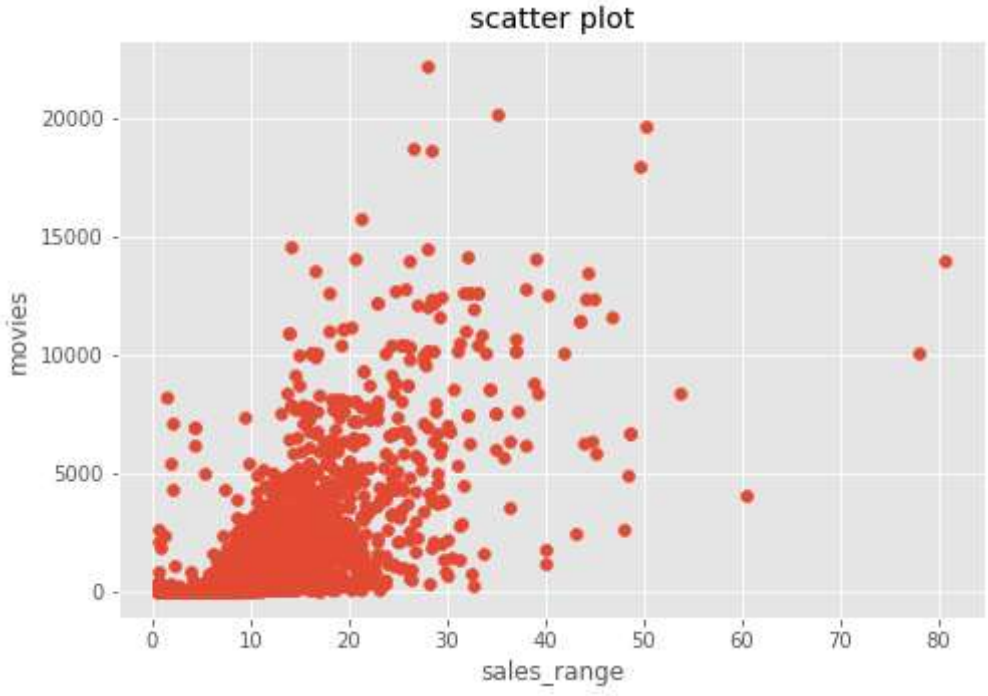
In [50]:

```
import matplotlib.cm as cm
y_axis = round(((budgets.worldwide_gross) / 100000000))
l_axis = round(((budgets.production_budget) / 100000000))
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
ax.scatter(budgets.worldwide_gross, budgets.production_budget, cmap = cm.brg)
# ax.scatter(bip, l_axis, cmap = cm.brg)
ax.set_xlabel('sales_range')
ax.set_ylabel('movies')
ax.set_title('scatter plot')
# plt.savefig('production_budget vs worldwide_gross.jpg')
plt.show()
```



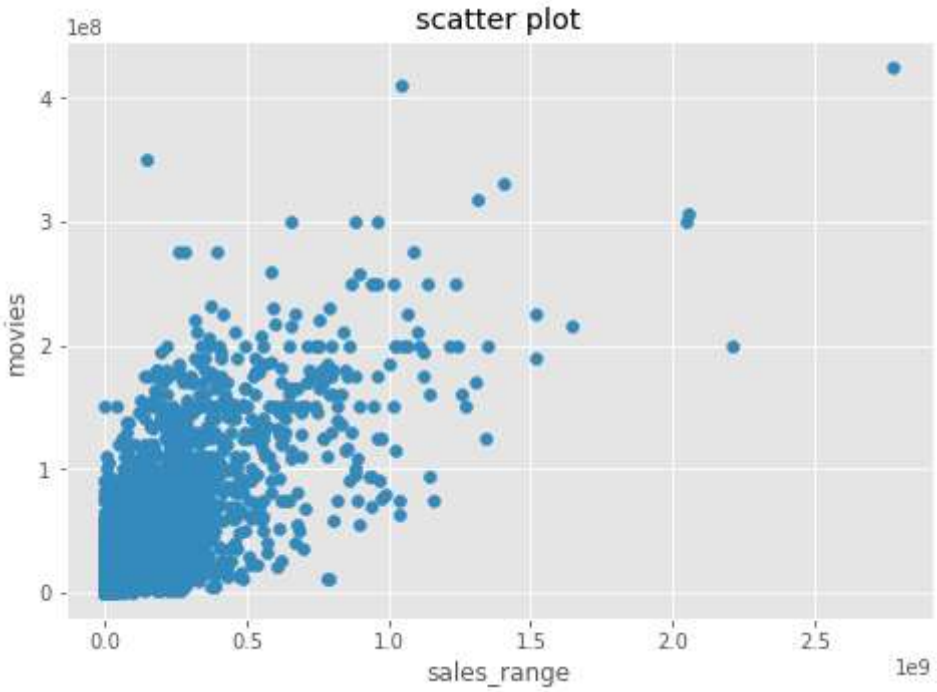
Relationship between popularity and vote counts

