5. Project YP

February 3, 2022

1 Project 5

Project description

You work for an online store that sells computer games worldwide. Historical game sales data, user and expert ratings, genres and platforms (such as Xbox or PlayStation) are available from open sources.

Task

It is necessary to identify patterns that determine the success of the game. This will allow you to bid on a potentially popular product and plan advertising campaigns.

Data Description

```
Name - name of the game
Platform - platform
Year_of_Release - year of release
Genre - game genre
NA_sales - sales in North America (millions sold)
EU_sales - sales in Europe (millions sold)
JP_sales - sales in Japan (millions sold)
Other_sales - sales in other countries (millions sold)
Critic_Score - Critics score (max 100)
User_Score - user rating (max 10)
Rating - rating from the ESRB (Entertainment Software Rating Board). This association determine
```

1.1 First look at the data

Import the necessary libraries and display basic information

[3]:
```
from scipy import stats as st
import numpy as np
import pandas as pd
import math
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv('./games.csv')

print(df.info())
```

```
display(df.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16715 entries, 0 to 16714
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Name             16713 non-null  object
 1   Platform         16715 non-null  object
 2   Year_of_Release  16446 non-null  float64
 3   Genre            16713 non-null  object
 4   NA_sales         16715 non-null  float64
 5   EU_sales         16715 non-null  float64
 6   JP_sales         16715 non-null  float64
 7   Other_sales      16715 non-null  float64
 8   Critic_Score     8137 non-null   float64
 9   User_Score       10014 non-null  object
 10  Rating           9949 non-null   object
dtypes: float64(6), object(5)
memory usage: 1.4+ MB
None
```

	Name	Platform	Year_of_Release	Genre	NA_sales	EU_sales	JP_sales	Other_sales	Critic_Score	User_Score	Rating
0	Wii Sports	Wii	2006.0	Sports	41.36	28.96	3.77	8.45	76.0	8	E
1	Super Mario Bros.	NES	1985.0	Platform	29.08	3.58	6.81	0.77	NaN	NaN	NaN
2	Mario Kart Wii	Wii	2008.0	Racing	15.68	12.76	3.79	3.29	82.0	8.3	E
3	Wii Sports Resort	Wii	2009.0	Sports	15.61	10.93	3.28	2.95	80.0	8	E
4	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	11.27	8.89	10.22	1.00	NaN	NaN	NaN

Display information about passes

[4]:
```
df.isna().sum()
```

[4]:
```
Name                 2
Platform             0
Year_of_Release    269
Genre                2
NA_sales             0
EU_sales             0
JP_sales             0
Other_sales          0
```

```
Critic_Score       8578
User_Score         6701
Rating             6766
dtype: int64
```

df

[5]:
```
df.describe()
```

[5]:
```
       Year_of_Release      NA_sales      EU_sales      JP_sales  \
count     16446.000000  16715.000000  16715.000000  16715.000000
mean       2006.484616      0.263377      0.145060      0.077617
std           5.877050      0.813604      0.503339      0.308853
min        1980.000000      0.000000      0.000000      0.000000
25%        2003.000000      0.000000      0.000000      0.000000
50%        2007.000000      0.080000      0.020000      0.000000
75%        2010.000000      0.240000      0.110000      0.040000
max        2016.000000     41.360000     28.960000     10.220000

        Other_sales  Critic_Score
count  16715.000000   8137.000000
mean       0.047342     68.967679
std        0.186731     13.938165
min        0.000000     13.000000
25%        0.000000     60.000000
50%        0.010000     71.000000
75%        0.030000     79.000000
max       10.570000     98.000000
```

1.1.1 Conclusion

- There are gaps in some columns, then we will remove them
- There are also lines with zero sales in the regions
- There are also “tbd” values in the Rating column. Let’s work it out next

1.2 Data preparation

Let’s bring the Name, Platform, Genre, Rating columns to lowercase

[6]:
```
df_lower = ['Name', 'Platform', 'Genre', 'Rating']

for i in df_lower:
    df[i] = df[i].str.lower()
```

Display rows with empty values in the Name column

[7]:
```
df.query('Name.isna() == True')
```

[7]:
```
       Name Platform  Year_of_Release Genre  NA_sales  EU_sales  JP_sales  \
659     NaN      gen           1993.0   NaN      1.78      0.53      0.00
14244   NaN      gen           1993.0   NaN      0.00      0.00      0.03

       Other_sales  Critic_Score User_Score Rating
659           0.08           NaN        NaN    NaN
14244         0.00           NaN        NaN    NaN
```

Delete this rows

[8]:
```
df = df.query('Name.isna() == False')
```

From the description of the table, it can be seen that the median of Year_of_Release is 2007. Let's replace the empty values with this number, because there are not very many of them

[9]:
```
df['Year_of_Release'] = df['Year_of_Release'].fillna(2007)
```

Let's replace the gaps in the Critic_Score column with -1 and in the future we will know that -1 is empty

[10]:
```
df['Critic_Score'] = df['Critic_Score'].fillna(-1)
```

The User_Score column contains the values "tbd" - To Be Determined, that is, "Will be determined". Change all values to -1.

[11]:
```
df.loc[df['User_Score'] == 'tbd', 'User_Score'] = -1
```

Replace all empty values in the User_Score column with -1

[12]:
```
df['User_Score'] = df['User_Score'].fillna(-1)
```

[13]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16713 entries, 0 to 16714
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Name             16713 non-null  object
 1   Platform         16713 non-null  object
 2   Year_of_Release  16713 non-null  float64
 3   Genre            16713 non-null  object
 4   NA_sales         16713 non-null  float64
 5   EU_sales         16713 non-null  float64
 6   JP_sales         16713 non-null  float64
 7   Other_sales      16713 non-null  float64
 8   Critic_Score     16713 non-null  float64
 9   User_Score       16713 non-null  object
 10  Rating           9949 non-null   object
```

```
dtypes: float64(6), object(5)
memory usage: 1.5+ MB
```

Replace all empty values in the Rating column with “No information”

[14]:
```
df['Rating'] = df['Rating'].fillna('No info')
```

Let’s convert the column names to lowercase

[15]:
```
df.columns = df.columns.str.lower()
```

Show duplicates

[16]:
```
df.duplicated().sum()
```

[16]:
```
0
```

There aren’t any duplicates

Let’s change the data type

[17]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16713 entries, 0 to 16714
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   name             16713 non-null  object
 1   platform         16713 non-null  object
 2   year_of_release  16713 non-null  float64
 3   genre            16713 non-null  object
 4   na_sales         16713 non-null  float64
 5   eu_sales         16713 non-null  float64
 6   jp_sales         16713 non-null  float64
 7   other_sales      16713 non-null  float64
 8   critic_score     16713 non-null  float64
 9   user_score       16713 non-null  object
 10  rating           16713 non-null  object
dtypes: float64(6), object(5)
memory usage: 1.5+ MB
```

