Мухаметшин А.Р. ББМО-01-25

```
!git clone https://github.com/neuralcomputer/ML_School.git
```

```
Cloning into 'ML_School'...
remote: Enumerating objects: 94, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 94 (delta 5), reused 0 (delta 0), pack-reused 79 (from 1)
Receiving objects: 100% (94/94), 33.83 MiB | 29.24 MiB/s, done.
Resolving deltas: 100% (29/29), done.
```

```
# будем отображать графики внутри ноутбука
%matplotlib inline
#увеличим размер по умолчанию рисуемых графиков
from pylab import rcParams
rcParams['figure.figsize'] = 8, 5
# pylab часть matplotlib, но ее уже не рекомендуют использовать. Пока же работает.
import matplotlib.pyplot as plt
#графики в svg выглядят более четкими
%config InlineBackend.figure_format = 'svg'

import pandas as pd # подключаем pandas
# Читаем данные из файла
df = pd.read_csv('ML_School/Video_Games_Sales_as_at_22_Dec_2016.csv') # read_csv читает данные из csv файла
df.info() # посмотрим на информацию о содержимом и типе значений
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16719 entries, 0 to 16718
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Name             16717 non-null  object 
 1   Platform         16719 non-null  object 
 2   Year_of_Release  16450 non-null  float64
 3   Genre            16717 non-null  object 
 4   Publisher        16665 non-null  object 
 5   NA_Sales         16719 non-null  float64
 6   EU_Sales         16719 non-null  float64
 7   JP_Sales         16719 non-null  float64
 8   Other_Sales      16719 non-null  float64
 9   Global_Sales     16719 non-null  float64
 10  Critic_Score     8137 non-null   float64
 11  Critic_Count     8137 non-null   float64
 12  User_Score       10015 non-null  object 
 13  User_Count       7590 non-null   float64
 14  Developer        10096 non-null  object 
 15  Rating           9950 non-null   object 
dtypes: float64(9), object(7)
memory usage: 2.0+ MB
```

```
df = df.dropna() # отбросим строки с пропущенными данными
print(df.shape) # посмотрим, сколько осталось
```

```
(6825, 16)
```

```
df['User_Score'] = df.User_Score.astype('float64') # преобразуем в число с плавающей точкой
df['Year_of_Release'] = df.Year_of_Release.astype('int64') # преобразуем в целое число
df['User_Count'] = df.User_Count.astype('int64') # преобразуем в целое число
df['Critic_Count'] = df.Critic_Count.astype('int64') # преобразуем в целое число
# Этого можно не делать, так как данные столбца Critic_Score уже float64
# df['Critic_Score'] = df.Critic_Score.astype('float64') # преобразуем в число с плавающей точкой.
# столбцы *Sales также не трогаем, они уже в float64
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 6825 entries, 0 to 16706
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Name             6825 non-null   object 
 1   Platform         6825 non-null   object 
 2   Year_of_Release  6825 non-null   int64  
 3   Genre            6825 non-null   object 
 4   Publisher        6825 non-null   object 
 5   NA_Sales         6825 non-null   float64
 6   EU_Sales         6825 non-null   float64
 7   JP_Sales         6825 non-null   float64
 8   Other_Sales      6825 non-null   float64
 9   Global_Sales     6825 non-null   float64
```

```
 10  Critic_Score     6825 non-null   float64
 11  Critic_Count     6825 non-null   int64
 12  User_Score       6825 non-null   float64
 13  User_Count       6825 non-null   int64
 14  Developer        6825 non-null   object
 15  Rating           6825 non-null   object
dtypes: float64(7), int64(3), object(6)
memory usage: 906.4+ KB
```

```
# названия столбцов
useful_cols = ['Name', 'Platform', 'Year_of_Release', 'Genre',
'Global_Sales', 'Critic_Score', 'Critic_Count',
'User_Score', 'User_Count', 'Rating'
]
df[useful_cols].head() # отображаем их
```

