

Визуализация данных

В самом начале лирическое отступление. Вы, наверное, уже видели, в каких красивых блокнотах мы отдаем вам файлы. Хорошая новость - делать их несложно. Плохая новость - вам придется в таком же виде сдать нам задание #4. Поэтому крошечный тьюториал.

Чтобы поменять тип ячейки на markdown (размеченный текст) - нужно выбрать markdown для конкретной ячейке в выпадающей строке наверху файла (по умолчанию там стоит code). И все - теперь мы можем писать здесь текст, а при исполнении ячейки он станет как будто частью файла (если кликнуть на такую ячейку два раза - то ее опять можно будет редактировать).

Markdown поддерживает разметку файла - а значит, вы можете использовать теги, менять параметры текста, вставлять картинки. Мы ограничимся базовыми параметрами текста (сделайте эту ячейку редактируемой, чтобы увидеть невидимые символы).

▼ Вот так делаем заголовок 1 уровня

А так второго

А так третьего

Ну, вы поняли

Жирный текст

- курсив *

Список:

- раз
- два
- три

На этом достаточно :)

▼ Вернемся к визуализации

Во-первых, кроме привычных вам графиков существует еще огромное множество всего (и некоторые вещи работают гораздо лучше привычных нам). Python умеет строить и сложные штуки. Есть несколько классных сайтов, которые помогают выбрать подходящий график для ваших данных:

<https://www.data-to-viz.com/>

<https://datavizproject.com/>

<https://datavizcatalogue.com/RU/>

Сегодня будем делать упражнения по мотивам вот этого блокнота

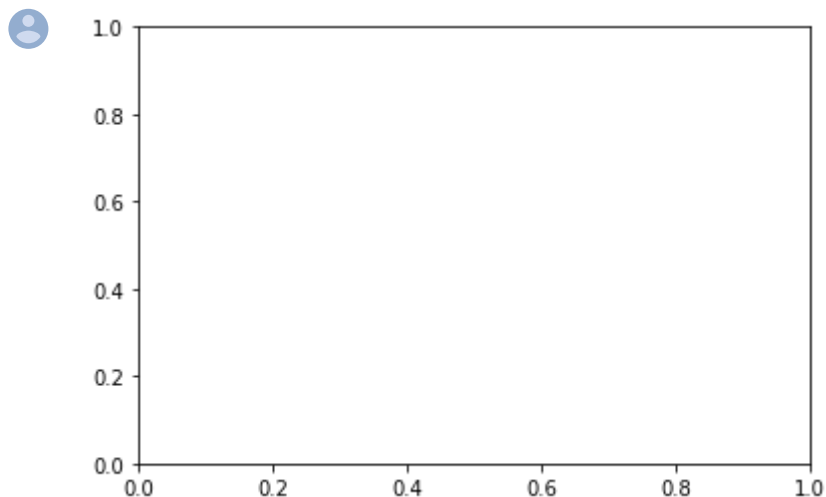
<https://nbviewer.jupyter.org/github/yaph/ipython-notebooks/blob/master/movie-body-counts.ipynb>

и работать с датасетом, который подсчитывает количество смертей в фильмах (методологию сбора данных можно посмотреть по ссылке выше).

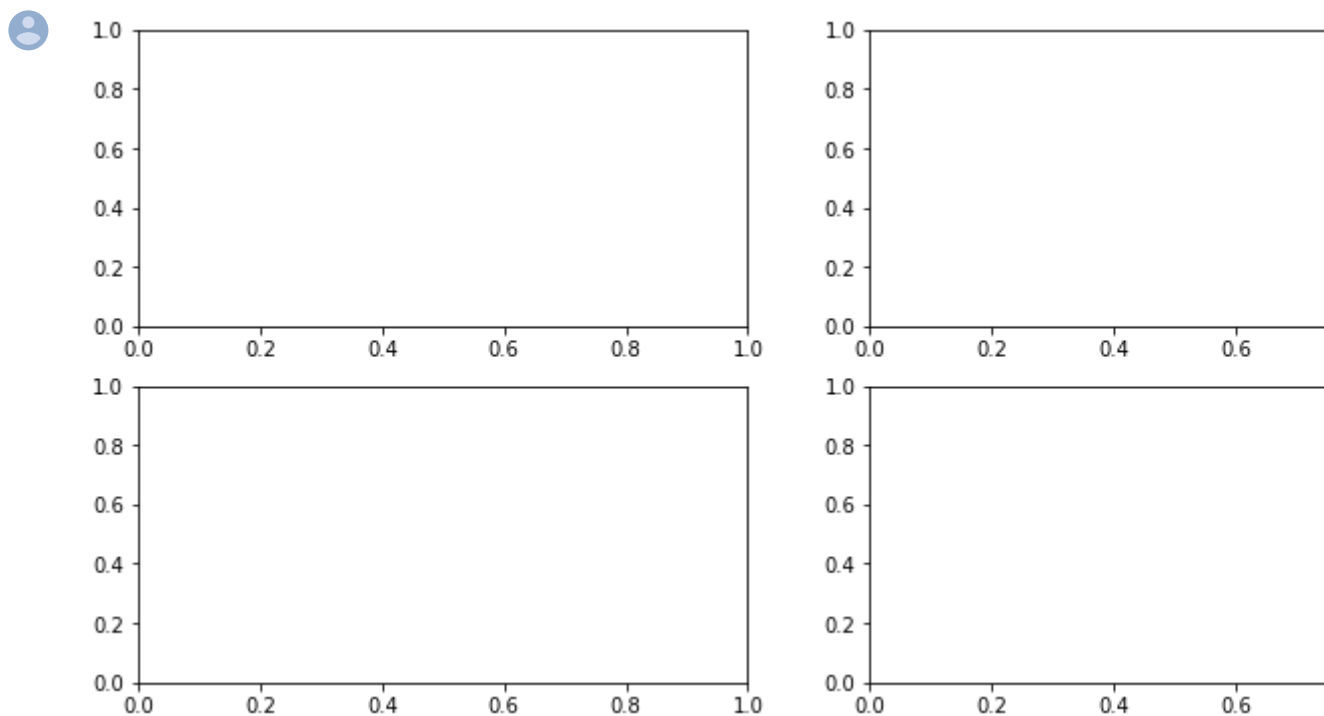
```
import os
os.chdir("C:/Users/Rogov/Desktop/Data")
os.getcwd ()
```

 'C:\\Users\\Rogov\\Desktop\\Data'

```
fig, ax= plt.subplots(1, 1)
```



```
fig, ax = plt.subplots(2, 2, figsize=(12,6))
```



```
data = pd.read_csv(r'populations.txt', sep='\t')
```

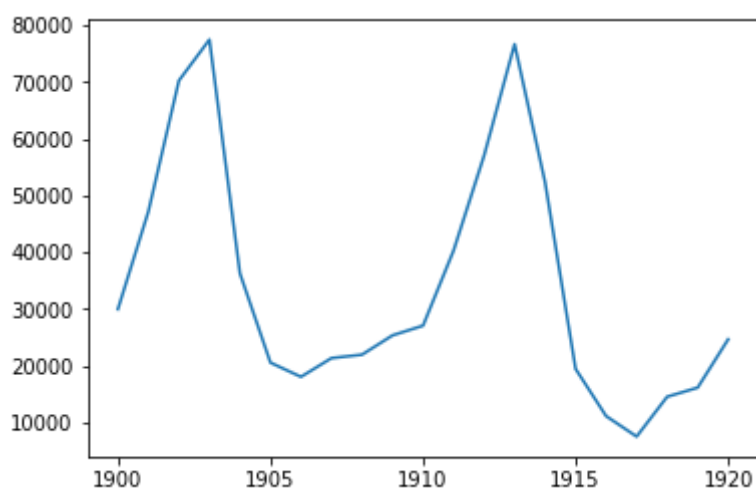
```
data.head()
```



	year	hare	lynx	carrot
0	1900	30000.0	4000.0	48300
1	1901	47200.0	6100.0	48200
2	1902	70200.0	9800.0	41500
3	1903	77400.0	35200.0	38200
4	1904	36300.0	59400.0	40600

```
data.year = data.year.apply(int)
```

```
fig, ax= plt.subplots(1, 1)  
ax.plot(data.year, data.hare)  
ax.locator_params(integer=True)
```



```
fig, ax1= plt.subplots(2, 2, figsize=(12,8))
```





```
ax1[0][0]
```

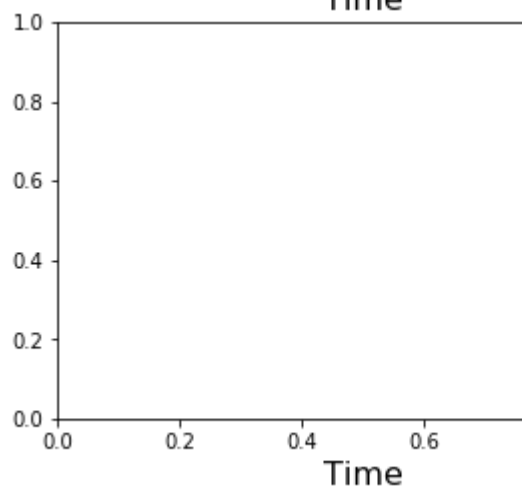
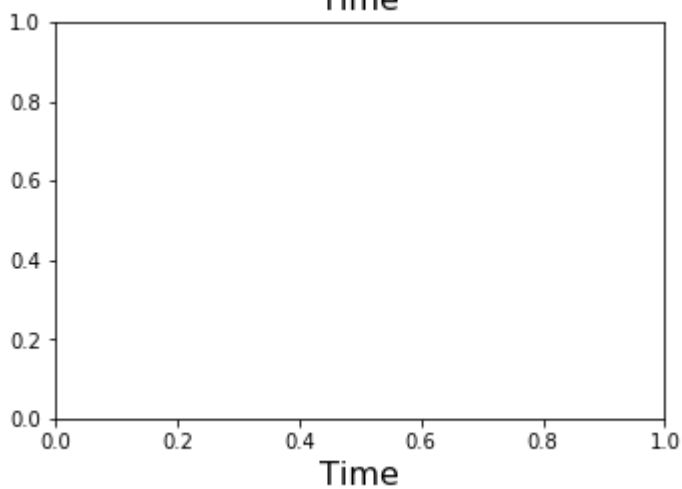
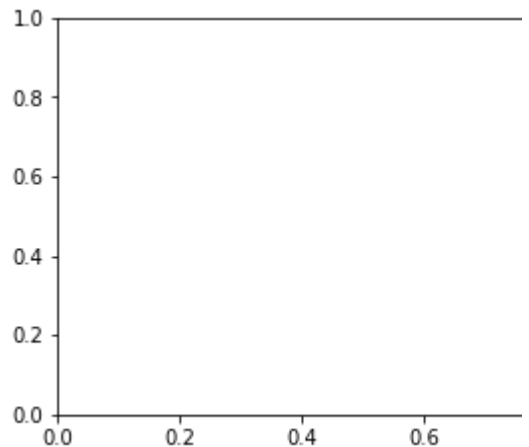
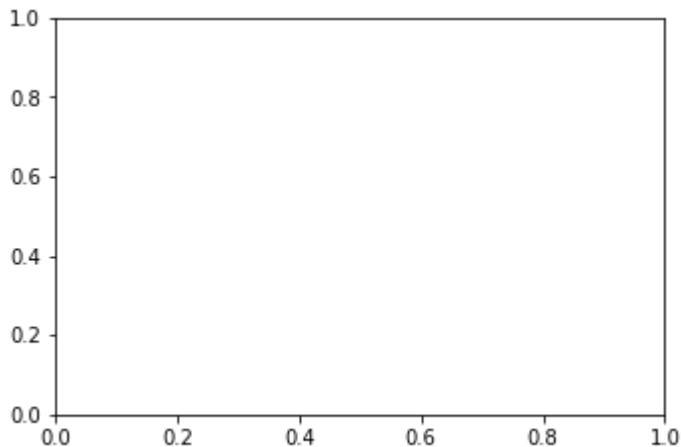


```
<matplotlib.axes._subplots.AxesSubplot at 0x1c6559bd438>
```

```
ax1[0][0]
```

```
fig, ax1= plt.subplots(2, 2, figsize=(12,8))
```

```
for x in range(2):
    for y in range(2):
        ax1[x][y].set_xlabel('Time', fontsize = 16)
```