[24]:
```
df['year_of_release'] = df['year_of_release'].astype(int)
df['user_score'] = df['user_score'].astype(float)
```

Let’s calculate the total sales in all regions

[26]:
```
df['total_sales'] = df[['na_sales', 'eu_sales', 'jp_sales', 'other_sales']].
↪sum(axis=1)
```

1.2.1 Conclusion

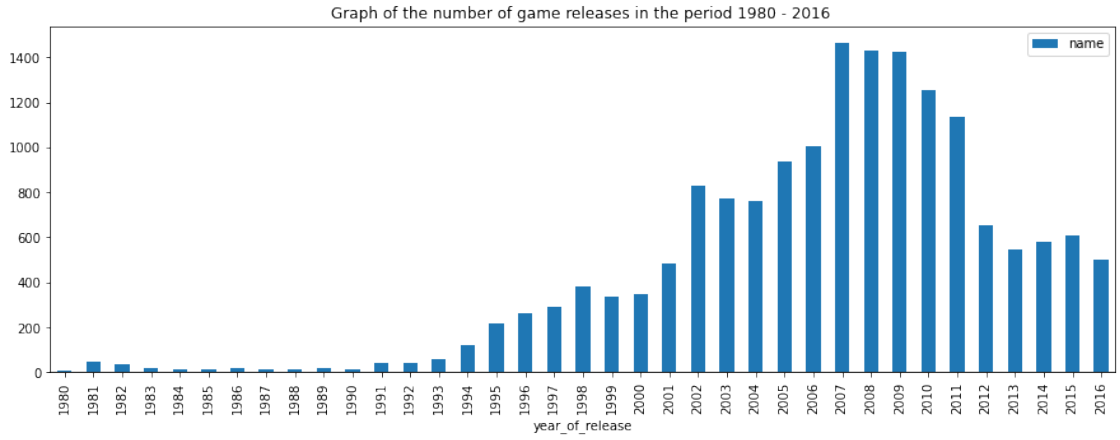
- Changed data type
- Reduced everything to lowercase
- Replaced all empty values

1.3 Exploratory data analysis

1.3.1 Let's see how many games were released in different years

[28]:
```
df_years = df.pivot_table(index='year_of_release', values='name',␣
↪aggfunc='count').reset_index()
df_years.plot(kind='bar', x='year_of_release', figsize=(15, 5))
plt.title("Graph of the number of game releases in the period 1980 - 2016 ")
```

[28]:
```
Text(0.5, 1.0, 'Graph of the number of game releases in the period 1980 - 2016
')
```



Before 1994 there were very few games

Find the most popular platforms

[29]:
```
df_years_platform = df.pivot_table(index=['year_of_release', 'platform'],␣
↪values='total_sales', aggfunc='sum').reset_index()
df_years_platform.groupby('platform')['total_sales'].sum().reset_index().
↪sort_values(by='total_sales', ascending=False).head(4)
```

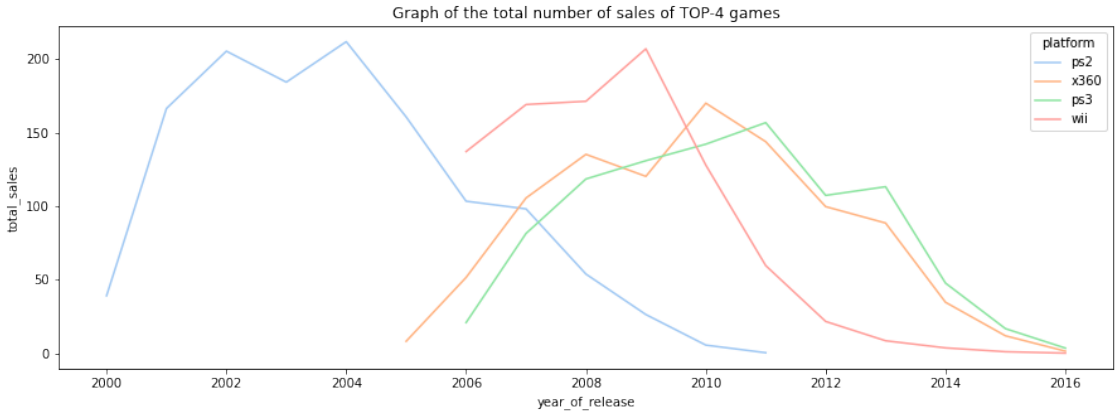
[29]:

	platform	total_sales
16	ps2	1255.77
28	x360	971.42
17	ps3	939.65
25	wii	907.51

[30]:
```
top_platform = ['ps2', 'x360', 'ps3', 'wii']
```

[31]:
```
df_years_platform_top = df_years_platform.query('platform in @top_platform')
```

[32]:
```
line,ax = plt.subplots(figsize=(15,5))
ax = sns.lineplot(x="year_of_release", y="total_sales",␣
 ↪data=df_years_platform_top, palette="pastel", hue="platform")
plt.title('Graph of the total number of sales of TOP-4 games')
plt.show()
```



From 2000 to 2005, ps2 was in the top, in the period 2005-2010 - wii, from 2010 - ps3 and x360 are approximately on the same level

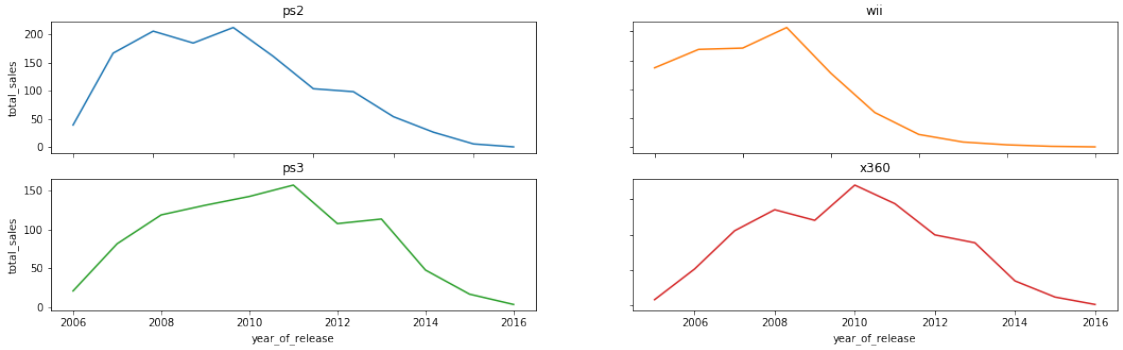
Let's build a sales chart for each TOP-4 platform