	Name	Platform	Year_of_Release	Genre	Global_Sales	Critic_Score	Critic_Count	User_Score	User_Count	Rating
0	Wii Sports	Wii	2006	Sports	82.53	76.0	51	8.0	322	E
2	Mario Kart Wii	Wii	2008	Racing	35.52	82.0	73	8.3	709	E
3	Wii Sports Resort	Wii	2009	Sports	32.77	80.0	73	8.0	192	E

```
sales_df = df[[x for x in df.columns if 'Sales' in x] + ['Year_of_Release']]
sales_df
```

	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales	Year_of_Release
0	41.36	28.96	3.77	8.45	82.53	2006
2	15.68	12.76	3.79	3.29	35.52	2008
3	15.61	10.93	3.28	2.95	32.77	2009
6	11.28	9.14	6.50	2.88	29.80	2006
7	13.96	9.18	2.93	2.84	28.92	2006
...	...	...	...	...	...	...
16667	0.01	0.00	0.00	0.00	0.01	2001
16677	0.01	0.00	0.00	0.00	0.01	2002
16696	0.00	0.01	0.00	0.00	0.01	2014
16700	0.01	0.00	0.00	0.00	0.01	2011
16706	0.00	0.01	0.00	0.00	0.01	2011

6825 rows × 6 columns

Далее: New interactive sheet

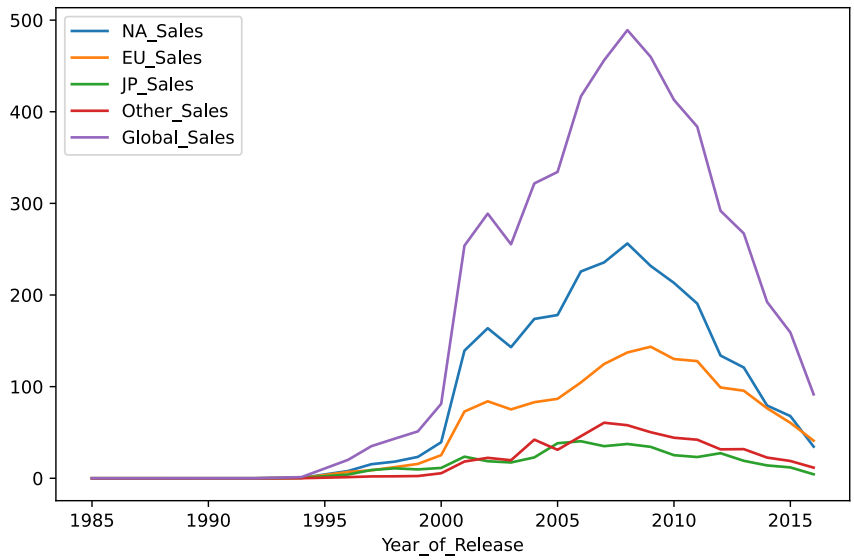
```
summarized_sales_df=sales_df.groupby('Year_of_Release').sum() # группируем по годам
summarized_sales_df # отображаем
```

Year_of_Release	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
1985	0.00	0.03	0.00	0.01	0.03
1988	0.00	0.02	0.00	0.01	0.03
1992	0.02	0.00	0.00	0.00	0.03
1994	0.39	0.26	0.53	0.08	1.27
1996	7.91	6.88	4.06	1.24	20.10
1997	15.34	8.67	9.01	2.02	35.01
1998	18.13	12.13	10.81	2.14	43.18
1999	23.32	15.69	9.67	2.45	51.17
2000	39.34	25.20	11.27	5.49	81.24
2001	139.32	72.85	23.57	18.26	253.88
2002	163.76	84.03	18.61	22.30	288.84
2003	143.08	75.16	17.24	19.68	255.35
2004	173.88	83.01	22.74	42.14	321.78
2005	178.15	86.70	38.23	31.05	334.32
2006	225.69	104.53	40.43	45.90	416.72
2007	235.61	124.71	35.04	60.62	456.23
2008	256.25	137.31	37.42	57.89	489.12
2009	231.72	143.56	34.28	50.25	459.85
2010	213.24	130.13	25.19	44.24	412.96
2011	190.62	127.86	23.16	42.10	383.69
2012	133.94	99.08	27.36	31.57	291.93
2013	120.89	95.54	19.05	31.80	267.17
2014	79.38	76.42	14.02	22.58	192.43
2015	67.85	60.51	11.85	18.86	159.16
2016	34.52	41.03	4.34	11.59	91.56