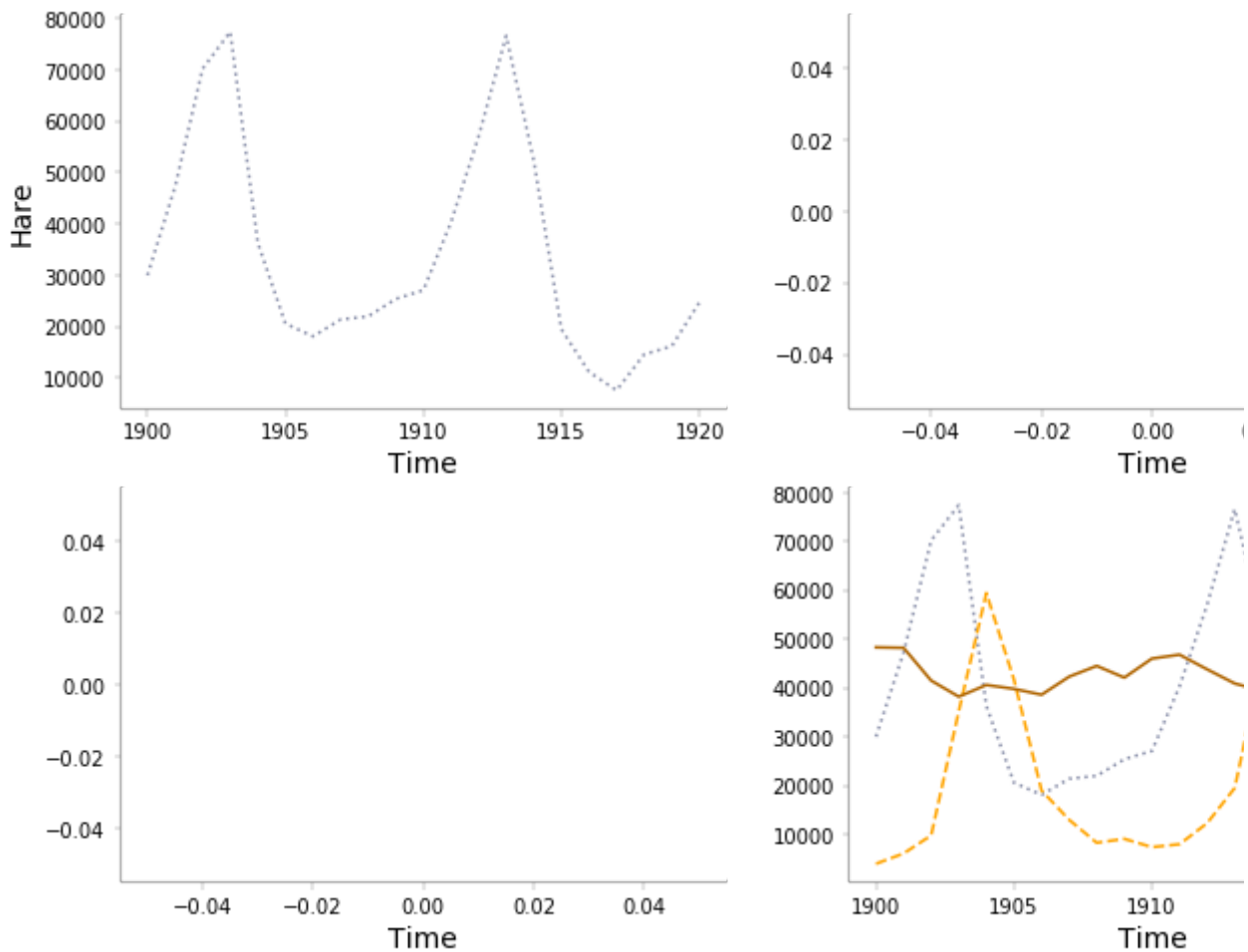
```
fig, ax1= plt.subplots(2, 2, figsize=(12,8))
```

```
for x in range(2):
    for y in range(2):
        ax1[x][y].set_xlabel('Time', fontsize=14)
        ax1[x][y].locator_params(integer=True)
        ax1[x][y].spines['right'].set_visible(False)
        ax1[x][y].spines['top'].set_visible(False)
        ax1[x][y].xaxis.set_tick_params(width=0.2)
        ax1[x][y].yaxis.set_tick_params(width=0.2)
        for axis in ['top', 'bottom', 'left', 'right']:
            ax1[x][y].spines[axis].set_linewidth(0.2)
```

```
ax1[0][0].plot(data.year, data.hare, color='#8c92ac', ls = ':')
ax1[0][0].set_ylabel('Hare', fontsize=14)
```

```
ax1[1][1].plot(data.year, data.hare, label = 'Hares', color='#8c92ac', ls = ':'); # here
ax1[1][1].plot(data.year, data.lynx, label = 'Carrots', color='#ffa500', ls = '-'); # he
ax1[1][1].plot(data.year, data.carrot, label = 'Lynxes', color='#b06500', ls = '-'); # he
ax1[1][1].legend(loc=1, fontsize=8, frameon=False) # upper left corner
```

 <matplotlib.legend.Legend at 0x1c65a897668>



```
fig.savefig("my_new_plot.png")
```

```
os.getcwd()
```

 'C:\\Users\\Rogov\\Desktop\\Data'

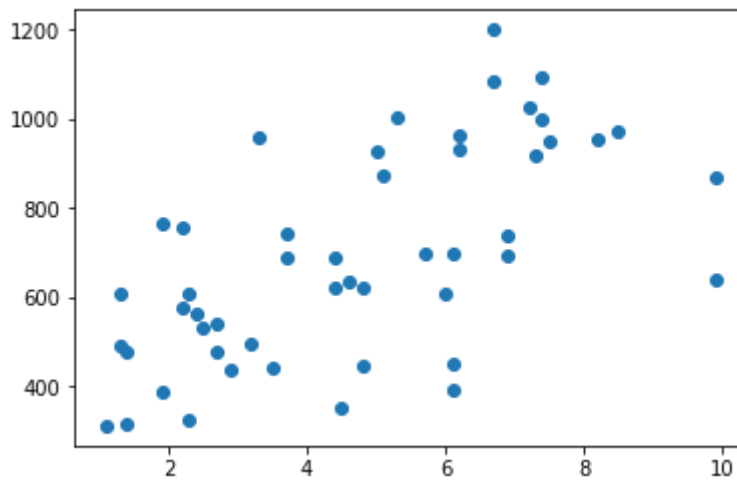
```
df = pd.read_csv('crimeRatesByState2005.tsv', header=0, sep='\\t')
```

```
df.head()
```

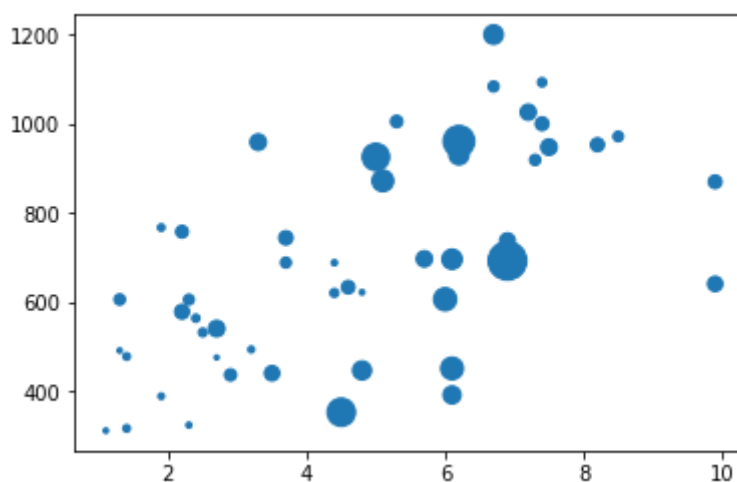


	state	murder	Forcible_rate	Robbery	aggravated_assult	burglary	larceny
0	Alabama	82	343	1414	2472	9522	

```
fig, ax= plt.subplots(1, 1)
sc=plt.scatter(df['murder'], df['burglary'])
```



```
fig, ax= plt.subplots(1, 1)
sc=plt.scatter(df['murder'], df['burglary'], s=df['population']/100000)
```



```
fig, ax= plt.subplots(1, 1)
sc=plt.scatter(df['murder'], df['burglary'], s=df['population']/100000, c=df['motor_vehic
plt.colorbar(sc)
```



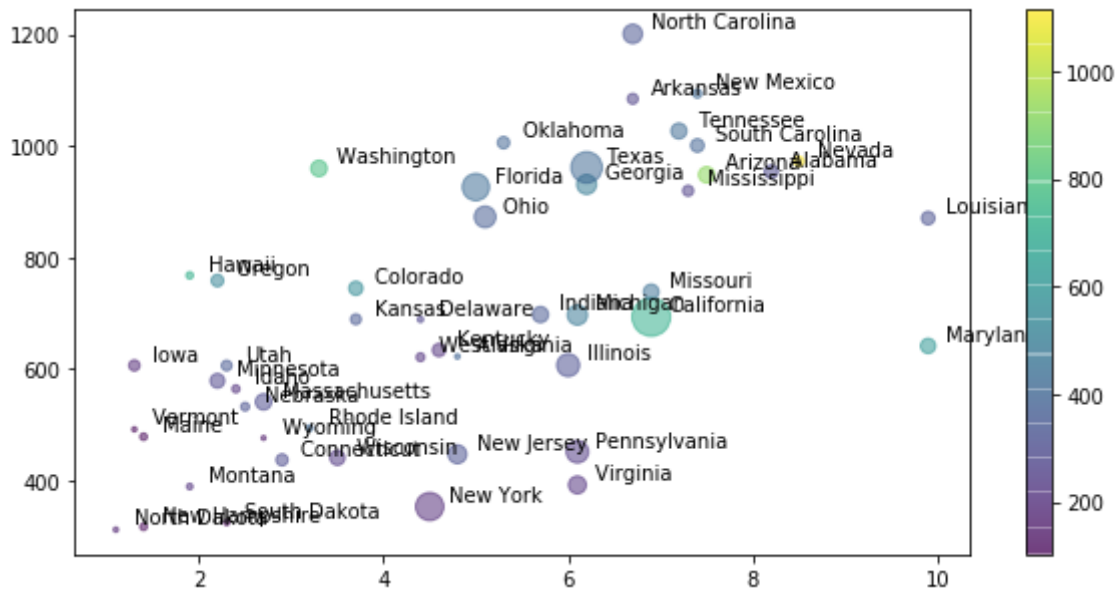
<matplotlib.colorbar.Colorbar at 0x1c65bfc7588>

```
fig, ax = plt.subplots(1, 1, figsize=(10,5))
plt.scatter(df['murder'], df['burglary'], s=df['population']/100000, c=df['motor_vehicle_
state_names = df['state']
state_data = df[df.columns[1:]]

for i in range(len(state_names)):
    ax.annotate(state_names[i], (state_data.loc[i, 'murder']+0.2, state_data.loc[i, 'burglary']))
plt.colorbar()
```