[457]:
```
ps2 = df_years_platform_top.query('platform == "ps2"')
x360 = df_years_platform_top.query('platform == "x360"')
ps3 = df_years_platform_top.query('platform == "ps3"')
wii = df_years_platform_top.query('platform == "wii"')

fig, axs = plt.subplots(2, 2, figsize=(18, 5))
axs[0, 0].plot(ps2['year_of_release'], ps2['total_sales'])
axs[0, 0].set_title('ps2')
axs[0, 1].plot(wii['year_of_release'], wii['total_sales'], 'tab:orange')
axs[0, 1].set_title('wii')
axs[1, 0].plot(ps3['year_of_release'], ps3['total_sales'], 'tab:green')
axs[1, 0].set_title('ps3')
axs[1, 1].plot(x360['year_of_release'], x360['total_sales'], 'tab:red')
axs[1, 1].set_title('x360')

for ax in axs.flat:
    ax.set(xlabel='year_of_release', ylabel='total_sales')
```

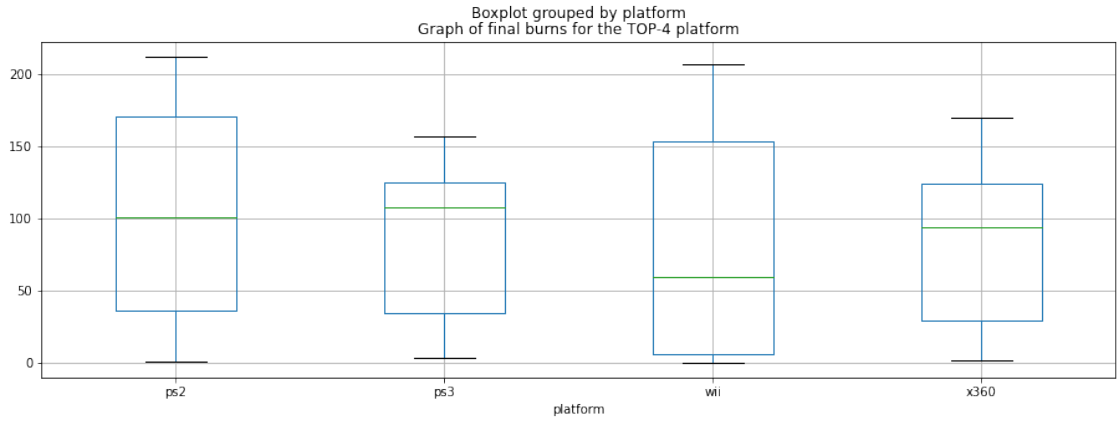
```
# Hide x labels and tick labels for top plots and y ticks for right plots.
for ax in axs.flat:
    ax.label_outer()
```



Since 2013, sales across all platforms have been falling.

[33]:
```
df_years_platform_top.boxplot('total_sales', by='platform', figsize=(15, 5))
plt.title('Graph of final burns for the TOP-4 platform')
```

[33]:
```
Text(0.5, 1.0, 'Graph of final burns for the TOP-4 platform')
```



The chart displayed the top 4 platforms. The highest median platforms are ps3 and ps2

Find TOP platforms since 2014

To plan a future advertising campaign, we will focus on the data of the last 3 years. I think it will be more correct than just focusing on the previous year

[36]:
```
df_years_platform_2014 = df.pivot_table(index=['year_of_release', 'platform'],␣
↪values='total_sales', aggfunc='sum').reset_index()
df_years_platform_2014 = df_years_platform_2014.query('year_of_release >= 2014')
```

```
df_years_platform_2014.groupby('platform')['total_sales'].sum().reset_index().
↪sort_values(by='total_sales', ascending=False).head(4)
```

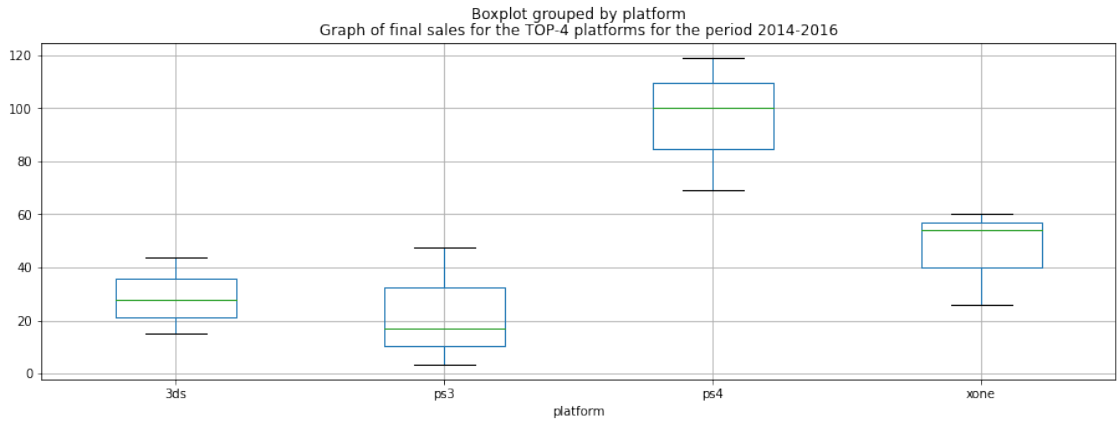
[36]:

	platform	total_sales
3	ps4	288.15
9	xone	140.36
0	3ds	86.68
2	ps3	68.18

[38]:
```
top_platform_2014 = ['ps4', 'xone', '3ds', 'ps3']
df_years_platform_2014 = df_years_platform_2014.query('platform in␣
↪@top_platform_2014')

df_years_platform_2014.boxplot('total_sales', by='platform', figsize=(15, 5))
plt.title('Graph of final sales for the TOP-4 platforms for the period␣
↪2014-2016')
```

[38]:
```
Text(0.5, 1.0, 'Graph of final sales for the TOP-4 platforms for the period
2014-2016')
```