Далее: New interactive sheet

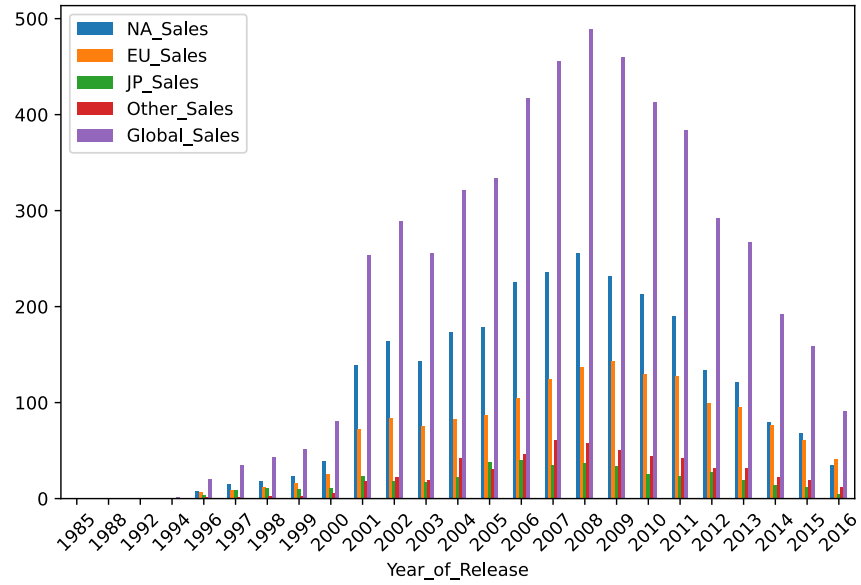
```
summarized_sales_df.plot() # нарисуем график продаж по годам, линейчатый график
```

```
<Axes: xlabel='Year_of_Release'>
```