In [51]:
```
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
ax.scatter(reviews.popularity, reviews.vote_count, cmap = cm.brg)
# ax.scatter(bip, l_axis, cmap = cm.brg)
ax.set_xlabel('sales_range')
ax.set_ylabel('movies')
ax.set_title('scatter plot')
# plt.savefig('relationship between popularity and vote_count.jpg')

plt.show()
```



In [52]:
```
fig=plt.figure()
```

```
ax=fig.add_axes([0,0,1,1])
ax.scatter(reviews.popularity, reviews.vote_count, cmap = cm.brg)
ax.scatter(budgets.worldwide_gross, budgets.production_budget, cmap = cm.brg)
# ax.scatter(bip, l_axis, cmap = cm.brg)
ax.set_xlabel('sales_range')
ax.set_ylabel('movies')
ax.set_title('scatter plot')
# plt.savefig('relationship between popularity and vote_count.jpg')

plt.show()
```



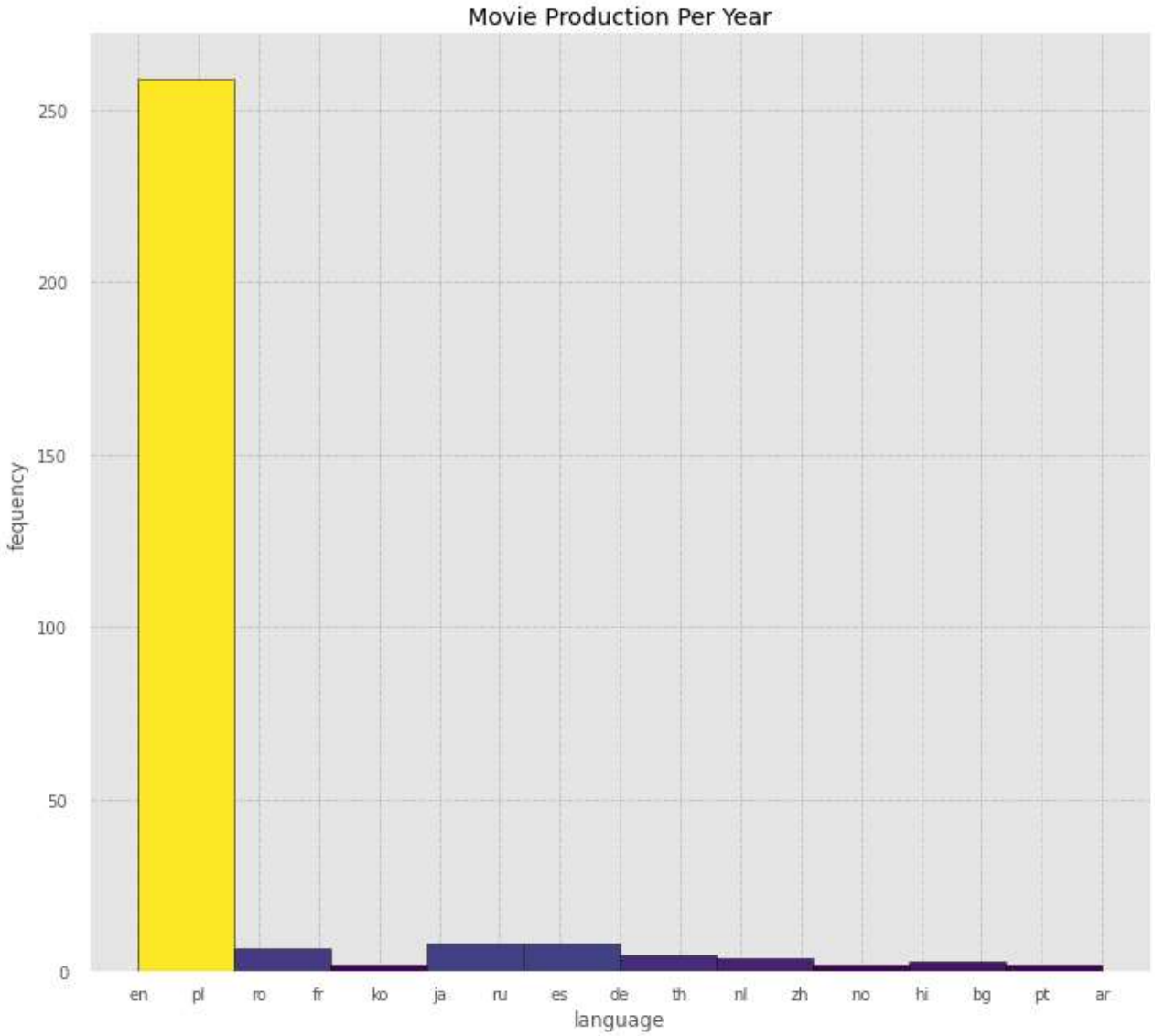
Distribution of Languages based on frequency