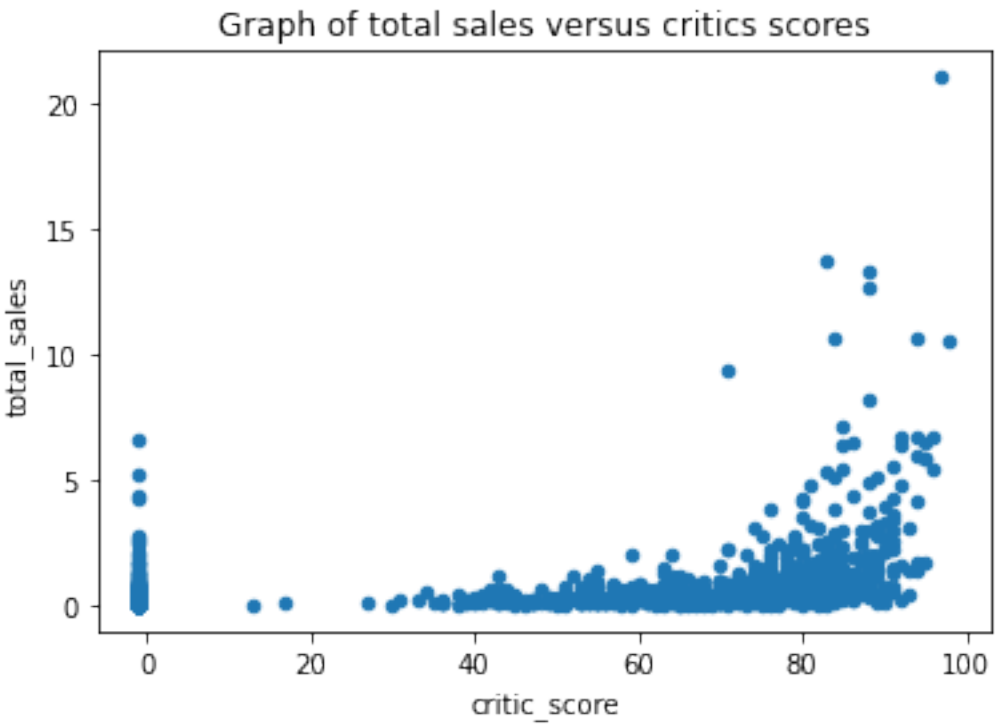
The number of games released for PS4 since 2014 far exceeds all other platforms. Second in sales - xone

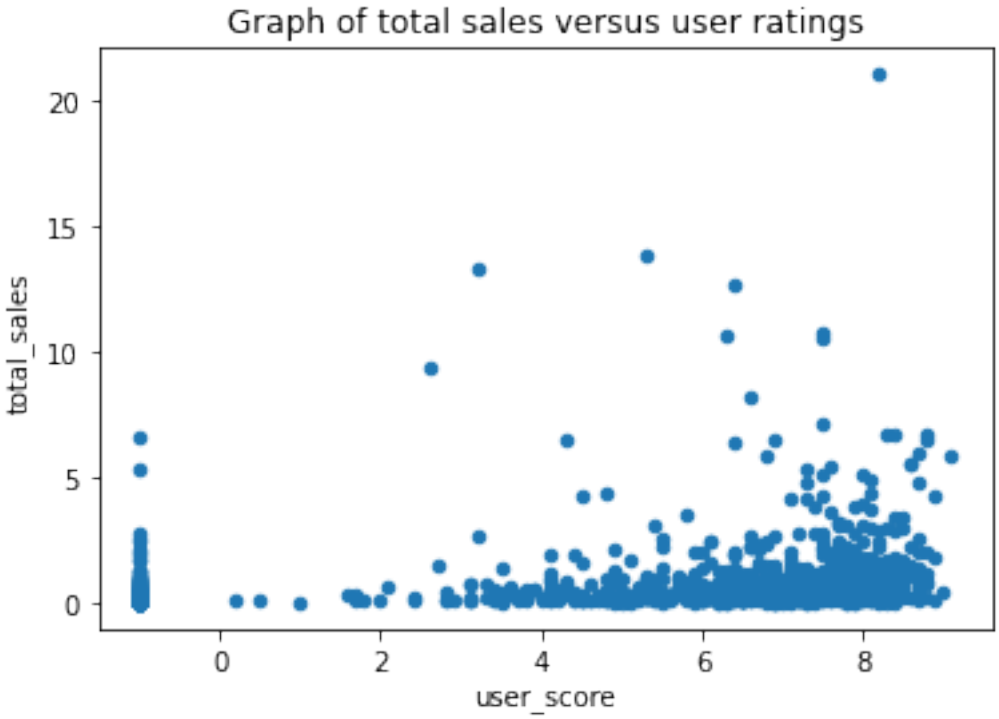
Let's build a scatterplot for the ps3 platform

[39]:
```
df_ps3 = df.query('platform == "ps3"')
```

[40]:
```
df_ps3.plot.scatter(x='critic_score', y='total_sales')
plt.title('Graph of total sales versus critics scores')
df_ps3.plot.scatter(x='user_score', y='total_sales')
plt.title('Graph of total sales versus user ratings')
```

[40]:
```
Text(0.5, 1.0, 'Graph of total sales versus user ratings')
```


Graph of total sales versus critics scores
total_sales
critic_score
0
5
10
15
20
0
20
40
60
80
100


Graph of total sales versus user ratings
total_sales
user_score
0
5
10
15
20
0
2
4
6
8

Regarding ps3:

From the scatterplot, it can be seen that sales are highly dependent on critical scores. Up to a score of 70, there are no outliers. After a score of 80, big sales begin to appear. If a game's critic score is less than 80, there's a good chance the game won't get more than 5 million sales. The goal is to score over 80 points

User rating is not so critical for sales. Of course, it affects sales, but outliers when evaluating users are more frequent, the game can collect a lot of sales even with an average user rating. The chance is not high, but nevertheless it is

[41]:
```
df_ps3[['critic_score', 'user_score', 'total_sales']].corr()
```

[41]:

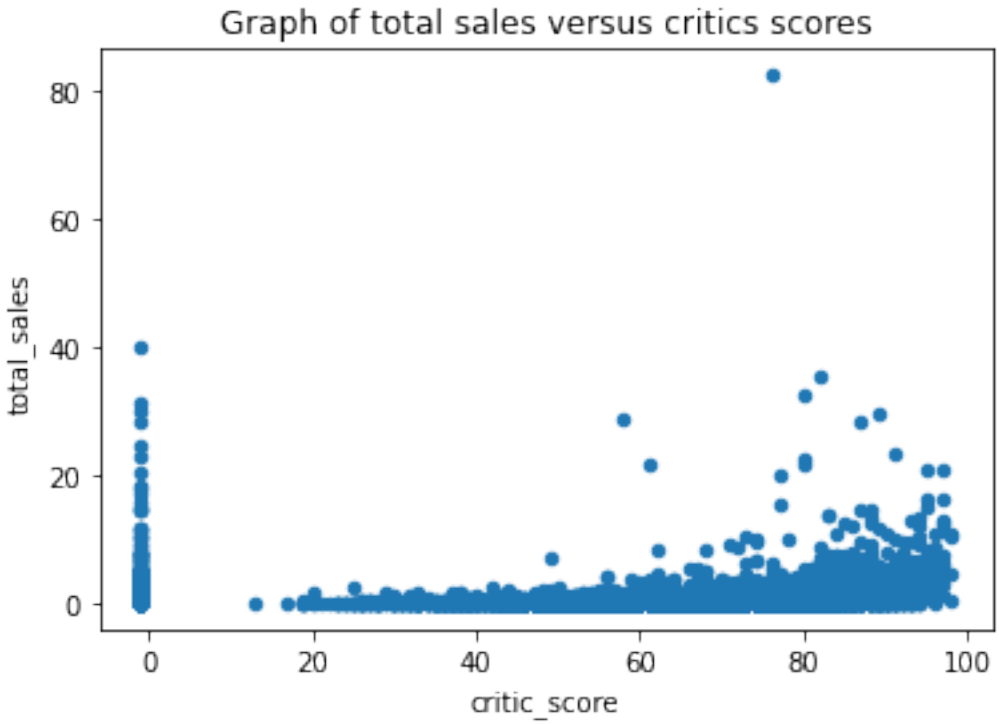
	critic_score	user_score	total_sales
critic_score	1.000000	0.823671	0.353793
user_score	0.823671	1.000000	0.272669
total_sales	0.353793	0.272669	1.000000

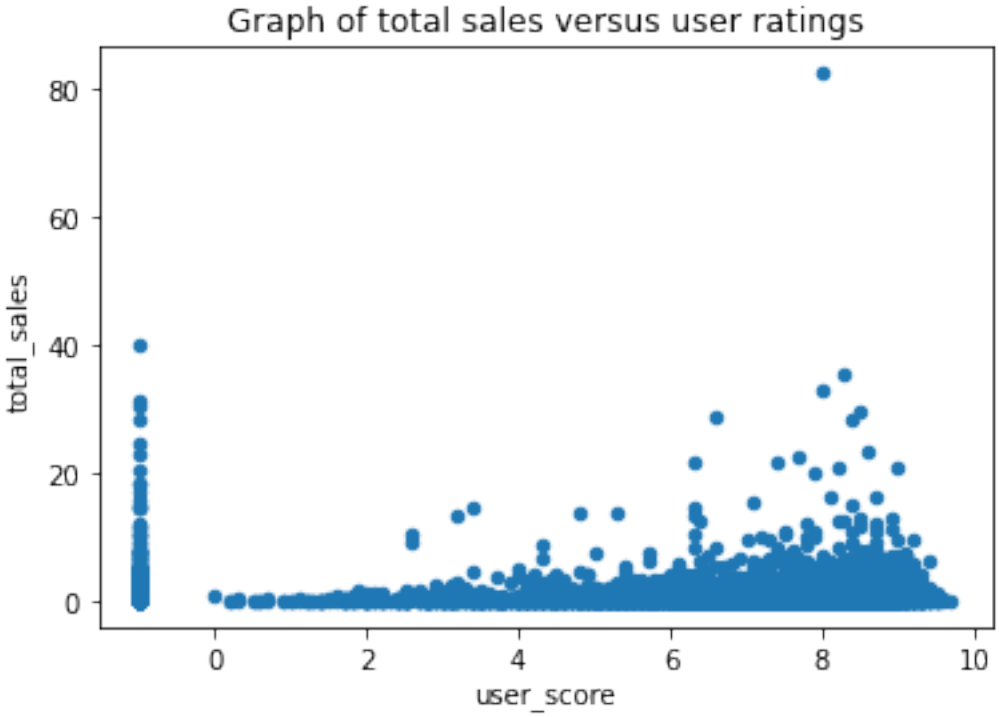
Dependence on the rating of critics is higher than on the rating of users. What I wrote above

Let's build a scatterplot for all platforms

[45]:
```
df.plot.scatter(x='critic_score', y='total_sales')
plt.title('Graph of total sales versus critics scores')
df.plot.scatter(x='user_score', y='total_sales')
plt.title('Graph of total sales versus user ratings')
```

[45]:
```
Text(0.5, 1.0, 'Graph of total sales versus user ratings')
```





[465]:
```
df[['critic_score', 'user_score', 'total_sales']].corr()
```

[465]:

	critic_score	user_score	total_sales
critic_score	1.000000	0.818959	0.148076
user_score	0.818959	1.000000	0.135123
total_sales	0.148076	0.135123	1.000000

The correlation shows that the dependence of final sales on critics is slightly higher than on users. However, both dependencies are rather weak.