<matplotlib.colorbar.Colorbar at 0x1c65c38be80>



```
fig, ax = plt.subplots()

ax.scatter(df['murder'], df['burglary'], s = np.array(df['population']) / 30000, c = df[

ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)

ax.yaxis.set_ticks_position('left')
ax.xaxis.set_ticks_position('bottom')

ax.spines['left'].set_linewidth(0.5)
ax.spines['bottom'].set_linewidth(0.5)

ax.set_xlabel('Murder', fontsize = 10)
ax.set_ylabel('Bulglarly', fontsize = 10)

for i, state in enumerate(df['state']):
    ax.annotate(state, (df['murder'][i], df['burglary'][i]), fontsize = 5)
```



```
1200 |
url = 'https://python-graph-gallery.com/wp-content/uploads/gapminderData.csv'
data = pd.read_csv(url)

data.head()
```

	country	year	pop	continent	lifeExp	gdpPercap
0	Afghanistan	1952	8425333.0	Asia	28.801	779.445314
1	Afghanistan	1957	9240934.0	Asia	30.332	820.853030
2	Afghanistan	1962	10267083.0	Asia	31.997	853.100710
3	Afghanistan	1967	11537966.0	Asia	34.020	836.197138
4	Afghanistan	1972	13079460.0	Asia	36.088	739.981106

```
my_dpi=96

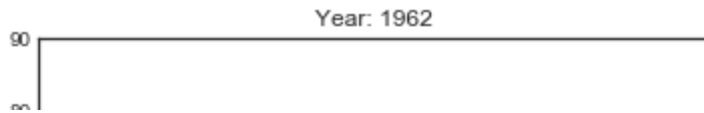
data['continent']=pd.Categorical(data['continent'])
data.head()
```

	country	year	pop	continent	lifeExp	gdpPercap
0	Afghanistan	1952	8425333.0	Asia	28.801	779.445314
1	Afghanistan	1957	9240934.0	Asia	30.332	820.853030
2	Afghanistan	1962	10267083.0	Asia	31.997	853.100710
3	Afghanistan	1967	11537966.0	Asia	34.020	836.197138
4	Afghanistan	1972	13079460.0	Asia	36.088	739.981106

```
tmp = data[data.year == 1962 ]
plt.scatter(tmp['gdpPercap'], tmp['lifeExp'], s=tmp['pop']/200000 , c=tmp['continent'].cat
plt.xscale('log')
plt.xlabel("GDP per Capita")
plt.ylabel("Life Expectancy")
plt.title("Year: "+str(1962) )
plt.ylim(30, 90)
plt.xlim(0,100000)
```



C:\Users\rogov\Anaconda3\lib\site-packages\matplotlib\axes_base.py:3129: UserWarning
 'Attempted to set non-positive xlimits for log-scale axis; '
 (355.1974801060595, 100000)



```
# For each year:
for i in data.year.unique():

    # initialize a figure
    fig = plt.figure(figsize=(680/my_dpi, 480/my_dpi), dpi=my_dpi)

    # Change color with c and alpha. I map the color to the X axis value.
    tmp = data[data.year == i ]
    plt.scatter(tmp['gdpPercap'], tmp['lifeExp'], s=tmp['pop']/200000 , c=tmp['continent'])

    plt.xscale('log')
    plt.xlabel("GDP per Capita")
    plt.ylabel("Life Expectancy")
    plt.title("Year: "+str(i) )
    plt.ylim(30, 90)
    plt.xlim(0,100000)

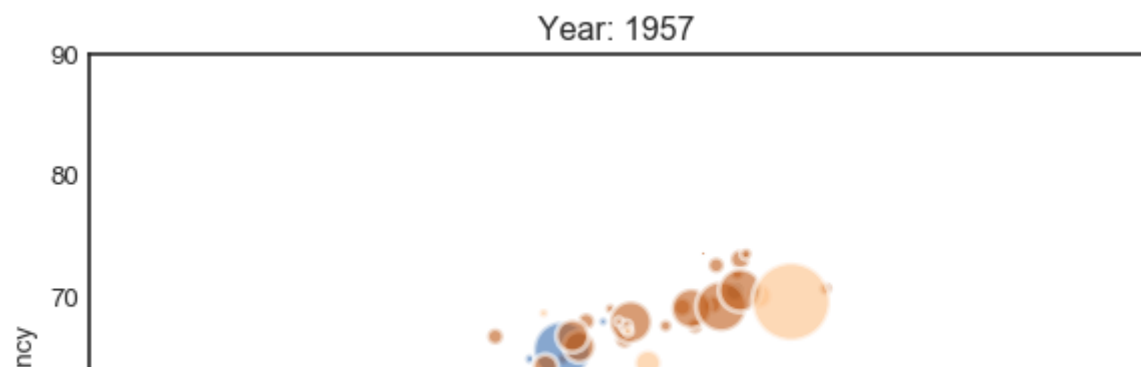
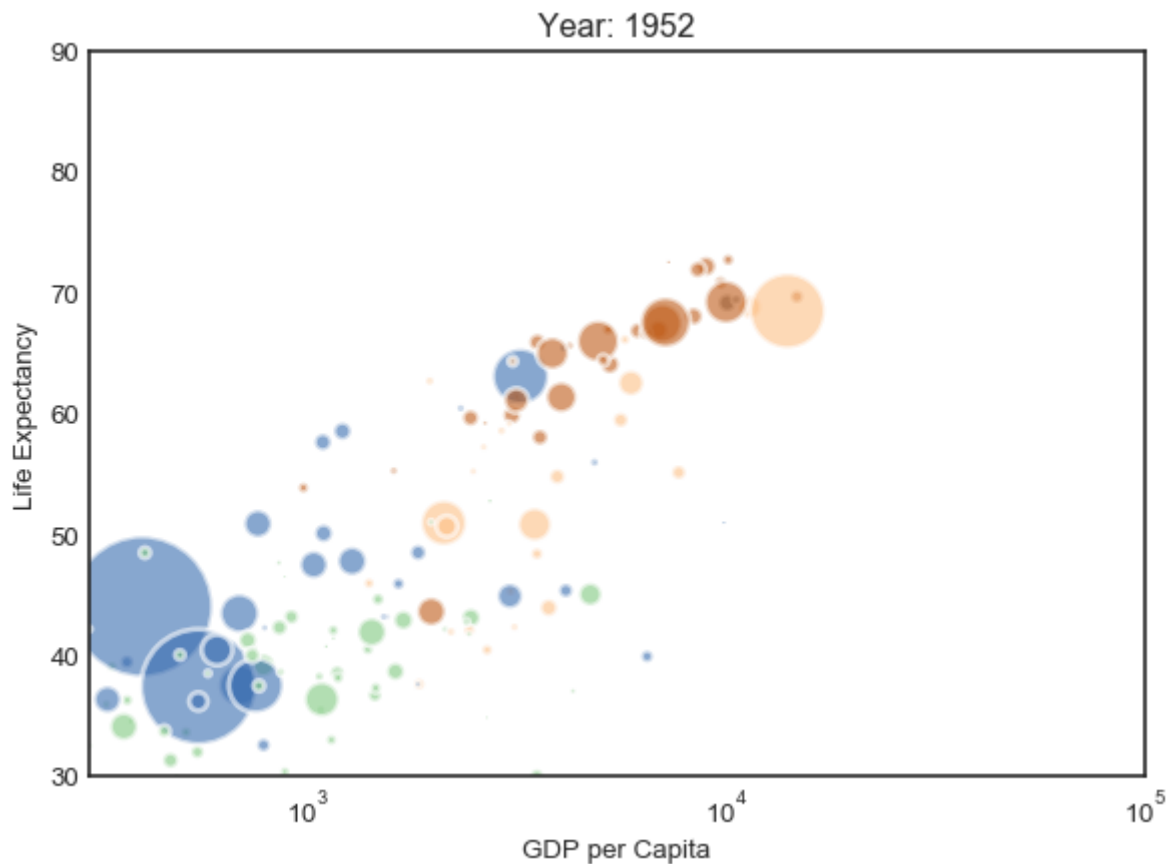
    # Save it
    filename='Gapminder_step'+str(i)+'.png'
    plt.savefig(filename, dpi=96)
    plt.gca()
```

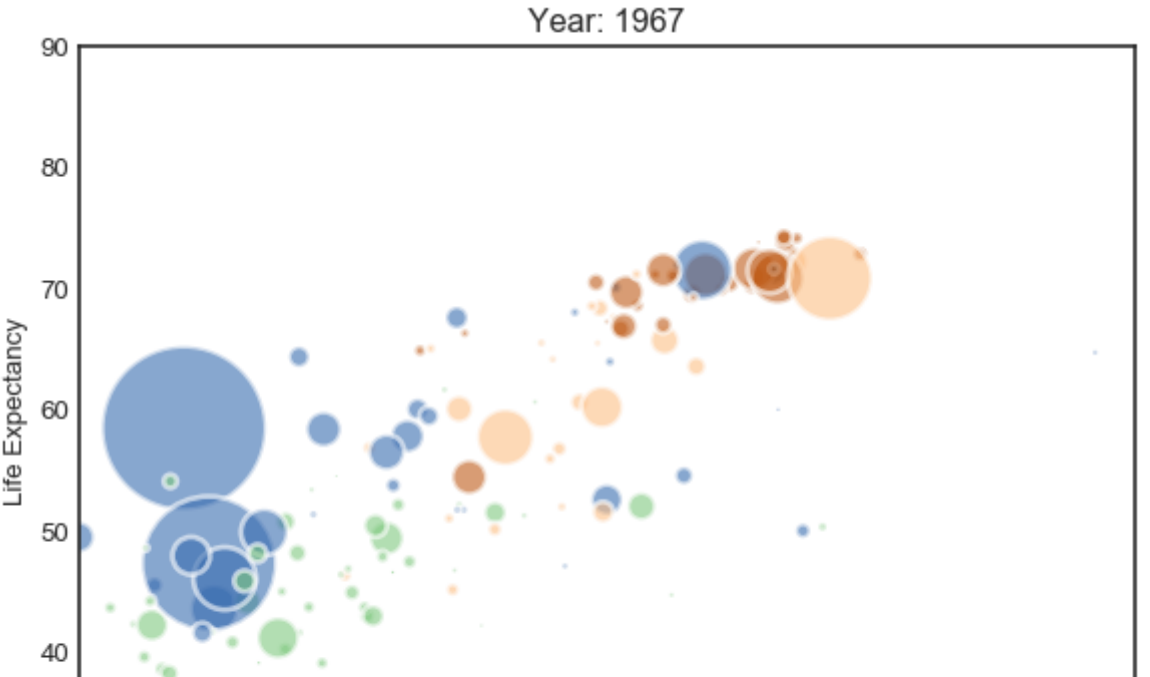
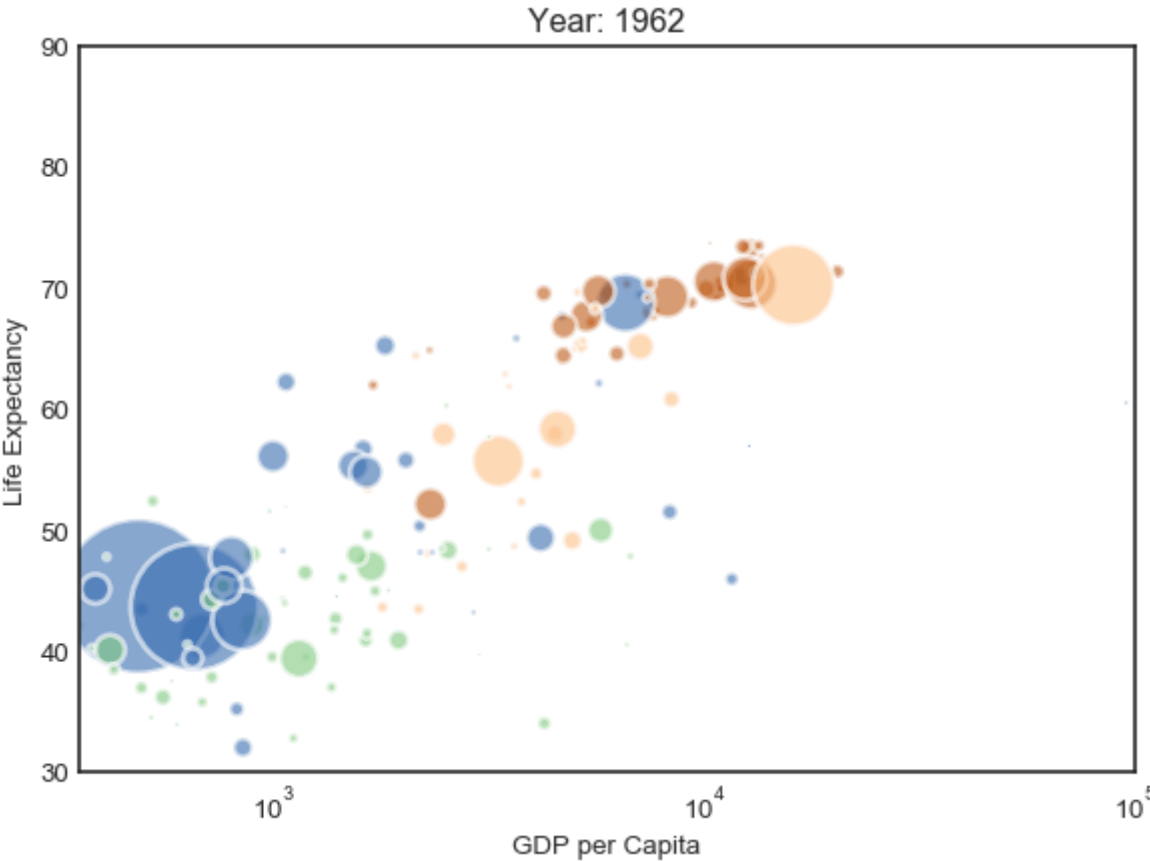
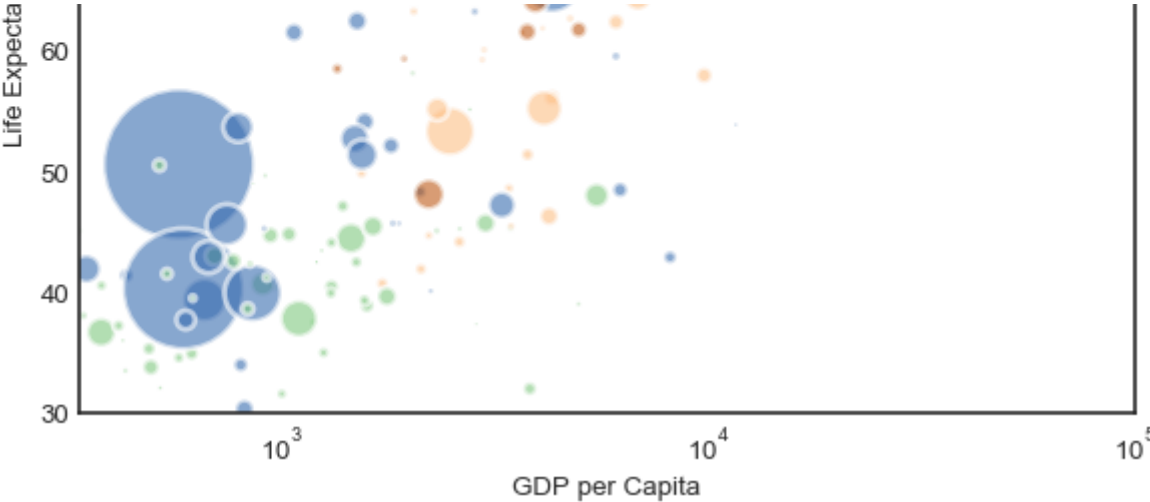