In [53]:

```
from matplotlib.ticker import PercentFormatter

fig, axs = plt.subplots(1,1,figsize=(10,9),tight_layout=True)
# Remove axes splines
for s in ['top', 'bottom', 'left', 'right']:
    axs.spines[s].set_visible(False)
 # Remove x, y ticks
axs.xaxis.set_ticks_position('none')
axs.yaxis.set_ticks_position('none')

# Add padding between axes and labels
axs.xaxis.set_tick_params(pad = 5)
axs.yaxis.set_tick_params(pad = 10)

# Add x, y gridlines
axs.grid(b = True, color ='grey',
        linestyle ='-.', linewidth = 0.5,
        alpha = 0.6)

N, bins, patches = axs.hist(x, edgecolor='black')
# Setting color
fracs = ((N**(1 / 5)) / N.max())
norm = colors.Normalize(fracs.min(), fracs.max())
```

```
for thisfrac, thispatch in zip(fracs, patches):
    color = plt.cm.viridis(norm(thisfrac))
    thispatch.set_facecolor(color)
axs.set_title('Movie Production Per Year')
axs.set_xlabel('language')
axs.set_ylabel('fequency')
# plt.savefig('movie production in different years.jpg')
```

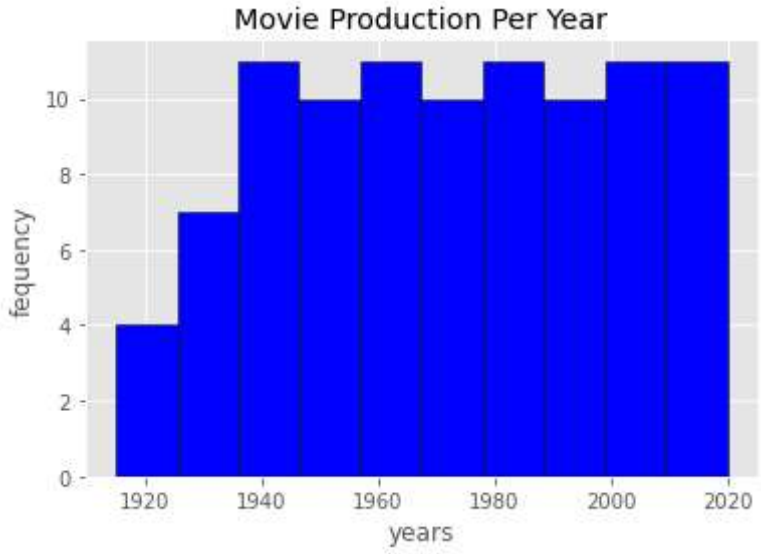
Out[53]: Text(0, 0.5, 'fequency')



Movie Production Per Year

In [54]:
```
x = gip
fig, ax = plt.subplots()
ax.hist(x, edgecolor='black', color = 'blue')
ax.set_title('Movie Production Per Year')
ax.set_xlabel('years')
ax.set_ylabel('fequency')
```

Out[54]: Text(0, 0.5, 'fequency')