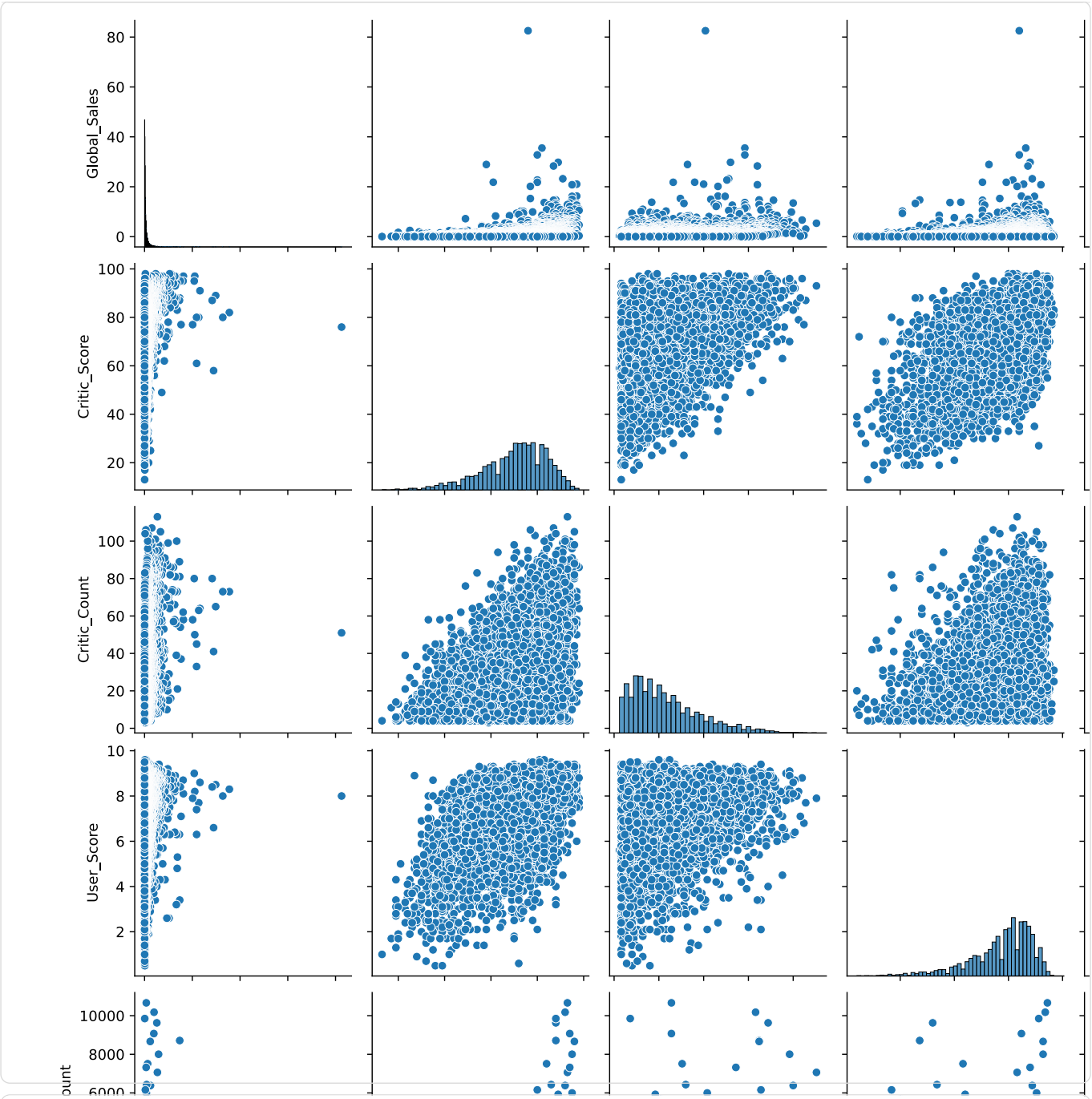

```
summarized_sales_df.plot(kind='bar', rot=45) # то же, но полосчатый график, аргумент rot - угол поворота текста в подписи оси
```

```
<Axes: xlabel='Year_of_Release'>
```



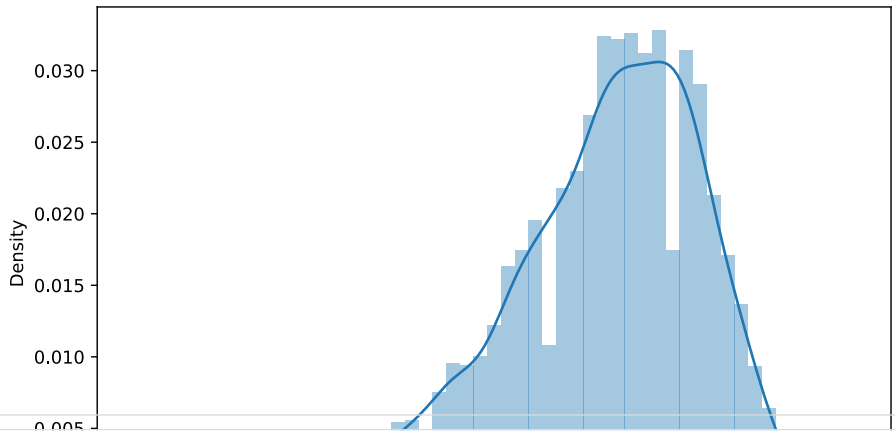

```
import seaborn as sns # подключим библиотеку
```

```
#%%capture
# Закоментируйте %%capture если график не "подвисает"
# выбранные столбцы
cols = ['Global_Sales', 'Critic_Score', 'Critic_Count', 'User_Score', 'User_Count']
sns_plot = sns.pairplot(df[cols]);# рисуем график
sns_plot.savefig('pairplot.png')#сохраняем его в файл
```