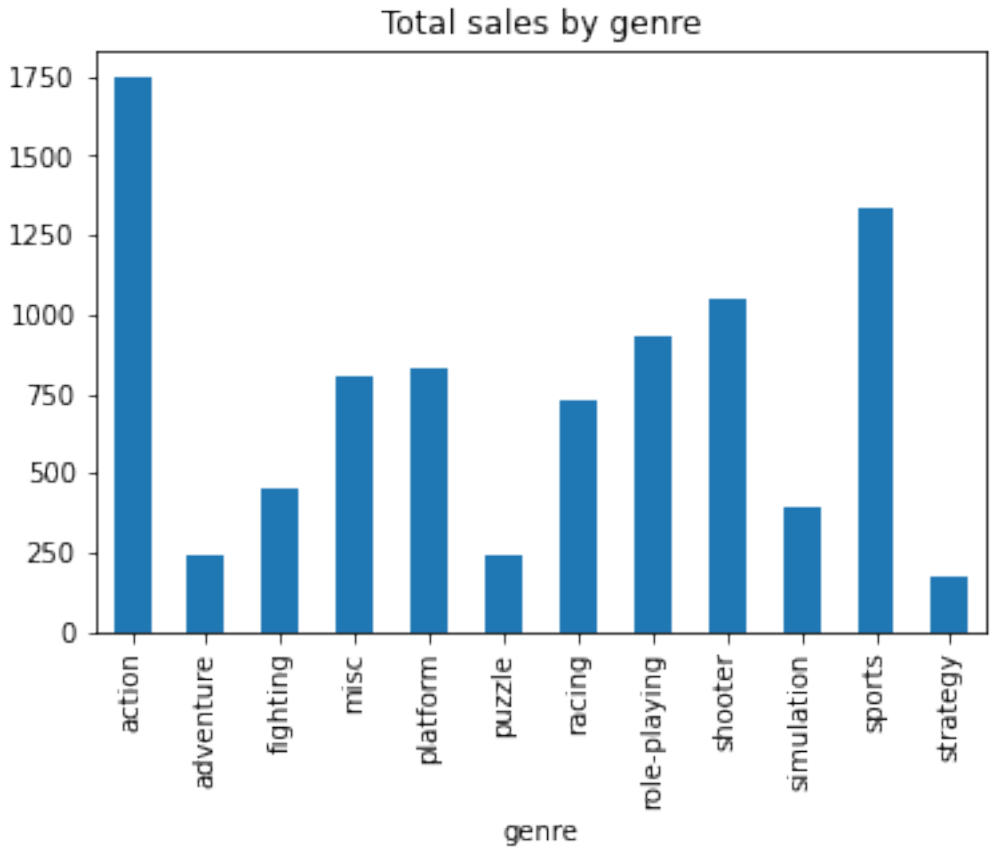
Let's see the distribution by genre

[46]:
```
df_genre = df.pivot_table(index=['year_of_release', 'genre'],␣
↪values='total_sales', aggfunc='sum').reset_index()
```

[47]:
```
#line,ax = plt.subplots(figsize=(15,10))
#ax = sns.lineplot(x="year_of_release", y="total_sales", data=df_genre,␣
↪palette="pastel", hue="genre")
#plt.title('              -                  1980 - 2016  ')
#plt.show()
```

[48]:
```
df_genre.groupby('genre')['total_sales'].sum().plot(kind='bar')
plt.title('Total sales by genre')
```

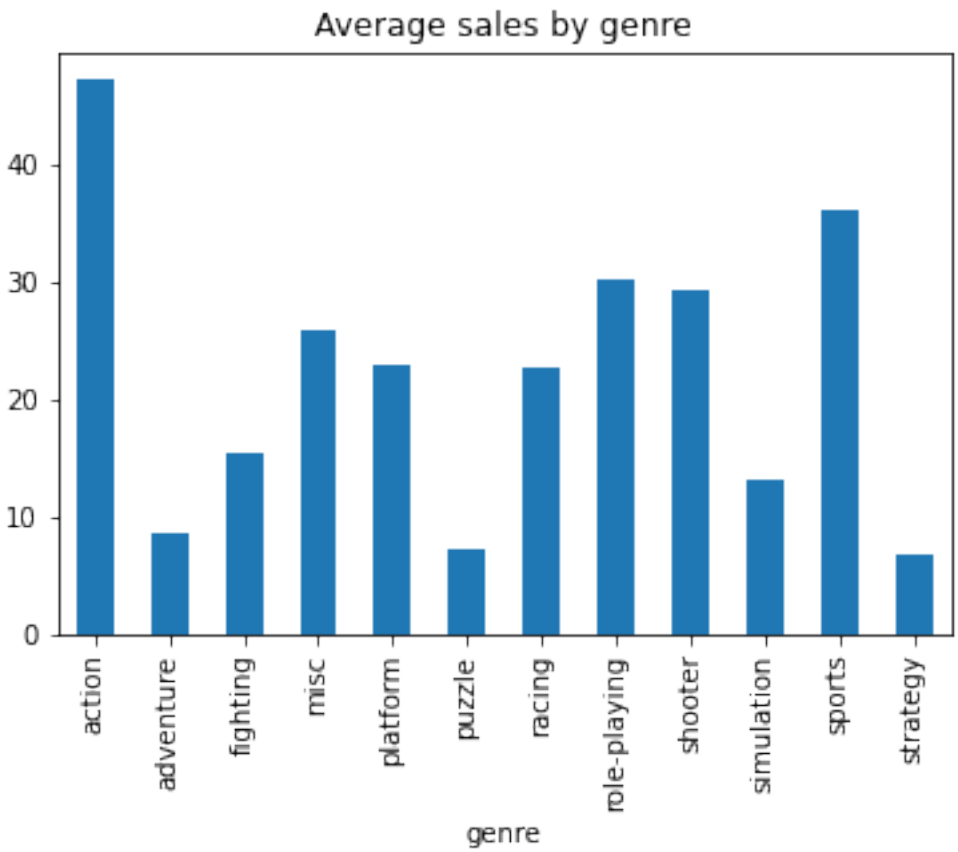
[48]:
```
Text(0.5, 1.0, 'Total sales by genre')
```



Action stands out from other genres.

[49]:
```
df_genre.groupby('genre')['total_sales'].mean().plot(kind='bar')
plt.title('Average sales by genre')
```

[49]:
```
Text(0.5, 1.0, 'Average sales by genre')
```



By average values, the picture is the same as by the sum

1.3.2 Conclusion

From the graphs, we learned the most successful year for game sales, the most popular genre, and the dependence on critic and user reviews.

1.4 User portrait of each region

Most popular platforms (top 5)

[50]:
```
df.columns
```

[50]:
```
Index(['name', 'platform', 'year_of_release', 'genre', 'na_sales', 'eu_sales',
       'jp_sales', 'other_sales', 'critic_score', 'user_score', 'rating',
       'total_sales'],
      dtype='object')
```

[61]:
```
```

```
df_flatform_refion = df.pivot_table(index='platform', values=['na_sales',␣
↪'eu_sales', 'jp_sales'], aggfunc='sum').reset_index()
df_flatform_refion['na_sales_%'] = df_flatform_refion['na_sales'] /␣
↪df_flatform_refion['na_sales'].sum()
df_flatform_refion['eu_sales_%'] = df_flatform_refion['eu_sales'] /␣
↪df_flatform_refion['eu_sales'].sum()
df_flatform_refion['jp_sales_%'] = df_flatform_refion['jp_sales'] /␣
↪df_flatform_refion['jp_sales'].sum()

display(df_flatform_refion.sort_values('na_sales_%', ascending=False).head())
print()
display(df_flatform_refion.sort_values('eu_sales_%', ascending=False).head())
print()
display(df_flatform_refion.sort_values('jp_sales_%', ascending=False).head())
```

	platform	eu_sales	jp_sales	na_sales	na_sales_%	eu_sales_%	jp_sales_%
28	x360	270.76	12.43	602.47	0.136907	0.111693	0.009581
16	ps2	339.29	139.20	583.84	0.132674	0.139963	0.107296
25	wii	262.21	69.33	496.90	0.112917	0.108166	0.053440
17	ps3	330.29	80.19	393.49	0.089418	0.136250	0.061811
4	ds	188.89	175.57	382.40	0.086898	0.077920	0.135331