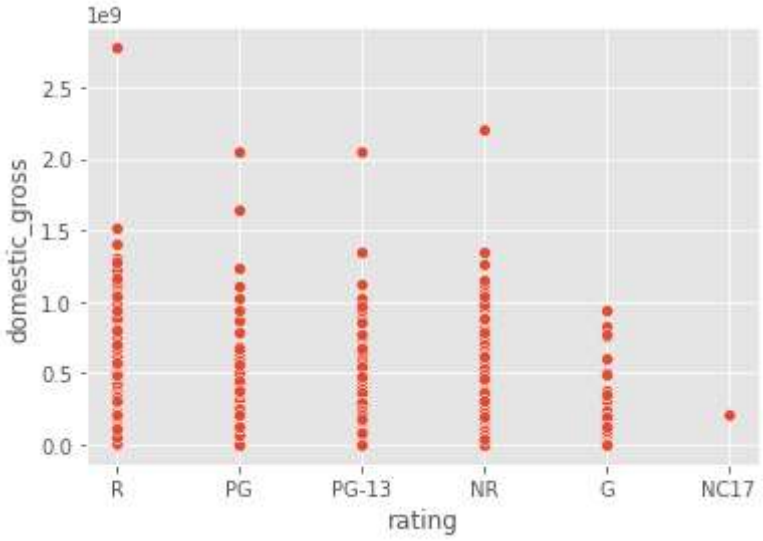
Rating vs vote count

In [57]:
```
unique = rt.rating.unique()
x_pos = np.arange(len(unique))
count = counts.sample(7)
# Create bars
plt.bar(x_pos,count)

# Create names on the x-axis
plt.xticks(x_pos, unique)

#
plt.xlabel('Rating', fontweight='bold', color = 'orange', fontsize='18')
# plt.savefig('ratings of movies depending on vote counts')

# Show graphic
plt.show()
```

Analysis Based on the Relationship between language, domestic gross and worldwide gross

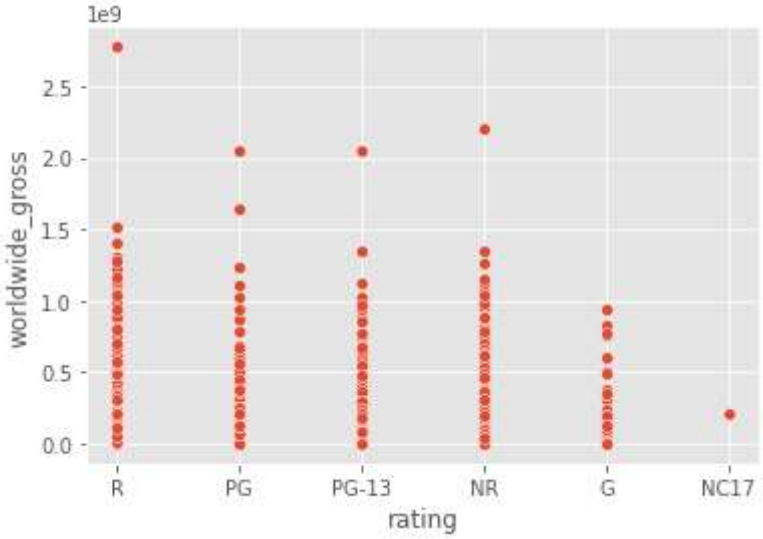
In [58]:
```
sns.scatterplot(x = rt.rating, y = budgets.domestic_gross)
```