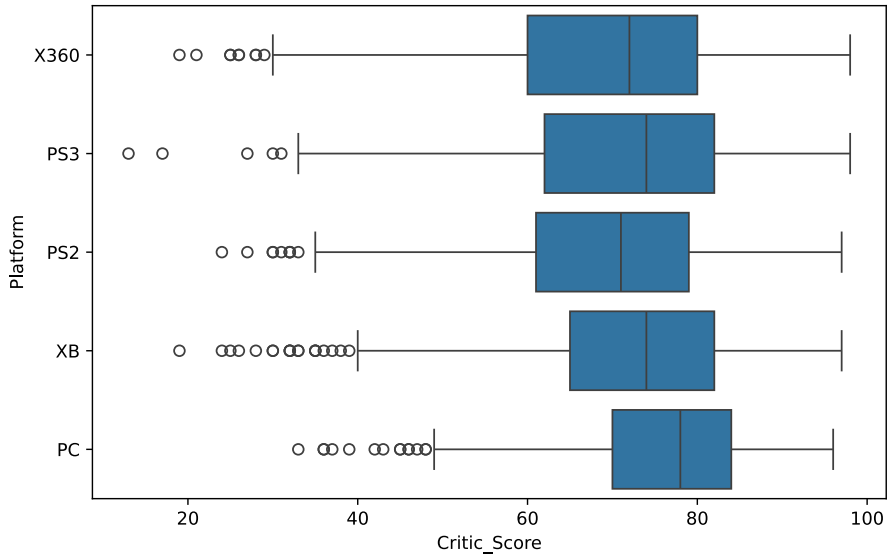

```
sns.distplot(df.Critic_Score)
```

```
/tmp/ipython-input-924927648.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df.Critic_Score)
<Axes: xlabel='Critic_Score', ylabel='Density'>
```



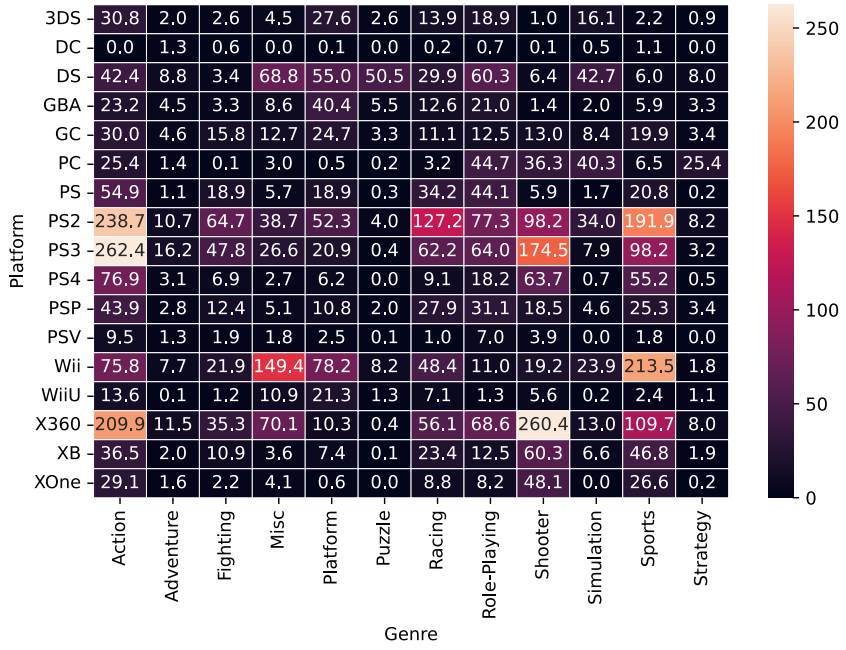

```
top_platforms = df.Platform.value_counts().sort_values(ascending = False).head(5).index.values
sns.boxplot(y="Platform", x="Critic_Score", data=df[df.Platform.isin(top_platforms)], orient="h")
```

```
<Axes: xlabel='Critic_Score', ylabel='Platform'>
```




```
platform_genre_sales = df.pivot_table(
  index='Platform',
  columns='Genre',
  values='Global_Sales',
  aggfunc=sum).fillna(0).applymap(float)
sns.heatmap(platform_genre_sales, annot=True, fmt=".1f", linewidths=.5)
```

```
/tmp/ipython-input-3281750136.py:1: FutureWarning: The provided callable <built-in function sum> is currently using DataFrameGr
  platform_genre_sales = df.pivot_table(
/tmp/ipython-input-3281750136.py:5: FutureWarning: DataFrame.applymap has been deprecated. Use DataFrame.map instead.
  aggfunc=sum).fillna(0).applymap(float)
<Axes: xlabel='Genre', ylabel='Platform'>
```