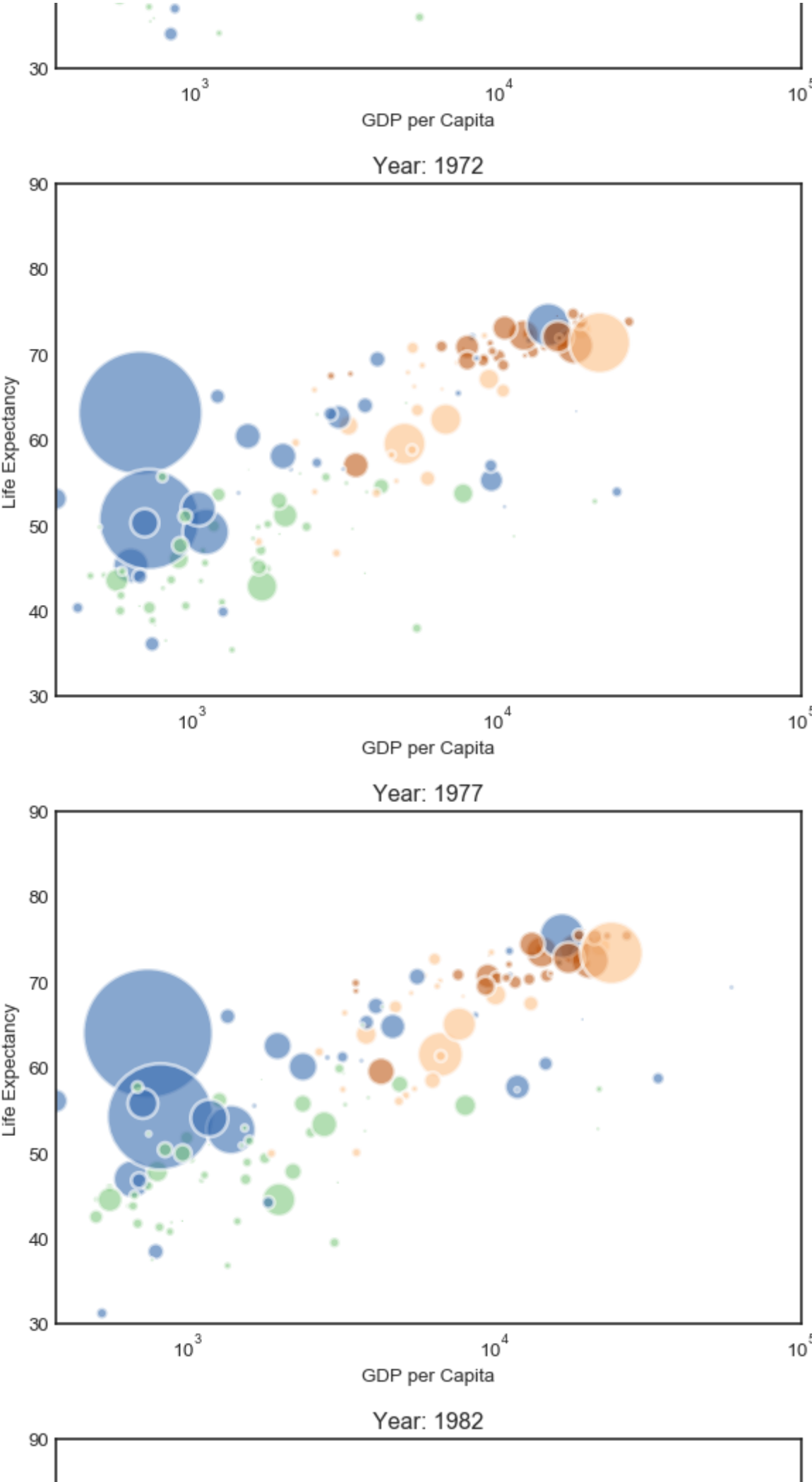
```

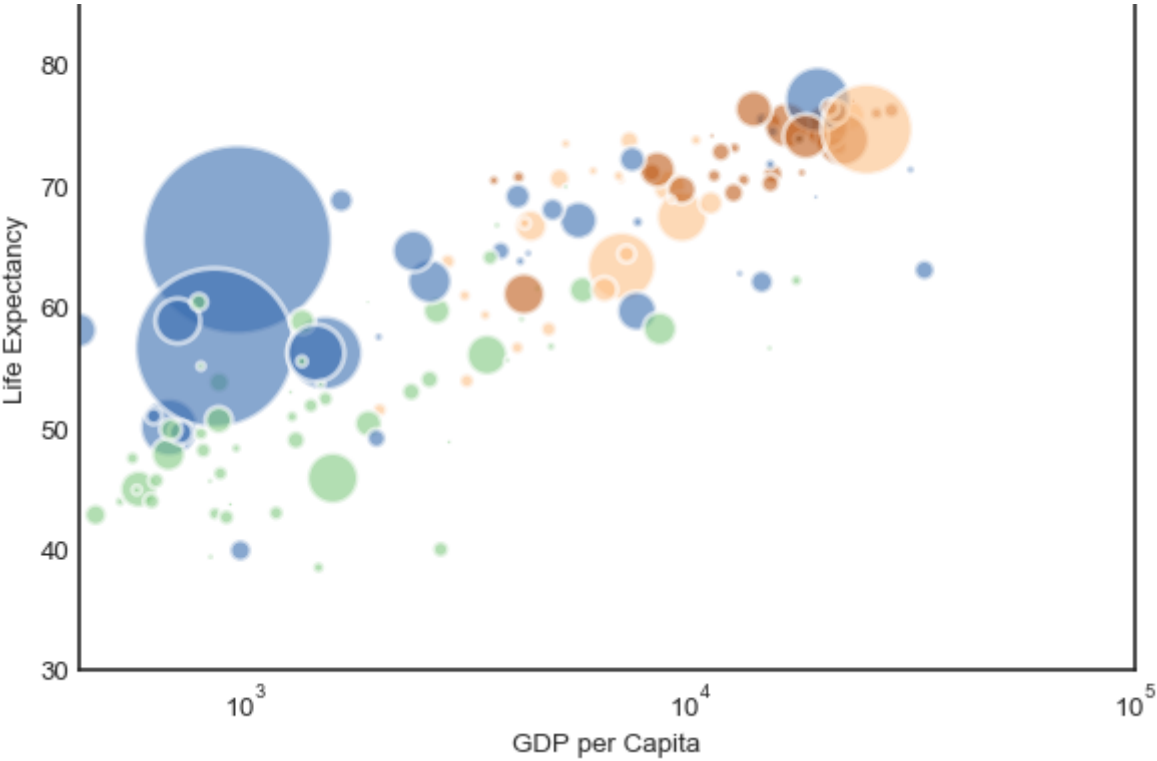
C:\Users\rogo\Anaconda3\lib\site-packages\matplotlib\axes\_base.py:3129: UserWarning
  'Attempted to set non-positive xlimits for log-scale axis; '
C:\Users\rogo\Anaconda3\lib\site-packages\matplotlib\axes\_base.py:3129: UserWarning
  'Attempted to set non-positive xlimits for log-scale axis; '
C:\Users\rogo\Anaconda3\lib\site-packages\matplotlib\axes\_base.py:3129: UserWarning
  'Attempted to set non-positive xlimits for log-scale axis; '
C:\Users\rogo\Anaconda3\lib\site-packages\matplotlib\axes\_base.py:3129: UserWarning
  'Attempted to set non-positive xlimits for log-scale axis; '
C:\Users\rogo\Anaconda3\lib\site-packages\matplotlib\axes\_base.py:3129: UserWarning
  'Attempted to set non-positive xlimits for log-scale axis; '
C:\Users\rogo\Anaconda3\lib\site-packages\matplotlib\axes\_base.py:3129: UserWarning
  'Attempted to set non-positive xlimits for log-scale axis; '
C:\Users\rogo\Anaconda3\lib\site-packages\matplotlib\axes\_base.py:3129: UserWarning
  'Attempted to set non-positive xlimits for log-scale axis; '
C:\Users\rogo\Anaconda3\lib\site-packages\matplotlib\axes\_base.py:3129: UserWarning
  'Attempted to set non-positive xlimits for log-scale axis; '
C:\Users\rogo\Anaconda3\lib\site-packages\matplotlib\axes\_base.py:3129: UserWarning
  'Attempted to set non-positive xlimits for log-scale axis; '
C:\Users\rogo\Anaconda3\lib\site-packages\matplotlib\axes\_base.py:3129: UserWarning
  'Attempted to set non-positive xlimits for log-scale axis; '
C:\Users\rogo\Anaconda3\lib\site-packages\matplotlib\axes\_base.py:3129: UserWarning
  'Attempted to set non-positive xlimits for log-scale axis; '
C:\Users\rogo\Anaconda3\lib\site-packages\matplotlib\axes\_base.py:3129: UserWarning
  'Attempted to set non-positive xlimits for log-scale axis; '
C:\Users\rogo\Anaconda3\lib\site-packages\matplotlib\axes\_base.py:3129: UserWarning
  'Attempted to set non-positive xlimits for log-scale axis; '
C:\Users\rogo\Anaconda3\lib\site-packages\matplotlib\axes\_base.py:3129: UserWarning
  'Attempted to set non-positive xlimits for log-scale axis; '
C:\Users\rogo\Anaconda3\lib\site-packages\matplotlib\axes\_base.py:3129: UserWarning
  'Attempted to set non-positive xlimits for log-scale axis; '