Out[58]: <AxesSubplot:xlabel='rating', ylabel='domestic_gross'>

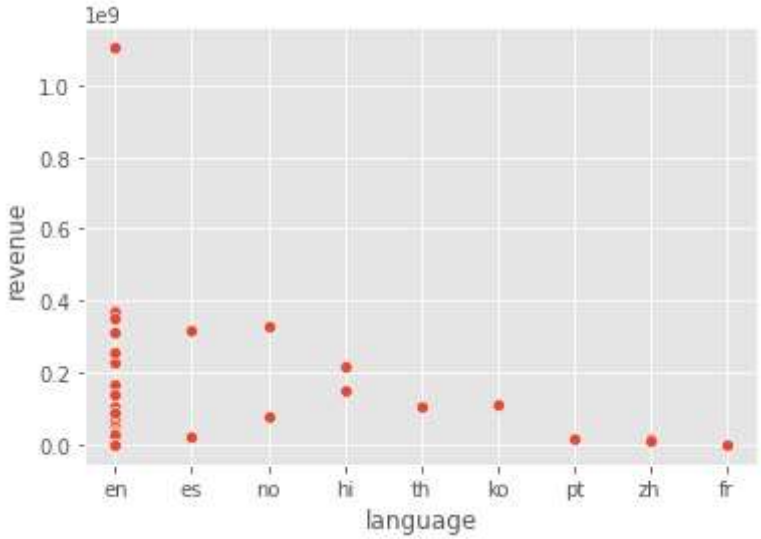


In [59]:
```
sns.scatterplot(x = rt.rating, y = budgets.worldwide_gross)
```

Out[59]: <AxesSubplot:xlabel='rating', ylabel='worldwide_gross'>

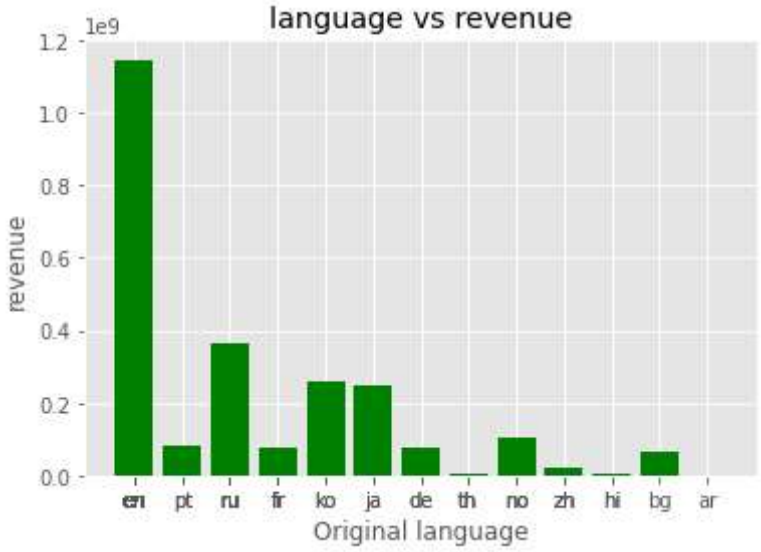


In [60]:
```
sns.scatterplot(x = lang, y = budgets.domestic_gross)
plt.xlabel('language')
plt.ylabel('revenue')
# plt.savefig('lang vs rev.jpg')
plt.show()
```



In [61]:

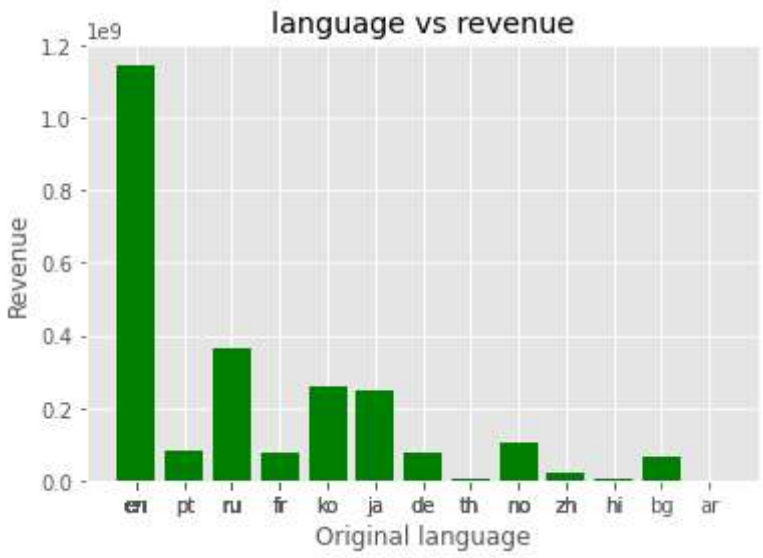
```
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('ggplot')
yes = budgets.domestic_gross.sample(300)
x = lang


x_pos = [i for i, _ in (x)]

plt.bar(x_pos, yes, color='green')
plt.xlabel("Original language")
plt.ylabel("revenue")
plt.title("language vs revenue")

plt.xticks(x_pos, x)
# plt.savefig('revenue vs language.jpg')
plt.show()
```