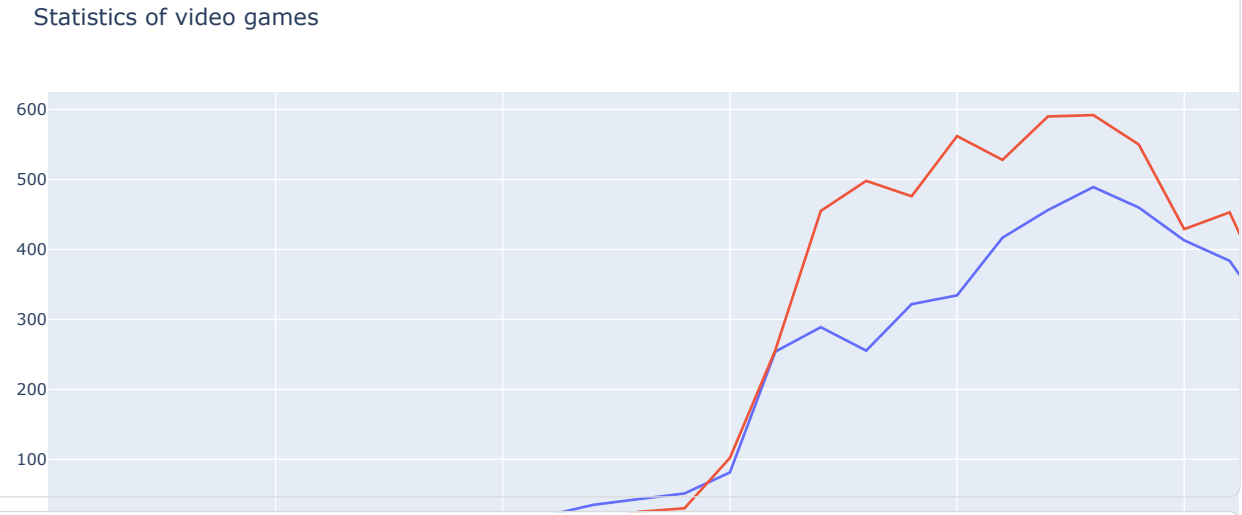



```
from plotly.offline import download_plotlyjs, plot, iplot
import plotly.graph_objs as go # графические объекты
```

```
# посчитаем число вышедших игр и проданных копий по годам
years_df = df.groupby('Year_of_Release')[['Global_Sales']].sum().join(
df.groupby('Year_of_Release')[['Name']].count()
)
years_df.columns = ['Global_Sales', 'Number_of_Games']
years_df
```

Year_of_Release	Global_Sales	Number_of_Games
1985	0.03	1
1988	0.03	1
1992	0.03	1
1994	1.27	1
1996	20.10	7
1997	35.01	13
1998	43.18	25
1999	51.17	30
2000	81.24	102
2001	253.88	256
2002	288.84	455
2003	255.35	498
2004	321.78	476
2005	334.32	562
2006	416.72	528
2007	456.23	590
2008	489.12	592
2009	459.85	550
2010	412.96	429
2011	383.69	453
2012	291.93	313
2013	267.17	266
2014	192.43	253
2015	159.16	211