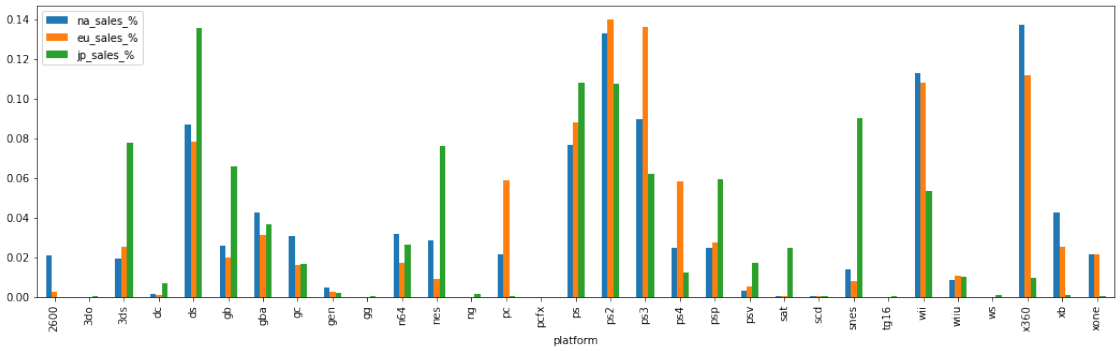
	platform	eu_sales	jp_sales	na_sales	na_sales_%	eu_sales_%	jp_sales_%
16	ps2	339.29	139.20	583.84	0.132674	0.139963	0.107296
17	ps3	330.29	80.19	393.49	0.089418	0.136250	0.061811
28	x360	270.76	12.43	602.47	0.136907	0.111693	0.009581
25	wii	262.21	69.33	496.90	0.112917	0.108166	0.053440
15	ps	213.61	139.82	336.52	0.076472	0.088118	0.107774

	platform	eu_sales	jp_sales	na_sales	na_sales_%	eu_sales_%	jp_sales_%
4	ds	188.89	175.57	382.40	0.086898	0.077920	0.135331
15	ps	213.61	139.82	336.52	0.076472	0.088118	0.107774
16	ps2	339.29	139.20	583.84	0.132674	0.139963	0.107296
23	snes	19.04	116.55	61.23	0.013914	0.007854	0.089838
2	3ds	61.48	100.67	83.49	0.018973	0.025362	0.077597

Among the top best platforms, we will highlight the ps2 platform, which is popular in all three regions

[62]:
```
df_flatform_refion[['platform', 'na_sales_%', 'eu_sales_%', 'jp_sales_%']].
↪plot(kind='bar', x='platform', figsize=(18,5))
```

[62]: <AxesSubplot:xlabel='platform'>



Most popular genres (top 5)

[63]:
```
df_genre_refion = df.pivot_table(index='genre', values=['na_sales', 'eu_sales',␣
 ↪'jp_sales'], aggfunc='sum').reset_index()
df_genre_refion['na_sales_%'] = df_genre_refion['na_sales'] /␣
 ↪df_genre_refion['na_sales'].sum()
df_genre_refion['eu_sales_%'] = df_genre_refion['eu_sales'] /␣
 ↪df_genre_refion['eu_sales'].sum()
df_genre_refion['jp_sales_%'] = df_genre_refion['jp_sales'] /␣
 ↪df_genre_refion['jp_sales'].sum()

display(df_genre_refion.sort_values('na_sales_%', ascending=False).head())
print()
display(df_genre_refion.sort_values('eu_sales_%', ascending=False).head())
print()
display(df_genre_refion.sort_values('jp_sales_%', ascending=False).head())
```

	genre	eu_sales	jp_sales	na_sales	na_sales_%	eu_sales_%	jp_sales_%
0	action	519.13	161.43	879.01	0.199749	0.214150	0.124432
10	sports	376.79	135.54	684.43	0.155532	0.155432	0.104475
8	shooter	317.34	38.76	592.24	0.134583	0.130908	0.029877
4	platform	200.35	130.83	445.50	0.101237	0.082648	0.100845
3	misc	212.74	108.11	407.27	0.092549	0.087759	0.083332

	genre	eu_sales	jp_sales	na_sales	na_sales_%	eu_sales_%	jp_sales_%
0	action	519.13	161.43	879.01	0.199749	0.214150	0.124432
10	sports	376.79	135.54	684.43	0.155532	0.155432	0.104475
8	shooter	317.34	38.76	592.24	0.134583	0.130908	0.029877
6	racing	236.51	56.71	359.35	0.081660	0.097564	0.043713
3	misc	212.74	108.11	407.27	0.092549	0.087759	0.083332

```
      genre  eu_sales  jp_sales  na_sales  na_sales_%  eu_sales_%  \
```

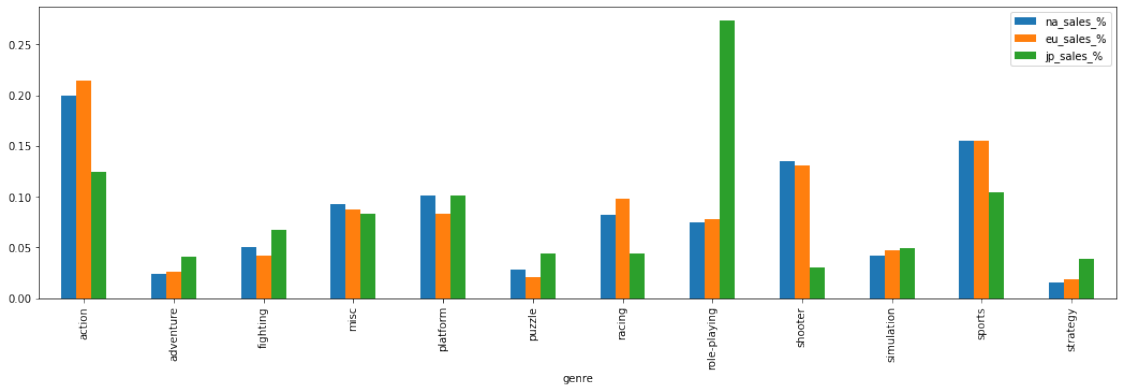
```
7   role-playing    188.71    355.41    330.81    0.075174    0.077846
0         action    519.13    161.43    879.01    0.199749    0.214150
10        sports    376.79    135.54    684.43    0.155532    0.155432
4       platform    200.35    130.83    445.50    0.101237    0.082648
3           misc    212.74    108.11    407.27    0.092549    0.087759

    jp_sales_%
7     0.273953
0     0.124432
10    0.104475
4     0.100845
3     0.083332
```

Each region has roughly the same top game genres

[64]:
```
df_genre_refion[['genre', 'na_sales_%', 'eu_sales_%', 'jp_sales_%']].
↪plot(kind='bar', x='genre', figsize=(18,5))
```

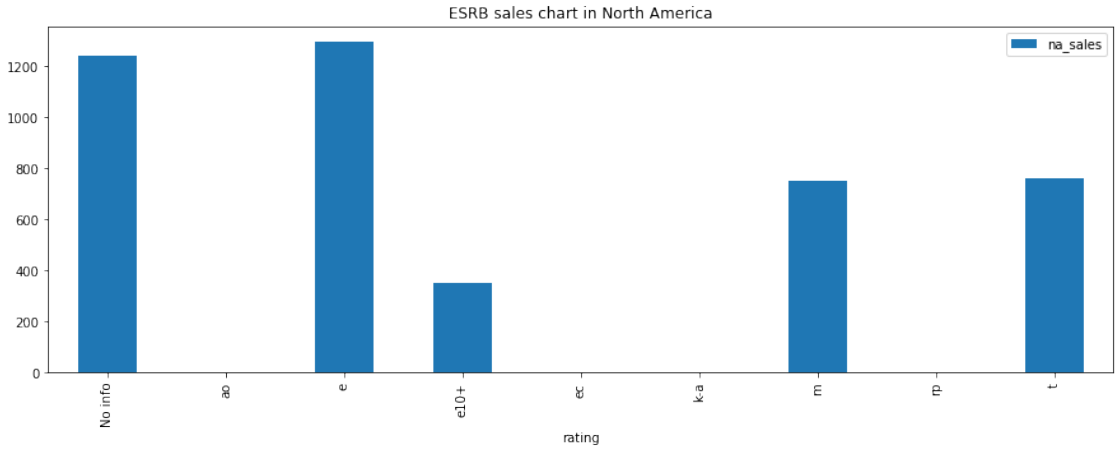
[64]:
```
<AxesSubplot:xlabel='genre'>
```



Let's see if the ESRB rating affects the example of the North American market

[65]:
```
df_esrb_na = df.groupby('rating')['na_sales'].sum().reset_index()
df_esrb_na.plot(kind='bar', x='rating', figsize=(15,5))
plt.title('ESRB sales chart in North America')
```

[65]:
```
Text(0.5, 1.0, 'ESRB sales chart in North America')
```



Rating definitely has an impact.

Rating “e” - Everyone - the most popular. What does the name actually say.

1.4.1 Conclusion

We found the top of the best platforms in each region and the top of the best genres.

1.5 Hypothesis testing

1.5.1 Hypothesis 1

H0 - Average user rating of the Xbox One platform = average user rating of the PC platform H1 - Average user rating of the Xbox One platform average user rating of the PC platform

First, let’s create a table without negative values, because instead of a pass left -1

[66]:
```
df_h = df.query('user_score >= 0')
```

[67]:
```
df_h_xone = df_h.query('platform == "xone"')

df_h_pc = df_h.query('platform == "pc"')
```

Testing the hypothesis

[68]:
```
alpha = .01

results = st.ttest_ind(
    df_h_xone['user_score'],
    df_h_pc['user_score'],
    equal_var = False)

print('p-value: ', results.pvalue)
```

```
if results.pvalue < alpha:
    print("Rejecting the null hypothesis ")
else:
    print("Failed to reject the null hypothesis ")
```

```
p-    :  4.935072360183565e-06
Rejecting the null hypothesis
```

The hypothesis was not confirmed. User rating averages between xone and pc are not equal

1.5.2 Hypothesis 2

H0 - Average user rating of the Action genre = average user rating of the Sports genre 1 - Average user rating of the Action genre average user rating of the Sports genre

[69]:
```
df_h_action = df_h.query('genre == "action"')
df_h_sports = df_h.query('genre == "sports"')
```

Testing the hypothesis

[70]:
```
alpha = .01

results = st.ttest_ind(
    df_h_action['user_score'],
    df_h_sports['user_score'],
    equal_var = False)

print('p-value: ', results.pvalue)

if results.pvalue < alpha:
    print("Rejecting the null hypothesis ")
else:
    print("Failed to reject the null hypothesis ")
```

```
p-    :  0.11483818791498286
Failed to reject the null hypothesis
```

The hypothesis was confirmed. The average user ratings between the Action and Sports genres are approximately equal

1.5.3 For the final task

[71]:
```
# Compare regions
print('NA', df['na_sales'].sum())
print('EU', df['eu_sales'].sum())
print('JP', df['jp_sales'].sum())
```

```
NA 4400.570000000001
EU 2424.1400000000003
JP 1297.34
```

[72]:
```
# Top platforms
print('Top-3 platforms', df.groupby('platform')['total_sales'].sum().
 ↪reset_index().sort_values('total_sales', ascending=False).head(3))
```

```
Top-3 platforms    platform  total_sales
16      ps2      1255.77
28     x360       971.42
17      ps3       939.65
```

[73]:
```
# Best genres
print('Top-3 genres', df.groupby('genre')['total_sales'].sum().reset_index().
 ↪sort_values('total_sales', ascending=False).head(3))
```

```
Top-3 genres       genre  total_sales
0    action      1744.17
10   sports      1331.27
8   shooter      1052.45
```

1.5.4 Conclusion

The hypothesis about the equality of the average user rating between the Xbox One platform and the AP was not confirmed

The hypothesis about the equality of the average user rating between the Action and Sports genres was confirmed

1.6 General conclusion

Answering the main question of the task, we list the main points: - Most games are sold in North America. - The best platforms for gaming are ps2, x360 and ps3 - Top selling genres are action, sports, shooter

Also, the criterion of success is influenced by the rating of critics and the rating of users.

These data should be taken into account in the next advertising campaign.