```

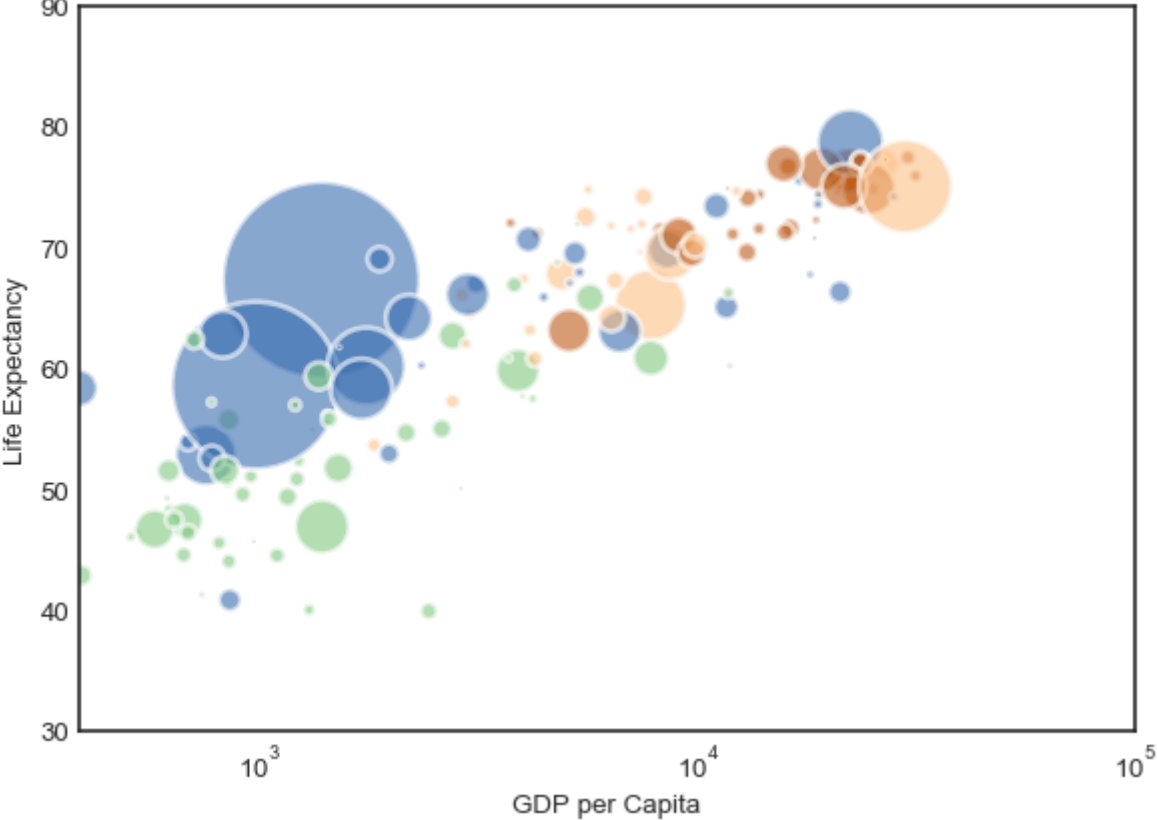




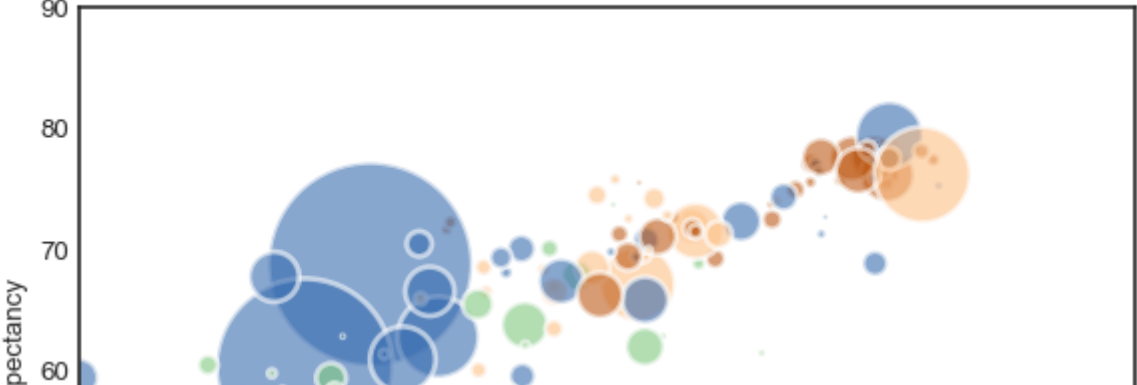


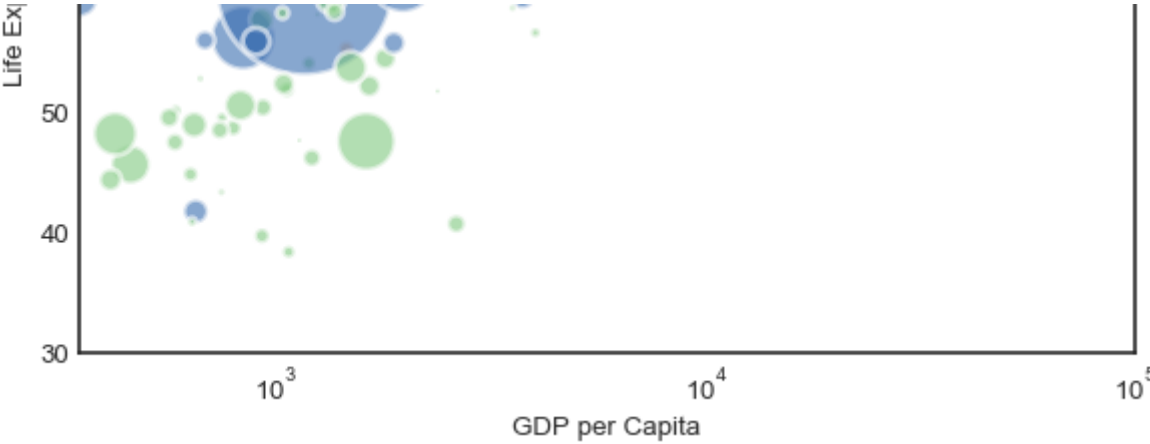


Year: 1987

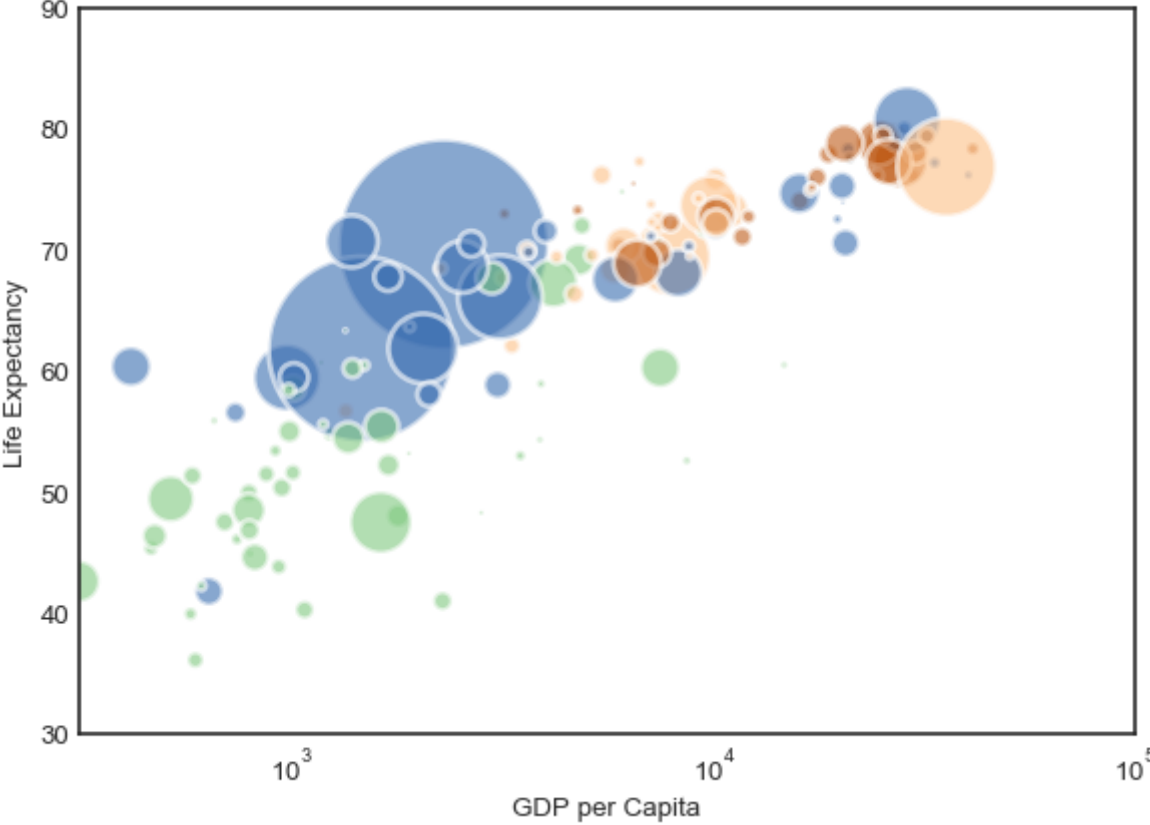


Year: 1992

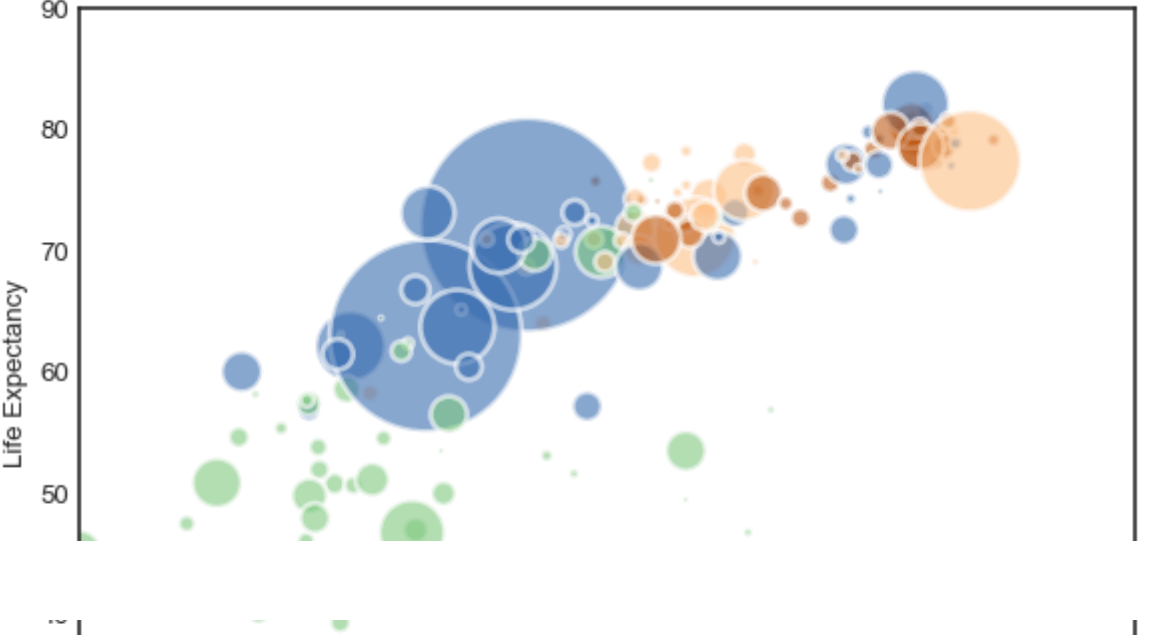




Year: 1997



Year: 2002



```
%matplotlib inline
```

```
import numpy as np # библиотека для работы с числами, пригодится для преобразований
import pandas as pd
import matplotlib.pyplot as plt # библиотека для визуализации
import seaborn as sns # библиотека для визуализации
```



File "<ipython-input-198-1699eaa8571f>", line 1

```
sns.set_style("whitegrid")
```

SyntaxError: unexpected EOF while parsing

SEARCH STACK OVERFLOW

```
df = pd.read_csv('filmdeathcounts.csv')
```

```
df.head()
```



	Film	Year	Body_Count	MPAA_Rating	Genre	Director
0	24 Hour Party People	2002	7	R	Biography Comedy Drama Music	Michael Winterbottom
1	28 Days Later	2002	53	R	Horror Sci-Fi Thriller	Danny Boyle

Давайте добавим две новых переменных: счетчик фильмов (понадобится позже при группировках, для всех фильмов равен 1). И переменная, измеряющая количество трупов в минуту экранного времени :)


```
df['Film_Count'] = 1
df['Body_Count_Min'] = df['Body_Count'] / df['Length_Minutes'].astype(float)
df.head()
```



	Film	Year	Body_Count	MPAA_Rating	Genre	Director
0	24 Hour Party People	2002	7	R	Biography Comedy Drama Music	Michael Winterbottom
1	28 Days Later	2002	53	R	Horror Sci-Fi Thriller	Danny Boyle

В прошлые разы мы с вами смотрели, как группировать таблицы по паре параметров и считать описательные статистики. В pandas мы можем создавать агрегированные таблицы (суммирующие, пивотальные - вы можете знать их под этими именами). Давайте сгруппируем фильмы по году и для каждой колонки посчитаем сумму, среднее и медиану.

```
group_year = df.groupby('Year').agg([np.mean, np.median, sum])
group_year.tail()
```




Year	Body_Count			Length_Minutes			IMDB_Rating		
	mean	median	sum	mean	median	sum	mean	median	sum
2007	85.312500	45.5	4095	114.062500	111.0	5475	6.829167	7.00	327.8
2008	68.653846	37.0	1785	109.615385	108.5	2850	6.573077	6.65	170.9
2009	55.000000	59.0	605	112.272727	110.0	1235	6.845455	6.60	75.3
2010	129.750000	126.0	519	115.750000	111.0	463	7.250000	7.25	29.0
2013	156.000000	156.0	156	119.000000	119.0	119	6.500000	6.50	6.5

```
df_bc = pd.DataFrame({'mean_value': group_year['Body_Count']['mean'],
                      'median_value': group_year['Body_Count']['median']})


df_bc_min = pd.DataFrame({'mean_value': group_year['Body_Count_Min']['mean'],
                          'median_value': group_year['Body_Count_Min']['median']})

df_bc.head()
```




Year	mean_value	median_value
1949	4.0	4.0
1954	57.5	57.5
1957	67.0	67.0
1959	7.0	7.0
1960	55.0	55.0

```
df_bc_min.head()
```



Year	mean_value	median_value
1949	0.038462	0.038462
1954	0.425822	0.425822
1957	0.761364	0.761364
1959	0.088608	0.088608
1960	0.429688	0.429688

```
plt.style.available
```

```
[ 'bmh',
  'classic',
  'dark_background',
  'fast',
  'fivethirtyeight',
  'ggplot',
  'grayscale',
  'seaborn-bright',
  'seaborn-colorblind',
  'seaborn-dark-palette',
  'seaborn-dark',
  'seaborn-darkgrid',
  'seaborn-deep',
  'seaborn-muted',
  'seaborn-notebook',
  'seaborn-paper',
  'seaborn-pastel',
  'seaborn-poster',
  'seaborn-talk',
  'seaborn-ticks',
  'seaborn-white',
  'seaborn-whitegrid',
  'seaborn',
  'Solarize_Light2',
  'tableau-colorblind10',
  '_classic_test']
```

```
# plt.style.use('seaborn-dark-palette')
```

```
sns.set_style("whitegrid")
```

```
fig, axes = plt.subplots(nrows=4, ncols=1, figsize=(16, 30))

group_year['Film_Count']['sum'].plot(kind='bar', ax=axes[0], color = '#ffa500')
axes[0].set_title('Film Count')

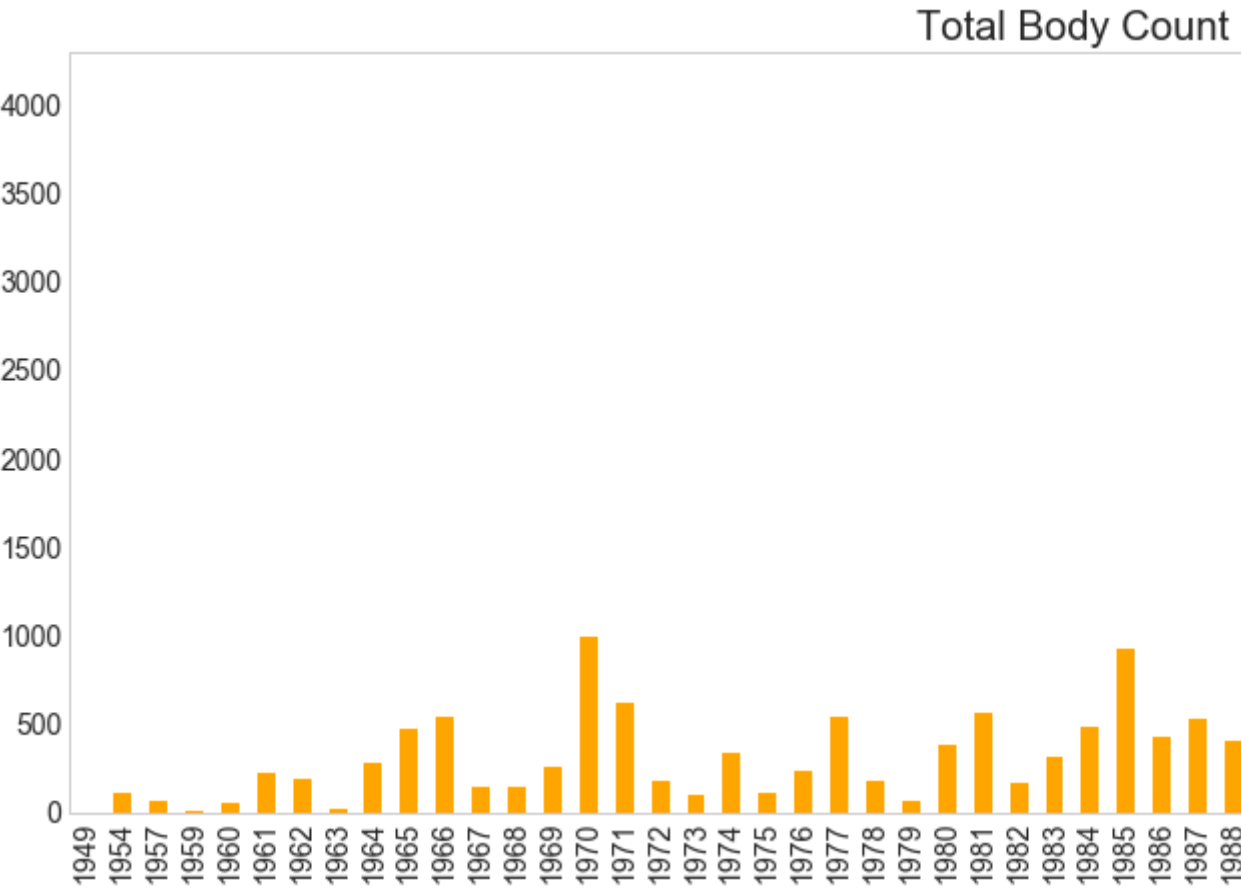
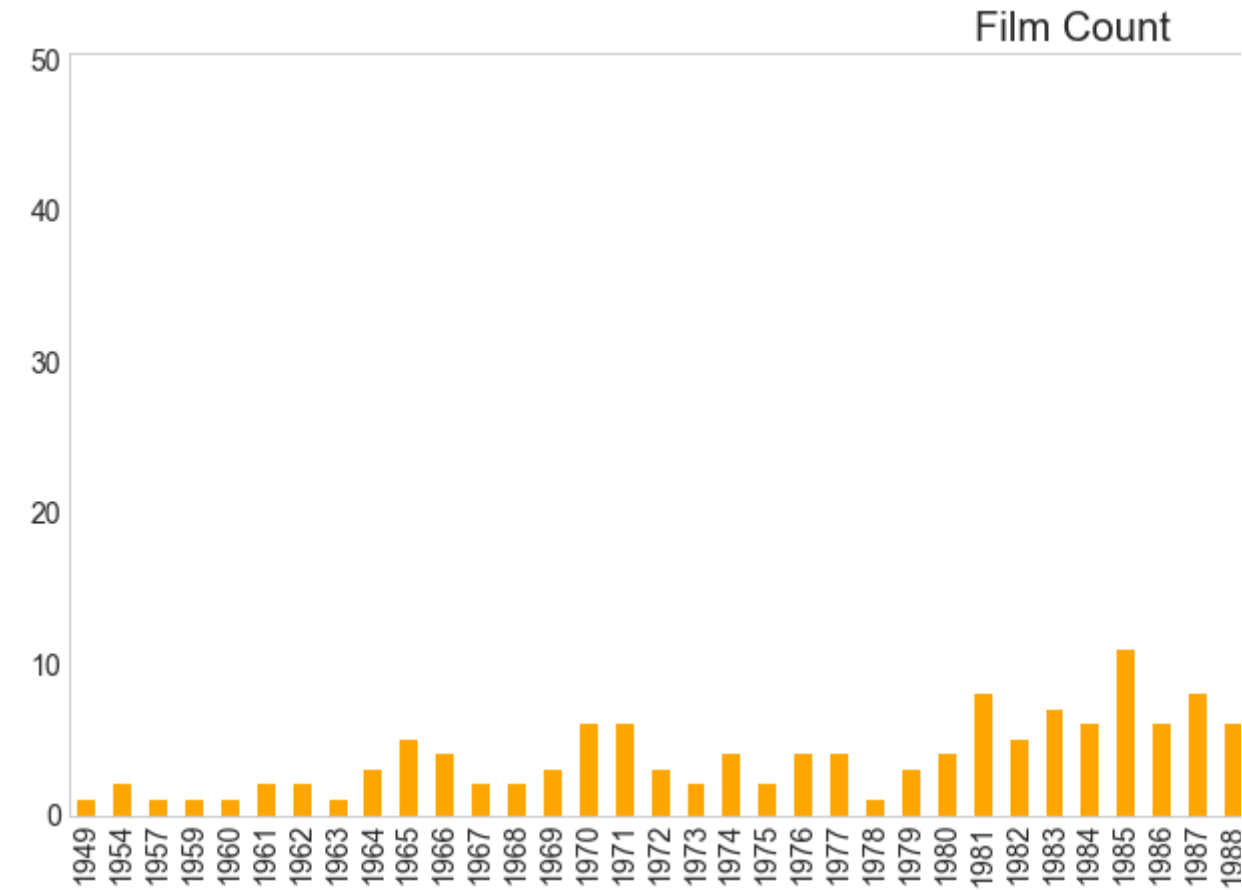
group_year['Body_Count']['sum'].plot(kind='bar', ax=axes[1], color = '#ffa500')
axes[1].set_title('Total Body Count')

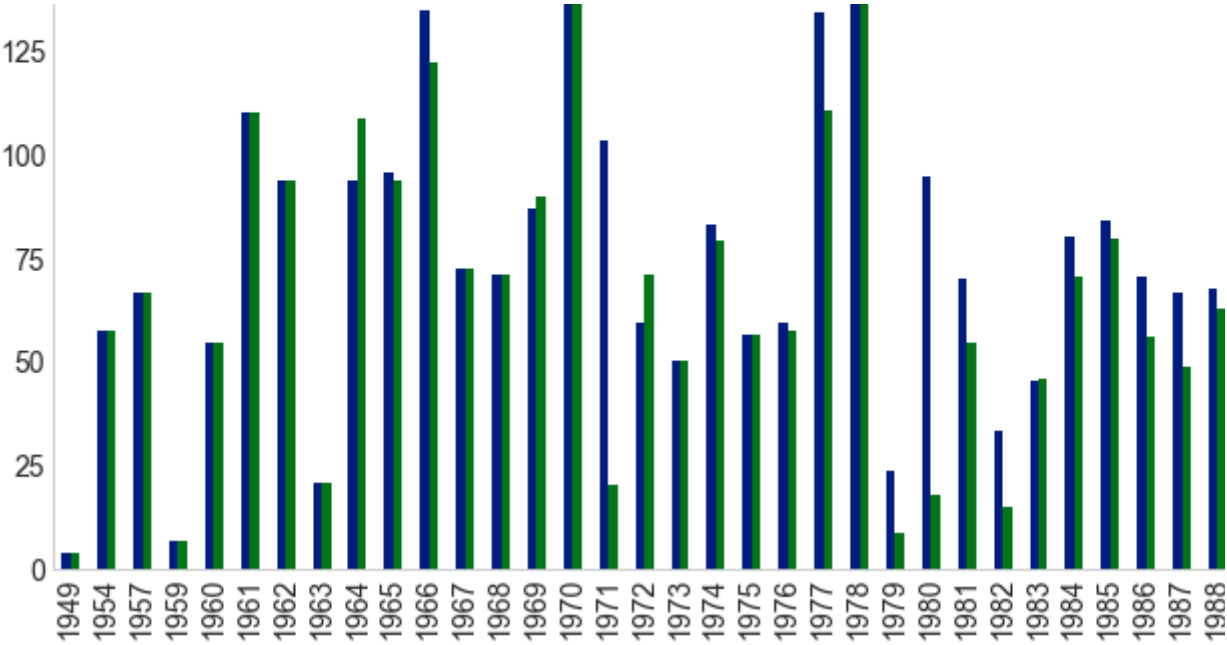
df_bc.plot(kind='bar', ax=axes[2])
axes[2].set_title('Body Count by Film')

df_bc_min.plot(kind='bar', ax=axes[3])
axes[3].set_title('Body Count by Minute')

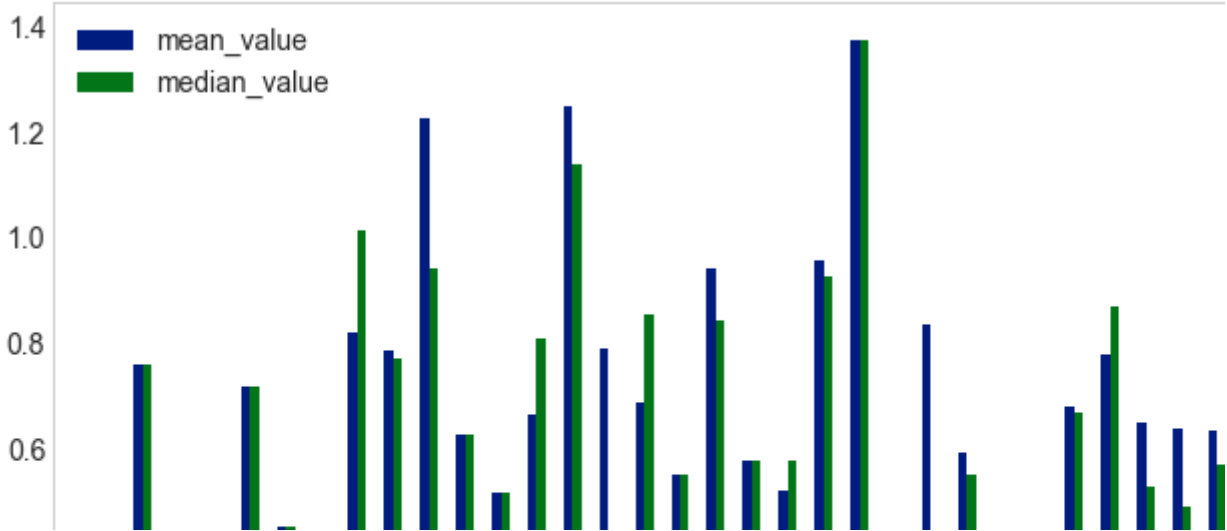
for i in range(4):
    axes[i].set_xlabel('', visible=False)
    axes[i].grid(b = False)
```







Body Count by Minute



df.head()




	Film	Year	Body_Count	MPAA_Rating	Genre	Director
0	24 Hour Party People	2002	7	R	Biography Comedy Drama Music	Michael Winterbottom
1	28 Days Later	2002	53	R	Horror Sci-Fi Thriller	Danny Boyle
28						

```
df_film = df.set_index('Film')
df_film.head()
```



	Year	Body_Count	MPAA_Rating	Genre	Director
Film					
24 Hour Partv	2002	7	R	Biography Comedy Drama Music	Michael Winterbottom

```
df_film.sort_values(by=[ 'Body_Count' ])[ 'Body_Count' ].tail(10)
```

 Film

King Arthur	378
Windtalkers	389
Lord of the Rings: Two Towers	468
A Fistful of Dynamite	471
The Last Samurai	558
Troy	572
Tae Guk Gi: The Brotherhood of War	590
300	600
Kingdom of Heaven	610
Lord of the Rings: Return of the King	836

Name: Body_Count, dtype: int64

```
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10, 8))

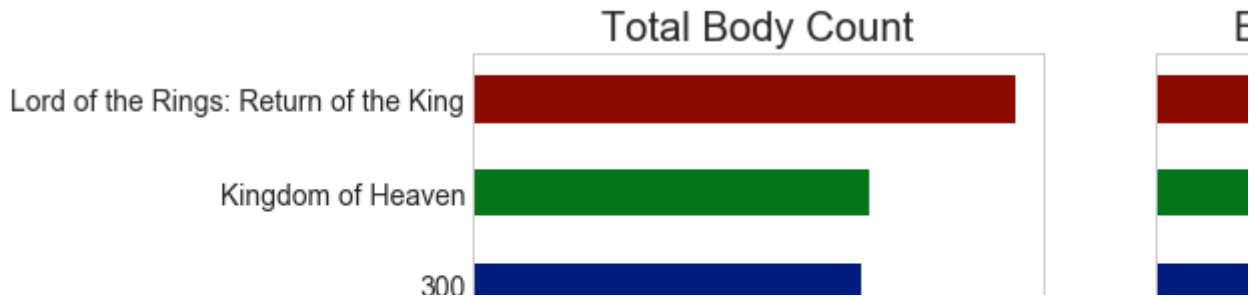
bc = df_film.sort_values(by=[ 'Body_Count' ])[ 'Body_Count' ].tail(10)
bc.plot(kind='barh', ax=axes[0])
axes[0].set_title('Total Body Count')

bc_min = df_film.sort_values(by=[ 'Body_Count_Min' ])[ 'Body_Count_Min' ].tail(10)
bc_min.plot(kind='barh', ax=axes[1])
axes[1].set_title('Body Count per Minute')
axes[1].yaxis.set_ticks_position('right')

for i in range(2):
    axes[i].set_ylabel('', visible=False)
    axes[i].grid('False')
```



```
C:\Users\rogov\Anaconda3\lib\site-packages\matplotlib\cbook\deprecation.py:107: Ma
warnings.warn(message, mplDeprecation, stacklevel=1)
C:\Users\rogov\Anaconda3\lib\site-packages\matplotlib\cbook\deprecation.py:107: Ma
warnings.warn(message, mplDeprecation, stacklevel=1)
```



➤ Most violent directors

```
df[df['Director'].apply(lambda x: -1 != x.find('|'))].head()
```



	Film	Year	Body_Count	MPAA_Rating	Genre	Director
26	Aliens vs. Predator: Requiem	2007	5	R	Action Horror Sci-Fi Thriller	Col Strause Gr Strau
38	Aqua Teen Hunger Force Colon Movie Film for	2007	67	R	Animation Action Adventure	Maiellaro Da Wil

```
def expand_col(df_src, col, sep='|'):
    di = {}
    idx = 0
    for i in df_src.iterrows():
        d = i[1]
        names = d[col].split(sep)
        for name in names:
            # operate on a copy to not overwrite previous director names
            c = d.copy()
            c[col] = name
            di[idx] = c
            idx += 1

    df_new = pd.DataFrame(di).transpose()
    # these two columns are not recognized as numeric
    df_new['Body_Count'] = df_new['Body_Count'].astype(float)
    df_new['Body_Count_Min'] = df_new['Body_Count_Min'].astype(float)

    return df_new

bc_sum = df_dir.groupby('Director').sum().sort_values(by=['Body_Count']).tail(10)
bc_sum
```



	Body_Count	Body_Count_Min
Director		
Je-kyu Kang	1770.0	12.642857
Uwe Boll	1818.0	16.766390
Steven Spielberg	1824.0	12.095150
Antoine Fuqua	2015.0	16.781665
Wolfgang Petersen	2095.0	15.862454
Zack Snyder	3155.0	27.126865
Ridley Scott	3562.0	24.768100

```
bc_mean = df_dir.groupby('Director').agg(np.mean).sort_values(by=['Body_Count_Min']).tail(7)
bc_mean
```



	Body_Count	Body_Count_Min
Director		
Randall Wallace	305.000000	2.210145
King Hu	213.000000	2.242105
Sylvester Stallone	225.000000	2.327828
Kevin Munroe	206.000000	2.367816
Zack Snyder	286.818182	2.466079
Edward Zwick	356.700000	2.476085
Katja von Garnier	256.000000	2.612245
Simon Hunter	310.000000	2.792793
Samuel Fuller	338.000000	2.991150
Je-kyu Kang	590.000000	4.214286

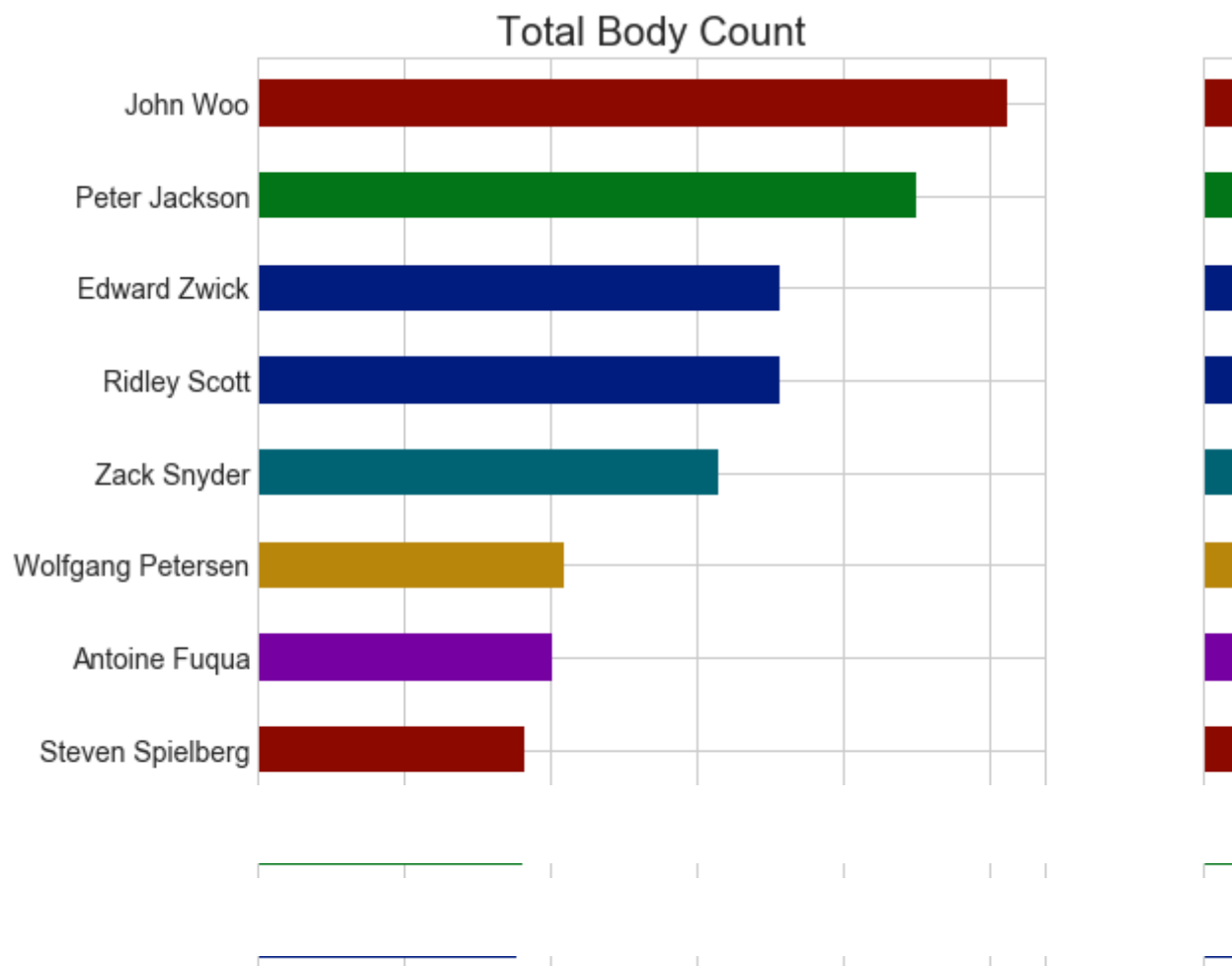
```
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(14, 8))
```

```
bc_sum['Body_Count'].plot(kind='barh', ax=axes[0])
axes[0].set_title('Total Body Count')
```

```
bc_mean['Body_Count_Min'].plot(kind='barh', ax=axes[1])
axes[1].set_title('Body Count per Minute')
axes[1].yaxis.set_ticks_position('right')
```

```
for i in range(2):
    axes[i].set_ylabel('', visible=False)
```





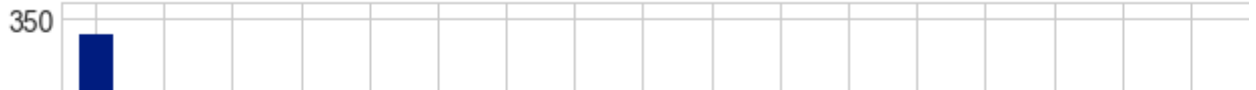
▾ by genre

```
df_dir['Genre'].value_counts().plot(kind='bar', figsize=(12, 6), title='Genres by film c')
```



<matplotlib.axes._subplots.AxesSubplot at 0x1c65e767668>

Genres by film count



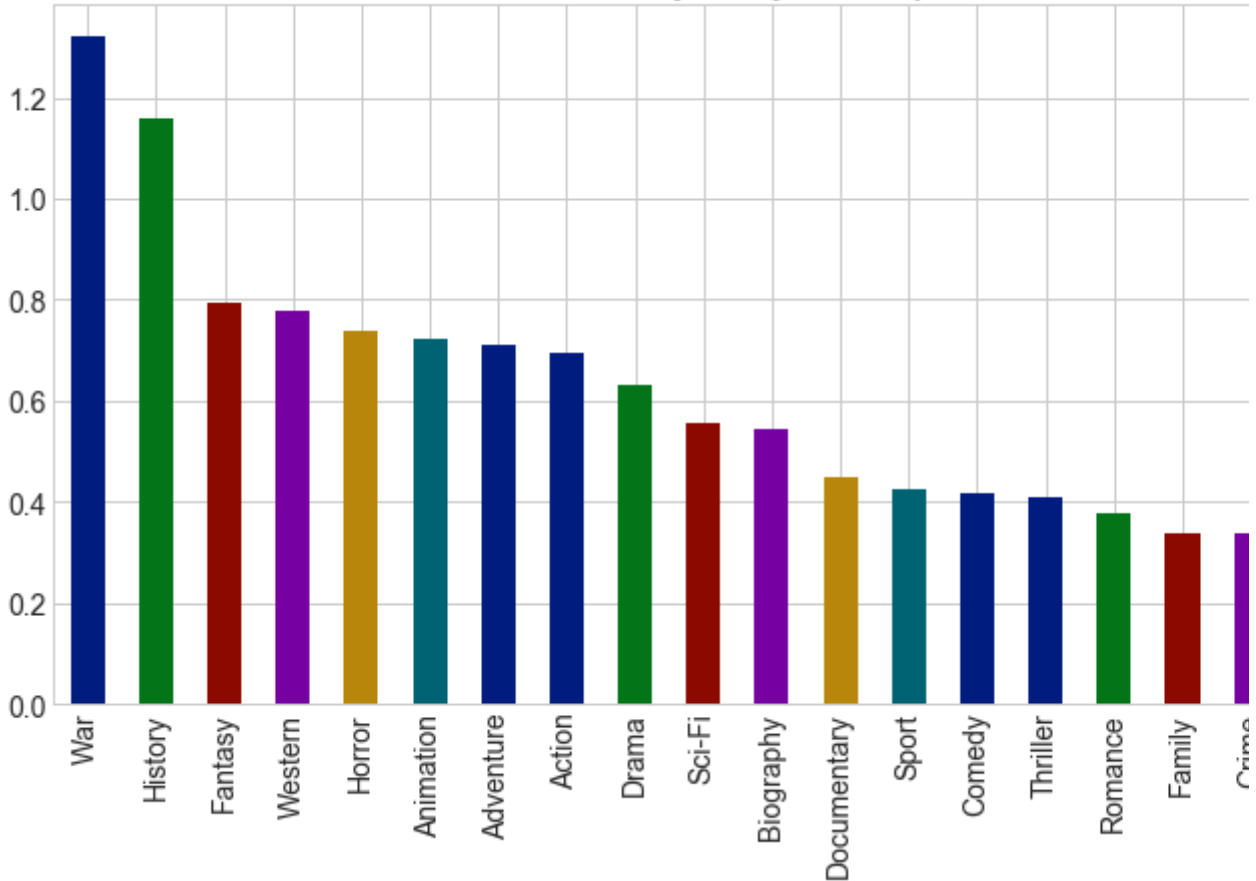
```
bc_mean = df_dir.groupby('Genre').agg(np.mean).sort_values(by=['Body_Count_Min'], ascend:
```



```
ax = bc_mean['Body_Count_Min'].plot(kind='bar', figsize=(12, 6), title='Genres by body c
ax.set_xlabel('', visible=False)
```

Text(0.5,0,'')

Genres by body count per minute



Rankings

```
df['MPAA_Rating'].value_counts()
```

R 338
 PG-13 118
 PG 35
 Unrated 28
 Approved 9
 M 5
 X 4
 GP 4
 G 3
 NR 1
 Name: MPAA_Rating, dtype: int64


```
df['Body_Count_Min'].max()
```

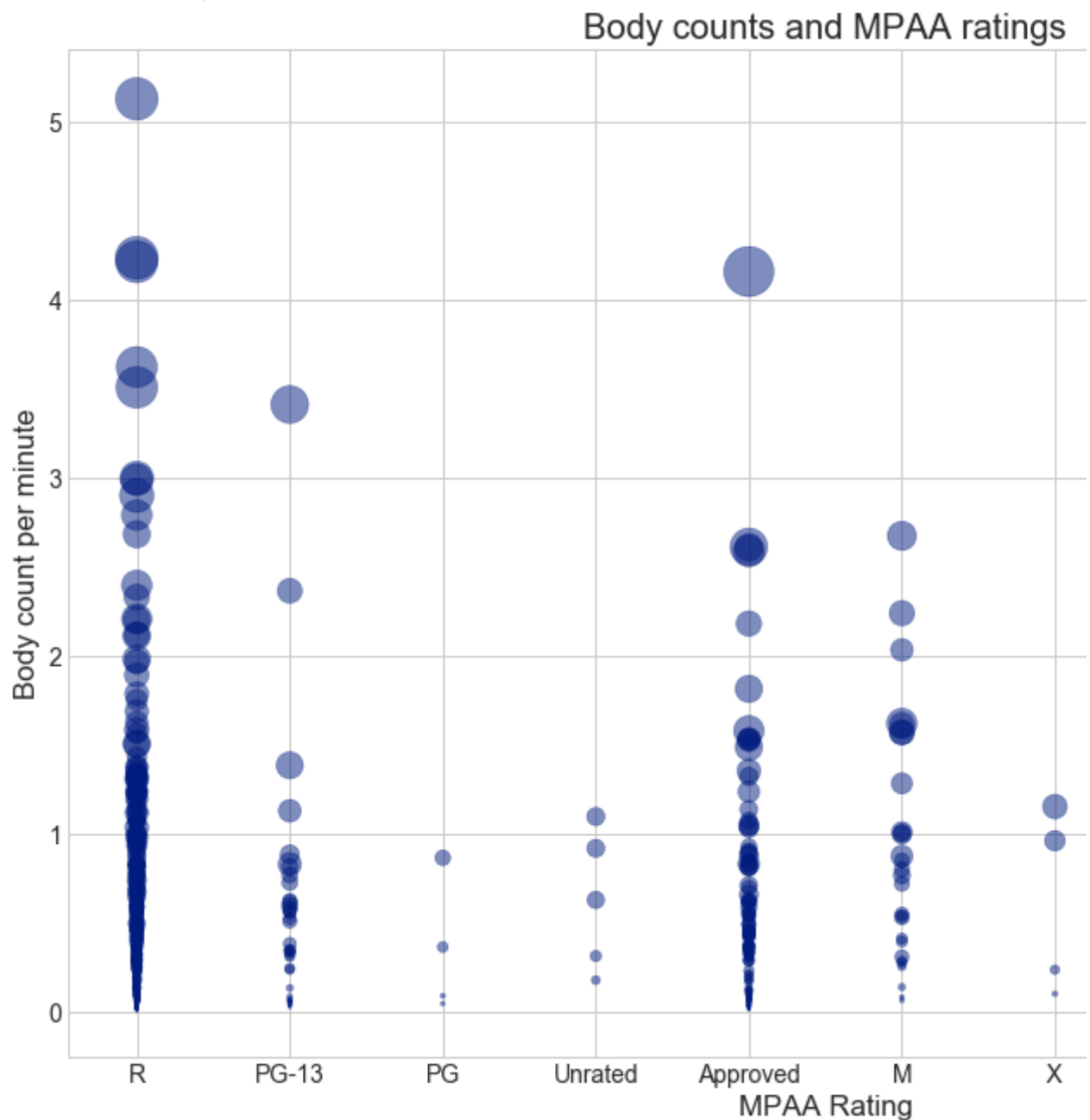


5.128205128205129

```
fig, ax = plt.subplots(figsize=(14, 10))
ax.scatter(df['MPAA_Rating'], df['Body_Count_Min'], s=df['Body_Count'], alpha=.5)
ax.set_title('Body counts and MPAA ratings')
ax.set_xlabel('MPAA Rating')
ax.set_xticks(rating_index)
ax.set_xticklabels(rating_names)
ax.set_ylabel('Body count per minute')
```



```
Text(0,0.5,'Body count per minute')
```



```
bc_top.reset_index()
```



index	Film	Year	Body_Count	MPAA_Rating	Genre
-------	------	------	------------	-------------	-------

Lord of
..

return

```
bc_top = df.sort_values('Body_Count', ascending=False)[:3].reset_index()
annotations = []
for r in range(len(bc_top)):
    annotations.append([bc_top['Film'][r], bc_top['IMDB_Rating'][r], bc_top['Body_Count_!
annotations
```

```
[[['Lord of the Rings: Return of the King', 8.9, 4.159203980099503],
  ['Kingdom of Heaven', 7.1, 4.236111111111111],
  ['300', 7.7, 5.128205128205129]]
```

```
fig, ax = plt.subplots(figsize=(14, 10))
ax.scatter(df['IMDB_Rating'], df['Body_Count_Min'], s=df['Body_Count'], alpha=.5)
ax.set_title('Body count and IMDB ratings')
ax.set_xlabel('IMDB Rating')
ax.set_ylabel('Body count per minute')
```

```
for annotation, x, y in annotations:
    plt.annotate(
        annotation,
        xy=(x, y),
        xytext=(0, 30),
        textcoords='offset points',
        ha='center',
        va='bottom',
        size=12.5,
        bbox=dict(boxstyle='round,pad=0.5', fc='yellow', alpha=0.5),
        arrowprops=dict(arrowstyle='->'))
```



```

-----
NameError                                Traceback (most recent call last)
<ipython-input-300-62f93a425715> in <module>()
    17         arrowprops=dict(arrowstyle='-'))
    18
--> 19 plt.annotate(chartinfo, xy=(0, -1.12), xycoords='axes fraction')

```

NameError: name 'chartinfo' is not defined

SEARCH STACK OVERFLOW