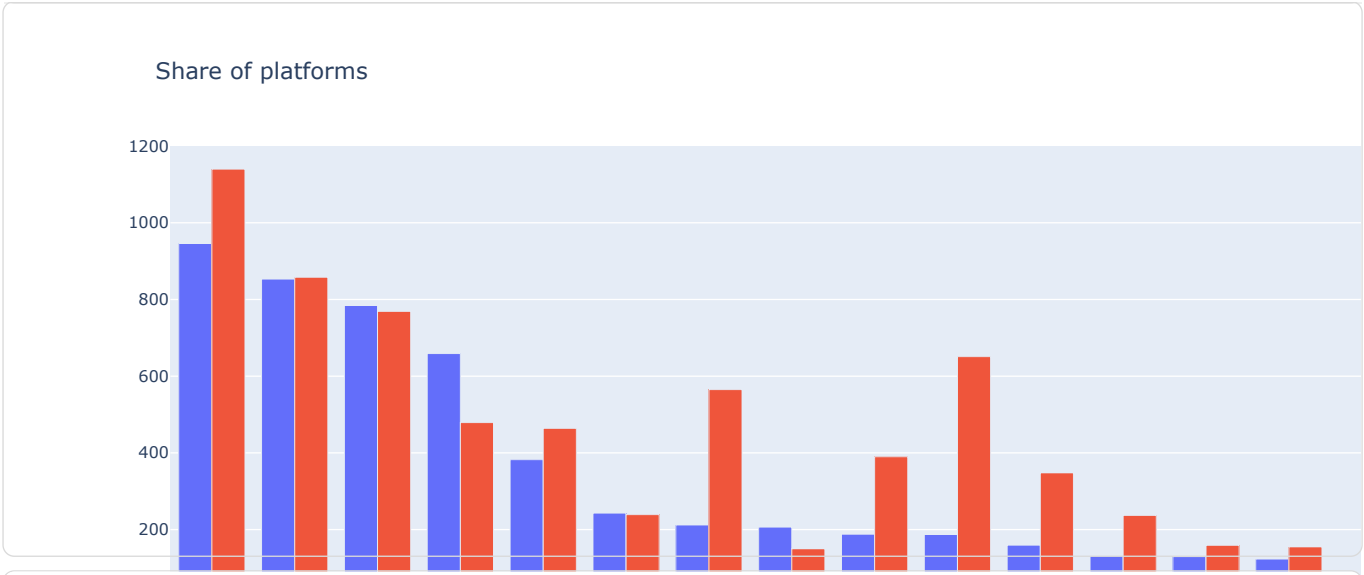
```
# создаем линию для числа проданных копий
trace0 = go.Scatter(
x=years_df.index,
y=years_df.Global_Sales,
name='Global Sales'
)
# создаем линию для числа вышедших игр
trace1 = go.Scatter(
x=years_df.index,
y=years_df.Number_of_Games,
name='Number of games released'
)
# объединяем графические объекты и задаем title графика в layout
data = [trace0, trace1]
layout = {'title': 'Statistics of video games'}
# создаем объект Figure и визуализируем его
fig = go.Figure(data=data, layout=layout)
#fig.show()
iplot(fig, show_link=False) # не показывать ссылку на экспортирование полотна
```




```
plot(fig, filename='years_stats.html', show_link=False)
```

```
'years_stats.html'
```

```
# считаем число проданных и вышедших игр по платформам
platforms_df = df.groupby('Platform')[['Global_Sales']].sum().join(
df.groupby('Platform')[['Name']].count()
)
platforms_df.columns = ['Global_Sales', 'Number_of_Games']
platforms_df.sort_values('Global_Sales', ascending=False, inplace=True)
# создаем объект для визуализации
trace0 = go.Bar(
x=platforms_df.index,
y=platforms_df.Global_Sales,
name='Global Sales'
)
# создаем объект для визуализации
trace1 = go.Bar(
x=platforms_df.index,
y=platforms_df.Number_of_Games,
name='Number of games released'
)
# объединяем графические объекты и задаем title графика в layout
data = [trace0, trace1]
layout = {'title': 'Share of platforms', 'xaxis': {'title': 'platform'}}
# создаем объект Figure и визуализируем его
fig = go.Figure(data=data, layout=layout)
iplot(fig, show_link=False)
```




```
# создаем ящик с усами для каждого жанра из наших данных
data = [] # пустой массив который будем постепенно заполнять.
for genre in df.Genre.unique(): #
  data.append(
    go.Box(y=df[df.Genre==genre].Critic_Score, name=genre)
  )
# пропускаем создание полотна
# визуализируем данные
iplot(data, show_link = False)
```