In [62]:

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('ggplot')
bull = budgets.worldwide_gross.sample(300)
x = lang
```

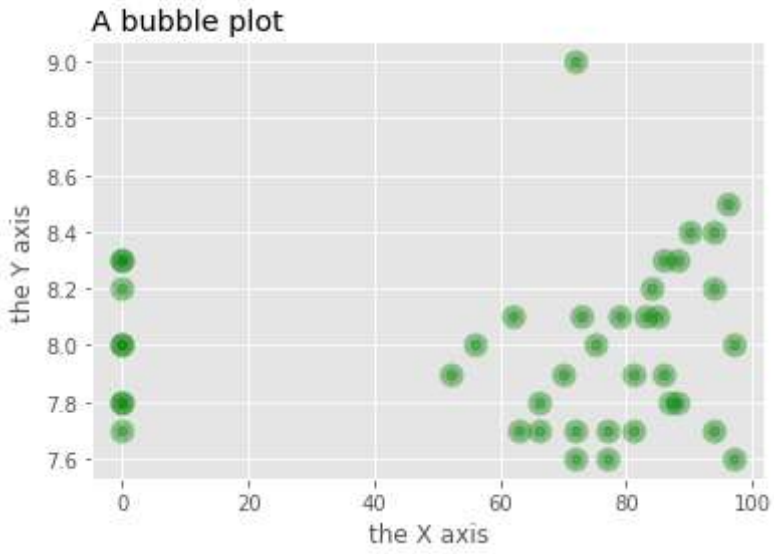
```
x_pos = [i for i, _ in (x)]

plt.bar(x_pos, yes, color='green')
plt.xlabel("Original language")
plt.ylabel("Revenue")
plt.title("language vs revenue")

plt.xticks(x_pos, x)
# plt.savefig('worlwide gorss vs language.jpg')
plt.show()
```



Relationship between rating and meta score

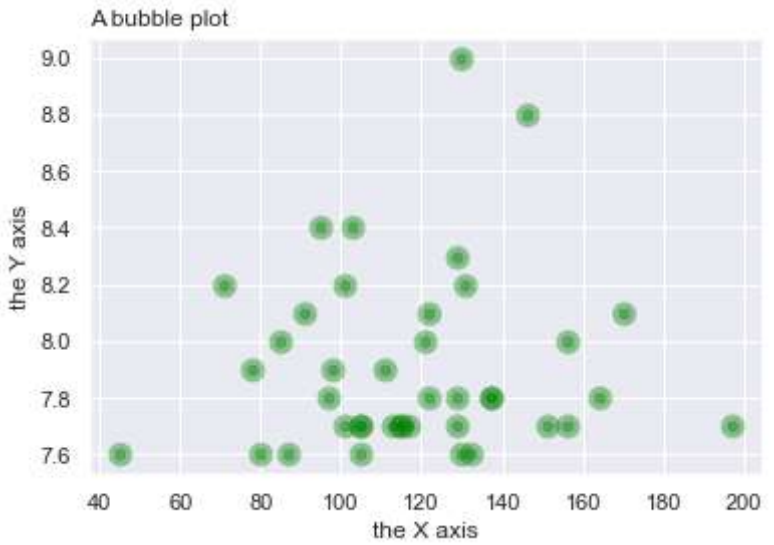
In [63]:

```
y = net.IMDB_Rating.sample(40)
x = net.Meta_score.sample(40)
# plot
plt.scatter(x, y, c="green", alpha=0.4, linewidth=6)

# Add titles (main and on axis)
plt.xlabel("the X axis")
plt.ylabel("the Y axis")
plt.title("A bubble plot", loc="left")
plt.savefig('bubble chart rating vs metascore.jpg')
# show the graph
plt.show()
```



Relationship between rating and runtime

In [64]:
```
# pimp your plot with the seaborn style
import seaborn as sns
sns.set_theme()
y = net.IMDB_Rating.sample(40)
x = net.Runtime.sample(40)
# plot
plt.scatter(x, y, c="green", alpha=0.4, linewidth=6)

# Add titles (main and on axis)
plt.xlabel("the X axis")
plt.ylabel("the Y axis")
plt.title("A bubble plot", loc="left")
plt.savefig(' bubble plot rating vs runtime.jpg')
# show the graph
plt.show()
```



In [ ]: