Integrated Project 1

Comparative analysis for an advertising campaign in the video game industry

Table of Contents

Goal

Prepare a report for the online store Ice to identify patterns that determine whether a game succeeds or not. This will allow us to spot potential big winners and plan advertising campaigns.

Hypotheses

1. Average user ratings of the Xbox One and PC platforms are the same.
2. Average user ratings for the Action and Sports genres are the same.

Description of the data

The Entertainment Software Rating Board (ESRB) evaluates a game's content and assigns an age rating such as Teen or Mature.

— Name

— Platform

— Year_of_Release

— Genre

— NA_sales (North American sales in USD million)

— EU_sales (sales in Europe in USD million)

— JP_sales (sales in Japan in USD million)

— Other_sales (sales in other countries in USD million)

— Critic_Score (maximum of 100)

— User_Score (maximum of 10)

— Rating (ESRB)

```
- E: everyone;
- E10+: everyone 10+;
- T: Teen;
- M: Mature 17+;
- A: Adults only 18+;
- RT: rating pending.
```

Data for 2016 may be incomplete.

Imports

In [2]:

```
import pandas as pd
import numpy as np
import scipy
import matplotlib
import seaborn as sns

from scipy import stats as st

import matplotlib.pyplot as plt
%matplotlib inline

import sys
import warnings
if not sys.warnoptions:
       warnings.simplefilter("ignore")

pd.set_option('display.max_rows', None)

print("Setup Complete")
```

```
Setup Complete
```

Library version check and update

In [3]:

```
version_dict = {pd:'1.0.1', np:'1.18.1', scipy:'1.6.0', matplotlib:'3.1.4'}
```

In [4]:

```
def get_value(my_key):
    """
    If the val can be found in the dictinary.values() list,
    returns the key of the dictionary item in which the val was found.
    """
    wrong_val = []

    for key, value in version_dict.items():
        try:
            if key == my_key:
                return value
        except:
            wrong_val.append(key, value)
```

In [5]:

```
for lib in version_dict.keys():
    if lib.__version__ != get_value(lib):
        print("Warning: to be able to run this code correctly, please install th
e version", get_value(lib), 'of', lib)
```

```
Warning: to be able to run this code correctly, please install the v
ersion 3.1.4 of <module 'matplotlib' from '/anaconda3/lib/python3.7/
site-packages/matplotlib/__init__.py'>
```

Input data

In [6]:

```
try:
    df = pd.read_csv('games.csv')
except:
    df = pd.read_csv('/datasets/games.csv')
```

Descriptive statistics

In [7]:

```
df.head(10)
```

Out[7]:

	Name	Platform	Year_of_Release	Genre	NA_sales	EU_sales	JP_sales	Other_sa
0	Wii Sports	Wii	2006.0	Sports	41.36	28.96	3.77	8
1	Super Mario Bros.	NES	1985.0	Platform	29.08	3.58	6.81	0
2	Mario Kart Wii	Wii	2008.0	Racing	15.68	12.76	3.79	3
3	Wii Sports Resort	Wii	2009.0	Sports	15.61	10.93	3.28	2
4	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	11.27	8.89	10.22	1
5	Tetris	GB	1989.0	Puzzle	23.20	2.26	4.22	0
6	New Super Mario Bros.	DS	2006.0	Platform	11.28	9.14	6.50	2
7	Wii Play	Wii	2006.0	Misc	13.96	9.18	2.93	2
8	New Super Mario Bros. Wii	Wii	2009.0	Platform	14.44	6.94	4.70	2
9	Duck Hunt	NES	1984.0	Shooter	26.93	0.63	0.28	0

Notes for preprocessing:

- We see "NaN"s in the last 3 columns;
- `Year_of_Release` data type should be changed to integer.

In [8]:

```
df.info()
```

```
<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
```

Notes for data preprocessing:

- The data set contains almost 17k observations and 11 columns;
- `Name`, `Year_of_Release`, `Genre`, `Critic_Score`, `User_Score` and `Rating` columns contain missing values;
- `Critic_Score` data type should be changed to integer (as there is no need for floating points in this column);
- `User_Score` data type should be changed from object to float;
- Column names should all be converted to lower case letters.

In [9]:

```
df.describe()
```

Out[9]:

	Year_of_Release	NA_sales	EU_sales	JP_sales	Other_sales	Critic_Score
count	16446.000000	16715.000000	16715.000000	16715.000000	16715.000000	8137.000000
mean	2006.484616	0.263377	0.145060	0.077617	0.047342	68.967679
std	5.877050	0.813604	0.503339	0.308853	0.186731	13.938165
min	1980.000000	0.000000	0.000000	0.000000	0.000000	13.000000
25%	2003.000000	0.000000	0.000000	0.000000	0.000000	60.000000
50%	2007.000000	0.080000	0.020000	0.000000	0.010000	71.000000
75%	2010.000000	0.240000	0.110000	0.040000	0.030000	79.000000
max	2016.000000	41.360000	28.960000	10.220000	10.570000	98.000000

Notes for data preprocessing:

- `Year_of_Release` ranges from 1980 to 2016, an average game was released in 2006, while most of them - in 2007. Since we don't see a big difference between mean and medium values while the standard deviation is quite low, we can assume the distribution of this variable is close to normal;
- All sales variables have a minimum value of 0, it should be checked and corrected if needed. They all looks like a very much positively skewed distributions - even the third quartile is less than 1 million USD while the max value is from 11 to 42 millions. These distributions should also be further analyzed with box plots for outliers check;
- `Critic_Score` ranges from 13 to 98. The average score is close to 67, the medium value is around 71, standard deviation is reasonable - 14, hence we can assume that this distribution is close to normal.

Preprocessing

Column names

In [10]:

```
df.columns = df.columns.str.lower()
```

Missing values

In [11]:

```
df.isnull().sum()/len(df)
```

Out[11]:

```
name               0.000120
platform           0.000000
year_of_release    0.016093
genre              0.000120
na_sales           0.000000
eu_sales           0.000000
jp_sales           0.000000
other_sales        0.000000
critic_score       0.513192
user_score         0.400897
rating             0.404786
dtype: float64
```

Name

In [12]:

```
df[df['name'].isnull()]
```

Out[12]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	criti
659	NaN	GEN	1993.0	NaN	1.78	0.53	0.00	0.08	
14244	NaN	GEN	1993.0	NaN	0.00	0.00	0.03	0.00	

These two games are quite old, most of data is missing and sales are very low. We will exclude these 2 observations from our analysis.

In [13]:

```
df = df.dropna(subset=['name'], axis=0)
df.reset_index(drop=True, inplace=True)

df.shape
```

Out[13]:

```
(16713, 11)
```

Release year

In [14]:

```
df[df['year_of_release'].isnull()].head()
```

Out[14]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales
183	Madden NFL 2004	PS2	NaN	Sports	4.26	0.26	0.01	0.71
377	FIFA Soccer 2004	PS2	NaN	Sports	0.59	2.36	0.04	0.51
456	LEGO Batman: The Videogame	Wii	NaN	Action	1.80	0.97	0.00	0.29
475	wwe Smackdown vs. Raw 2006	PS2	NaN	Fighting	1.57	1.02	0.00	0.41
609	Space Invaders	2600	NaN	Shooter	2.36	0.14	0.00	0.03

According to our previous analysis, `year_of_release` 's distribution is close to normal. We will then replace the missing values with the mean per genre and platform.

In [15]:

```
df['year_of_release'] = df.groupby(['platform','genre'])['year_of_release'].appl
y(lambda x: x.fillna(x.mean()))
```

In [16]:

```
df['year_of_release'].isnull().sum()
```

Out[16]:

0

Let's change the data type to integer as years are not float type.

In [17]:

```
df['year_of_release'] = np.floor(df['year_of_release']).astype(int)
```

Scores

There are some games for which all 3 scorings are missing. These games were released on old consoles that are practically not used any more. Besides, our goal is to build a prognosis for 2017, so the data from the 80s is not going to be very useful. We will keep these values they way they are and make a decision how to deal with them later if we need it.

For the rest of missing values, we will fill them based on the score column that is available. If the correlation between `critic_score` and `user_score` is high enough, then we can just pick a certain game, look for games with similar user score (plus/minus 0.5 points) -- get an array of such games and then simply take median value for `critic_score` .

First, it will be necessary to change `user_score` data type from object to float.

In [18]:

```
df['user_score'].unique()
```

Out[18]:

```
array(['8', nan, '8.3', '8.5', '6.6', '8.4', '8.6', '7.7', '6.3',
'7.4',
       '8.2', '9', '7.9', '8.1', '8.7', '7.1', '3.4', '5.3', '4.8',
'3.2',
       '8.9', '6.4', '7.8', '7.5', '2.6', '7.2', '9.2', '7', '7.3',
'4.3',
       '7.6', '5.7', '5', '9.1', '6.5', 'tbd', '8.8', '6.9', '9.4',
'6.8',
       '6.1', '6.7', '5.4', '4', '4.9', '4.5', '9.3', '6.2', '4.2',
'6',
       '3.7', '4.1', '5.8', '5.6', '5.5', '4.4', '4.6', '5.9', '3.
9',
       '3.1', '2.9', '5.2', '3.3', '4.7', '5.1', '3.5', '2.5', '1.
9', '3',
       '2.7', '2.2', '2', '9.5', '2.1', '3.6', '2.8', '1.8', '3.8',
'0',
       '1.6', '9.6', '2.4', '1.7', '1.1', '0.3', '1.5', '0.7', '1.
2',
       '2.3', '0.5', '1.3', '0.2', '0.6', '1.4', '0.9', '1', '9.7'],
      dtype=object)
```

Apart from missing values, we see some 'tbd'. It basically means that the rating is missing, so let's replace these values with "NaN"

In [19]:

```
df.loc[df['user_score'] == 'tbd', 'user_score'] = np.nan
```

Now let's check the correlation between the two columns.

In [20]:

```
df['critic_score'] = (df['critic_score'].astype(float))
df['user_score'] = df['user_score'].astype(float)
df['user_score'].corr(df['critic_score'])
```

Out[20]:

```
0.5808778320767239
```

It is quite high, so we can use the approach mentioned above. We will fill missing values in the `user_score` column using the `critic_score` column.

In [21]:

```
df['user_score'] = df.groupby('critic_score')['user_score'].apply(lambda x: x.fi
llna(x.median()))
```

In [22]:

```
df['user_score'] = df.groupby('rating')['user_score'].apply(lambda x: x.fillna(x
.median()))
df['critic_score'] = df.groupby('rating')['critic_score'].apply(lambda x: x.fill
na(x.median()))
```

Now, let's see if there are any games that are sold on different platforms. If the rating is filled on one of the platforms we can fill missing rating on other platforms as well.

In [23]:

```
df['user_score'] = df.groupby('name')['user_score'].apply(lambda x: x.fillna(x.m
ean()))
df['critic_score'] = df.groupby('name')['critic_score'].apply(lambda x: x.fillna
(x.mean()))
df['rating'] = df.groupby('name')['rating'].apply(lambda x: x.fillna(x.mode()))
```

In [24]:

```
df.isnull().sum()
```

Out[24]:

```
name                  0
platform              0
year_of_release       0
genre                 0
na_sales              0
eu_sales              0
jp_sales              0
other_sales           0
critic_score       6337
user_score         6337
rating             6764
dtype: int64
```

We will keep the rest of missing values for now.

Duplicates

Let's check if any rows are duplicated in any data frames.

In [25]:

```
df.duplicated().sum()
```

Out[25]:

```
0
```

Data type change

In [26]:

```
df.dtypes
```

Out[26]:

```
name                object
platform            object
year_of_release      int64
genre               object
na_sales           float64
eu_sales           float64
jp_sales           float64
other_sales        float64
critic_score       float64
user_score         float64
rating              object
dtype: object
```

All variables now have correct data types.

Total sales calculation

In [27]:

```
df['total_sales'] = df['na_sales'] + df['eu_sales'] + df['jp_sales'] + df['other
_sales']
```

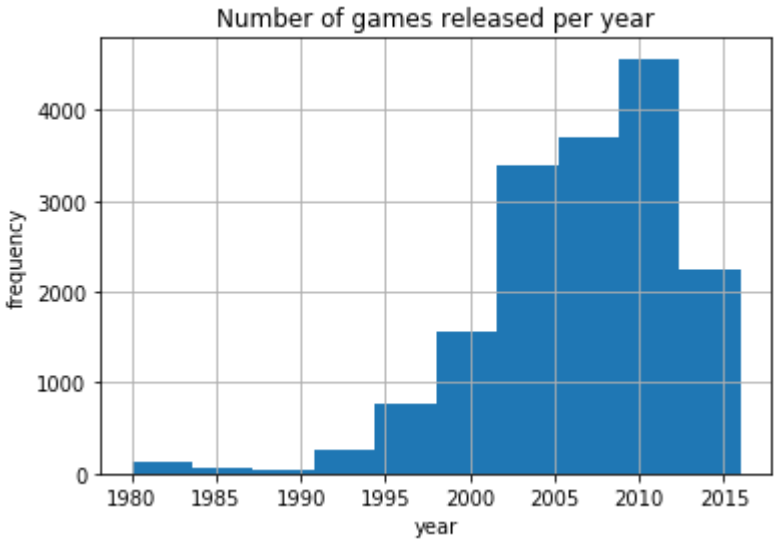
EDA

All platforms

Dynamics in game realizations

In [28]:

```
df.hist('year_of_release')
plt.title('Number of games released per year')
plt.xlabel('year')
plt.ylabel('frequency');
```

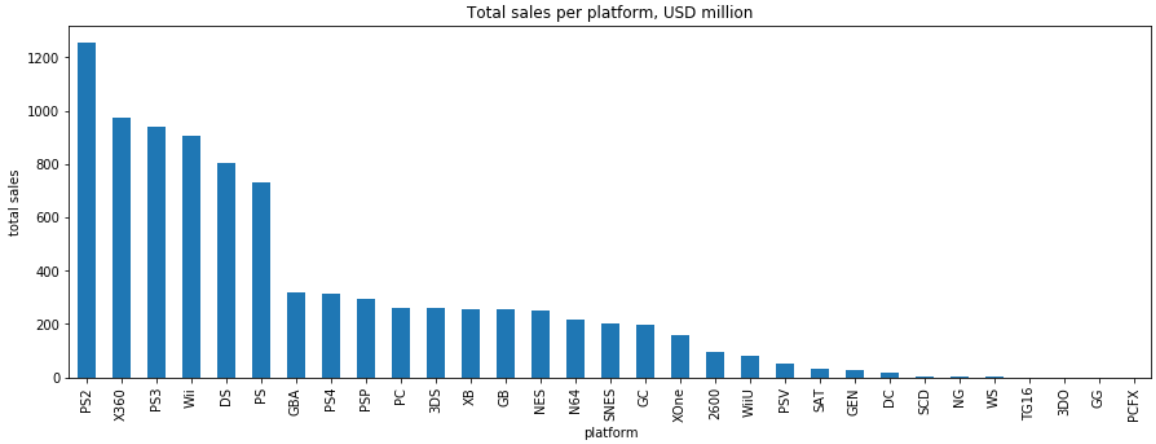


Most games in this data set were released around 2009-2012. The number of games released before the 21st century is not very significant.

Total sales per platform

In [29]:

```
plt.figure(figsize=(15,5))
df.groupby('platform')['total_sales'].sum().sort_values(ascending=False).plot(ki
nd='bar')
plt.title('Total sales per platform, USD million')
plt.ylabel('total sales');
```

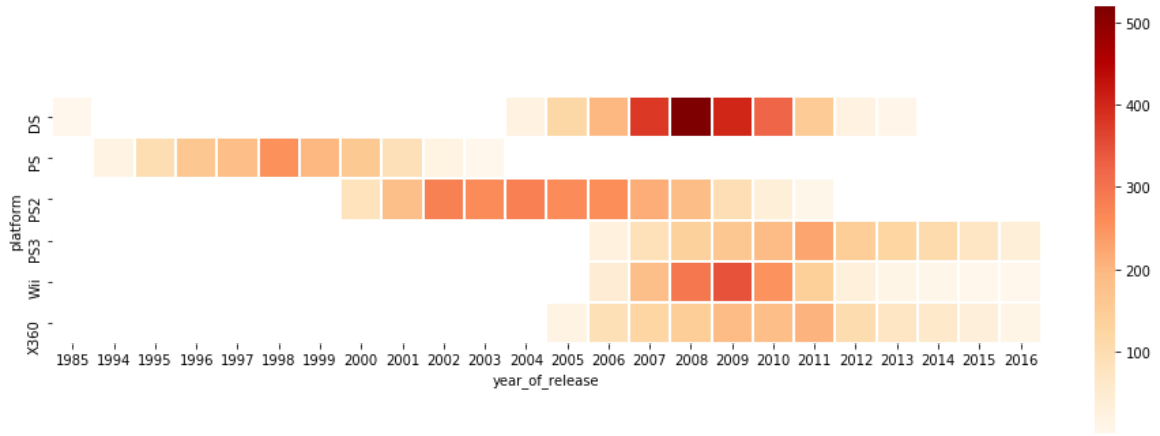


There are 6 platforms that stand out in terms of total sales in all regions: 'PS2','X360','PS3','Wii','DS','PS'. Let's build separate distributions for each of them to see how many games were released each year.

Dynamics in game realizations per platform

In [30]:

```
df_platform_year = df[df['platform'].isin(['PS2','X360','PS3','Wii','DS','PS'])]
.groupby(["platform","year_of_release"]).size().unstack()

plt.figure(figsize=(14,10))
g = sns.heatmap(
    df_platform_year,
    square=True, # make cells square
    cbar_kws={'fraction' : 0.02}, # shrink colour bar
    cmap='OrRd', # use orange/red colour map
    linewidth=1 # space between cells
)
```



Out of these 6 platforms only 3 still have sales - 'X360', 'PS3' and 'Wii' (very low amount from 2012 to 2016). The other 3 platforms used to be very popular but now (in 2016) have zero sales. On average it takes around 10 years for new platforms to appear and for the old ones to fade away. If we would like to build a prognosis for 2017, we need to identify those platforms that are still popular in 2016. The middle of a platform's lifespan when it's on the peak of popularity is 5 years from the beginning, so we will take data for the past 5 years to build a prognosis for 2017.

In [31]:

```
df_final = df[df['year_of_release'] >= 2012]
df_final.shape
```

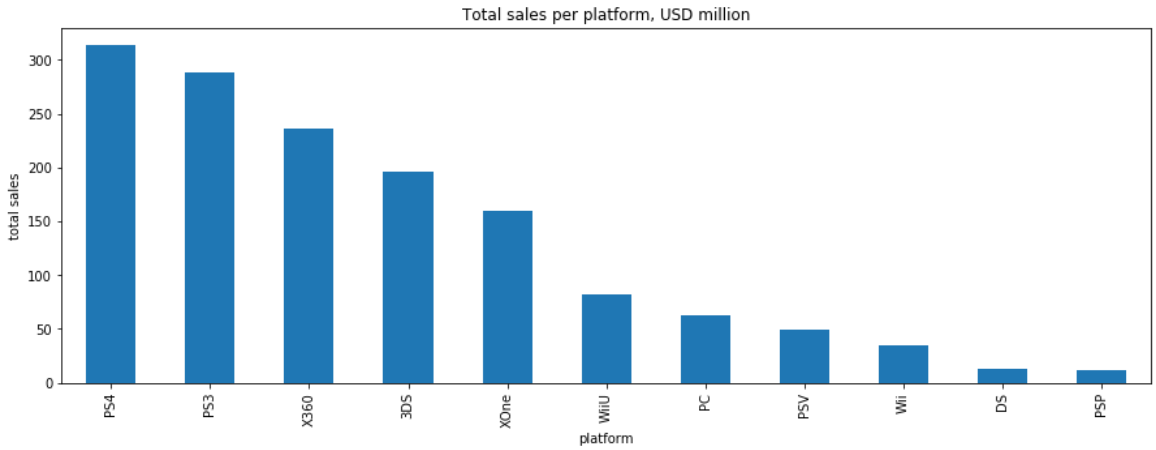
Out[31]:

```
(2893, 12)
```

Most popular platforms selection

In [32]:

```
plt.figure(figsize=(15,5))
df_final.groupby('platform')['total_sales'].sum().sort_values(ascending=False).p
lot(kind='bar')
plt.title('Total sales per platform, USD million')
plt.ylabel('total sales');
```



In this data frame the following 5 platforms are leading in sales: 'PS4', 'PS3', 'X360', '3DS', 'XOne'. Let's choose them for further analysis.

We will now build a box plot for the global sales of all games, broken down by platform in order to see if there are any differences in sales.

Top platforms

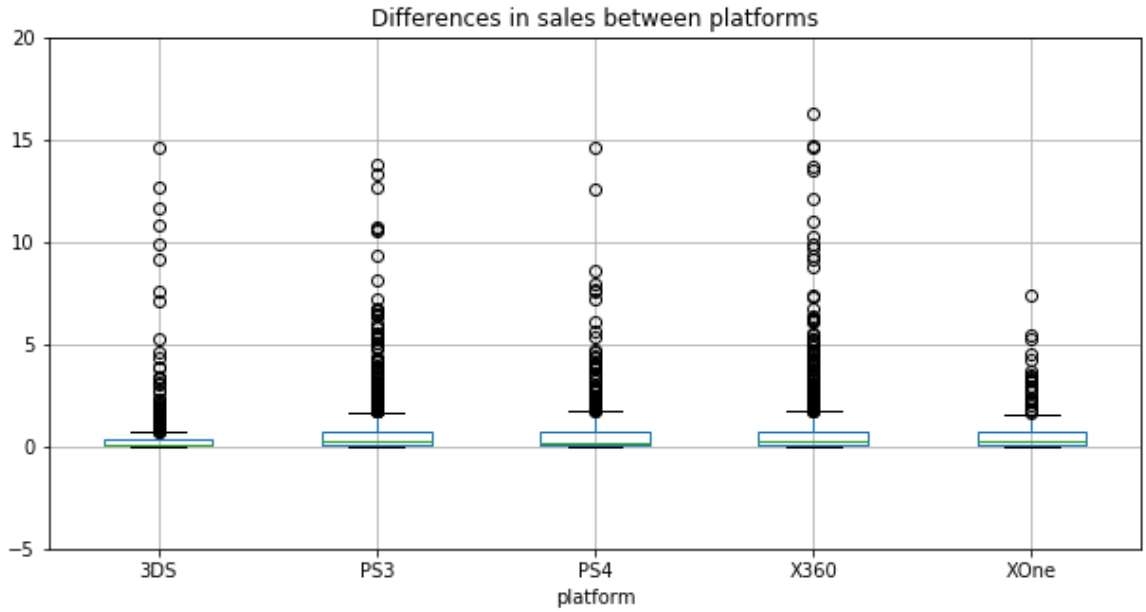
Differences in sales between platforms

In [33]:

```
df_top_platforms = df[df['platform'].isin(['PS4', 'PS3', 'X360', '3DS', 'XOne'
])]
```

In [34]:

```
df_top_platforms.boxplot(by='platform', column='total_sales')
plt.suptitle('')
plt.ylim(-5, 20)
plt.title('Differences in sales between platforms')
plt.gcf().set_size_inches(10, 5);
```



All box plots look similar, there are no significant differences in sales. All distributions are positively skewed. Let's calculate the median and average values for each platform.

In [35]:

```
df_top_platforms.pivot_table(index='platform', values='total_sales', aggfunc=['median','mean'])
```

Out[35]:

	median	mean
	total_sales	total_sales
platform		
3DS	0.12	0.498077
PS3	0.27	0.705973
PS4	0.20	0.801378
X360	0.28	0.769746
XOne	0.22	0.645020

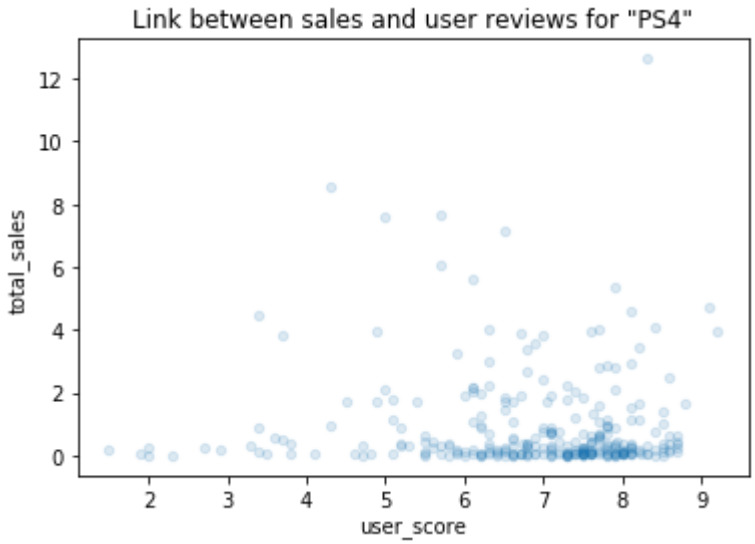
Mean values are much higher than medians which makes sense as the distributions are positively skewed as we noticed earlier.

Now let's take a look at how user and professional reviews affect sales for the most popular platform (according to sales) - 'PS4'.

Correlation between sales and reviews

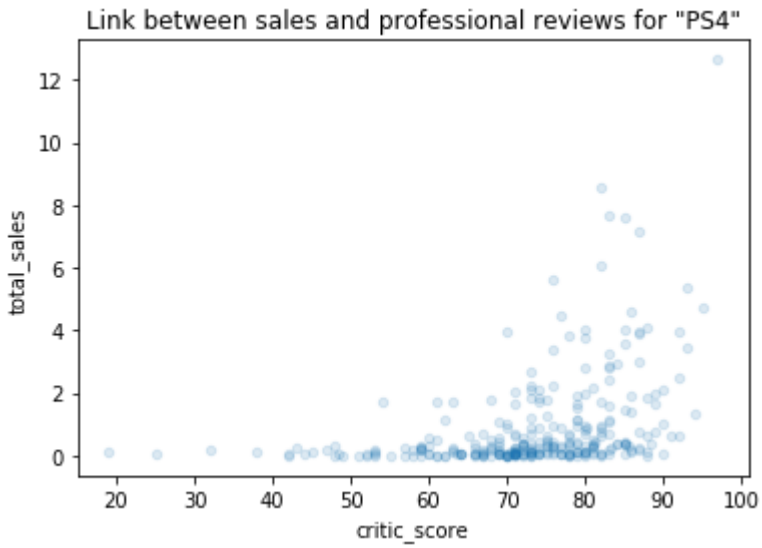
In [36]:

```
ps4 = df_final[df_final['platform']=='PS4']
ps4.plot.scatter(y='total_sales', x='user_score', alpha=.15)
plt.title('Link between sales and user reviews for "PS4"');
```



In [37]:

```
ps4.plot.scatter(y='total_sales', x='critic_score', alpha=.15)
plt.title('Link between sales and professional reviews for "PS4"');
```



In [38]:

```
print('Pearson correlation coefficient between sales and user reviews: {:.0%}'.f
ormat(ps4['total_sales'].corr(ps4['user_score'])))
```

```
Pearson correlation coefficient between sales and user reviews: -4%
```

In [39]:

```
print('Pearson correlation coefficient between sales and professional reviews:
{:.0%}'.format(ps4['total_sales'].corr(ps4['critic_score'])))
```

```
Pearson correlation coefficient between sales and professional revie
ws: 39%
```

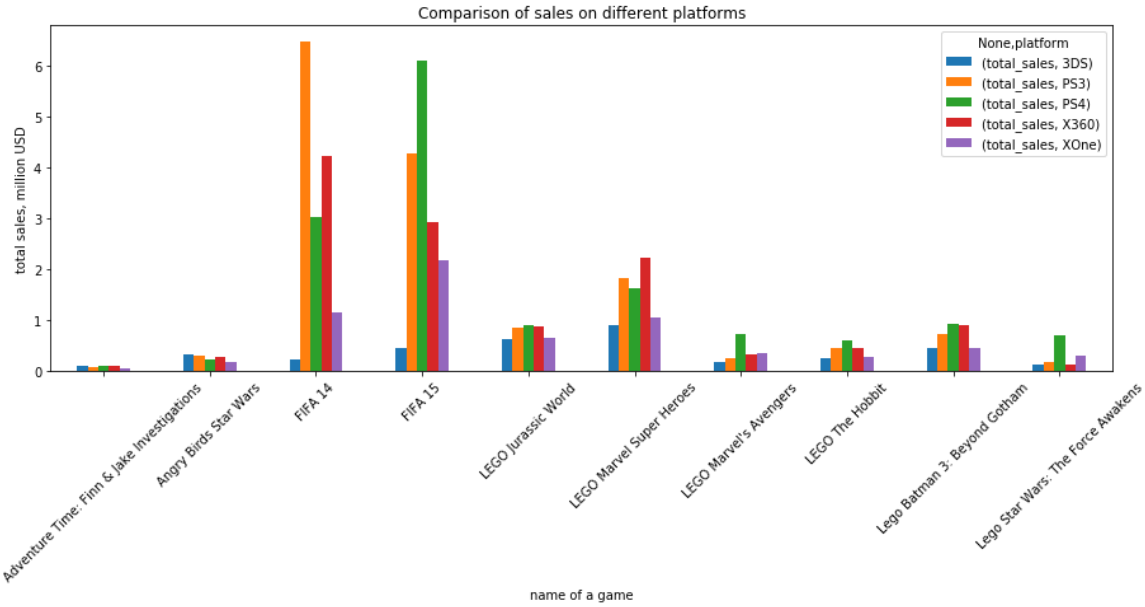
Based on the scatter plots and Pearson coefficients, there is almost no linear connection between sales and user reviews and quite a weak positive linear connection between game sales and professional reviews for the 'PS4' platform.

Games sold on several platforms

Let's compare total sales of the 10 top games sold by the 3 top platforms.

In [40]:

```
(df_top_platforms.pivot_table(index=['name','platform'], values='total_sales', a
ggfunc='sum')
                         .sort_values(by='total_sales', ascending=False).unstack(
1).dropna()[0:10].plot(kind='bar', figsize=(15,5)))
plt.title('Comparison of sales on different platforms')
plt.xlabel('name of a game')
plt.xticks(rotation=45)
plt.ylabel('total sales, million USD');
```

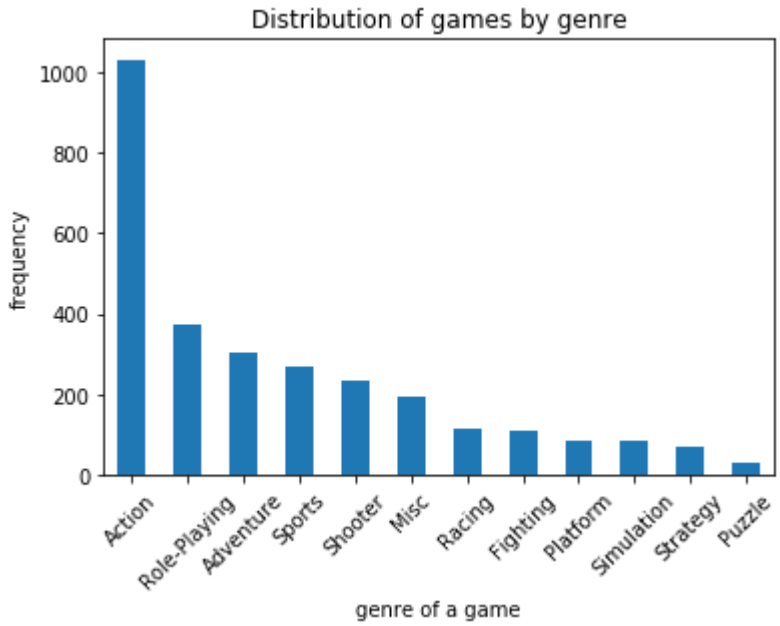


There is no significant difference between the amounts of total sales each platform made for the same games.

Games' genre analysis in terms of sales

In [41]:

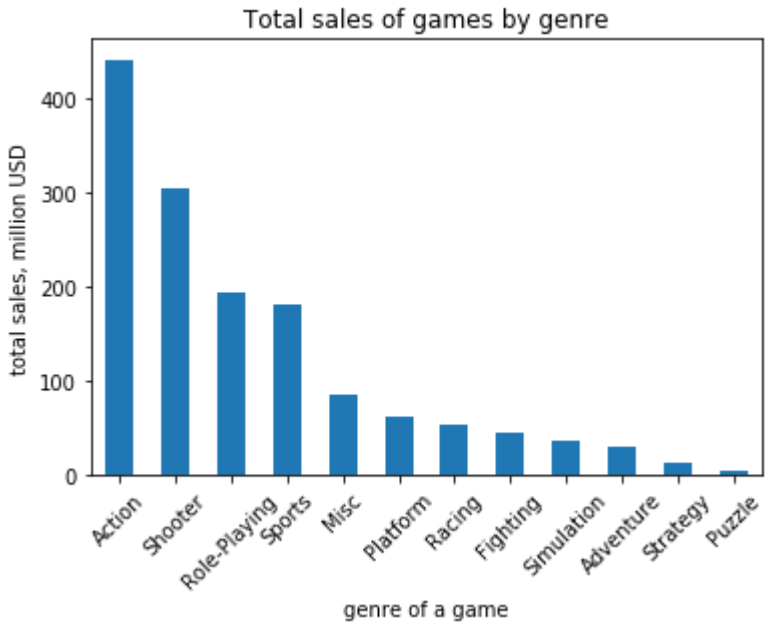
```
df_final.groupby('genre')['genre'].count().sort_values(ascending=False).plot(kin
d='bar')
plt.title('Distribution of games by genre')
plt.xlabel('genre of a game')
plt.xticks(rotation=45)
plt.ylabel('frequency');
```



Action games are by far the most popular in this data set. Now let's see if they are the most profitable as well.

In [104]:

```
df_final.groupby('genre')['total_sales'].sum().sort_values(ascending=False).plot
(kind='bar')
plt.title('Total sales of games by genre')
plt.xlabel('genre of a game')
plt.xticks(rotation=45)
plt.ylabel('total sales, million USD');
```



We can confirm that action games are the most profitable as well as popular. It looks like, generally, more active games (action, shooter, sports) have high sales, while more calm and intellectual games (puzzle, strategy) have low sales.

User profile per region

Top platforms per region

First, let's identify the top 5 platforms in terms of sales in every region.

In [42]:

```
na_top_platforms = df_final.pivot_table(index='platform', values='na_sales', agg
func=['sum']).reset_index()
na_top_platforms.columns = ['platform','total_na_sales']
na_top_platforms['market_share'] = na_top_platforms['total_na_sales']/(na_top_pl
atforms['total_na_sales']).sum()
na_top_platforms = na_top_platforms.sort_values(by='total_na_sales', ascending=F
alse).reset_index(drop=True)

others_sales = na_top_platforms.loc[5:,'total_na_sales'].sum()
others_share = na_top_platforms.loc[5:,'market_share'].sum()

na_top_platforms = na_top_platforms[0:5].append({'platform':'others', 'total_na_
sales':others_sales, 'market_share':others_share}, ignore_index=True)
```

In [43]:

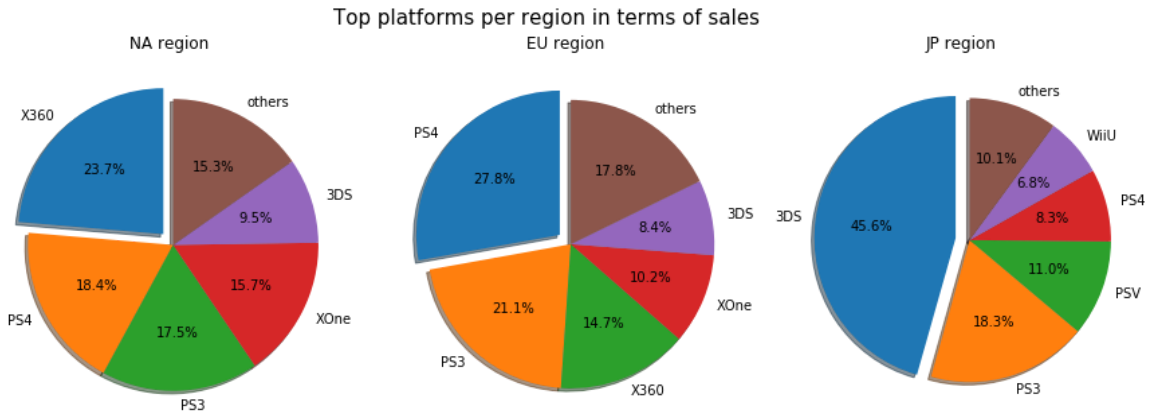
```
eu_top_platforms = df_final.pivot_table(index='platform', values='eu_sales', agg
func=['sum']).reset_index()
eu_top_platforms.columns = ['platform','total_eu_sales']
eu_top_platforms['market_share'] = eu_top_platforms['total_eu_sales']/(eu_top_pl
atforms['total_eu_sales']).sum()
eu_top_platforms = eu_top_platforms.sort_values(by='total_eu_sales', ascending=F
alse).reset_index(drop=True)

others_sales = eu_top_platforms.loc[5:,'total_eu_sales'].sum()
others_share = eu_top_platforms.loc[5:,'market_share'].sum()

eu_top_platforms = eu_top_platforms[0:5].append({'platform':'others', 'total_eu_
sales':others_sales, 'market_share':others_share}, ignore_index=True)
```

In [44]:

```
jp_top_platforms = df_final.pivot_table(index='platform', values='jp_sales', agg
func=['sum']).reset_index()
jp_top_platforms.columns = ['platform','total_jp_sales']
jp_top_platforms['market_share'] = jp_top_platforms['total_jp_sales']/(jp_top_pl
atforms['total_jp_sales']).sum()
jp_top_platforms = jp_top_platforms.sort_values(by='total_jp_sales', ascending=F
alse).reset_index(drop=True)

others_sales = jp_top_platforms.loc[5:,'total_jp_sales'].sum()
others_share = jp_top_platforms.loc[5:,'market_share'].sum()

jp_top_platforms = jp_top_platforms[0:5].append({'platform':'others', 'total_jp_
sales':others_sales, 'market_share':others_share}, ignore_index=True)
```

In [45]:

```
fig, (ax1,ax2,ax3) = plt.subplots(1,3,figsize=(15,5)) #ax1,ax2,ax3 refer to our
 3 pies
# 1,2 denotes 1 row, 2 columns - if you want to stack vertically, it would be 2,
1

fig.suptitle('Top platforms per region in terms of sales', fontsize=15)


labels = na_top_platforms['platform']
sizes = na_top_platforms['market_share']
explode = (0.1, 0, 0, 0, 0, 0)  # only "explode" the 1st slice

ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
ax1.set_title('NA region');


labels = eu_top_platforms['platform']
sizes = eu_top_platforms['market_share']
explode = (0.1, 0, 0, 0, 0, 0)  # only "explode" the 1st slice

ax2.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax2.axis('equal');  # Equal aspect ratio ensures that pie is drawn as a circle.
ax2.set_title('EU region')


labels = jp_top_platforms['platform']
sizes = jp_top_platforms['market_share']
explode = (0.1, 0, 0, 0, 0, 0)  # only "explode" the 1st slice

ax3.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax3.axis('equal');  # Equal aspect ratio ensures that pie is drawn as a circle.
ax3.set_title('JP region');
```



In each region different platforms prevail - most games are sold on 'X360' platform in the NA region, on 'PS4' platform in the EU region and on '3DS' in the JP region.

In the NA region, second place belongs to 'PS4' and third - to 'PS3'. The differences between the 3 platforms are not very big.

In the EU region, second place belongs to 'PS3' and third - to 'X360'. The differences between the 3 platforms are not very big either.

Interestingly, the biggest platform of the NA region is almost not represented at all in Japan. Instead, second place belongs to 'PS3' and third - to 'PSV'.

Most popular platforms fill up a quarter of all sales in NA and EU regions and almost a half in Japan.

Top genres per region

Next, let's look at the top 5 genres in terms of sales in every region.

In [46]:

```
na_top_genres = df_final.pivot_table(index='genre', values='na_sales', aggfunc=[
'sum']).reset_index()
na_top_genres.columns = ['genre','total_na_sales']
na_top_genres['market_share'] = na_top_genres['total_na_sales']/(na_top_genres[
'total_na_sales']).sum()
na_top_genres = na_top_genres.sort_values(by='total_na_sales', ascending=False).
reset_index(drop=True)

others_sales = na_top_genres.loc[5:,'total_na_sales'].sum()
others_share = na_top_genres.loc[5:,'market_share'].sum()

na_top_genres = na_top_genres[0:5].append({'genre':'others', 'total_na_sales':ot
hers_sales, 'market_share':others_share}, ignore_index=True)
```

In [47]:

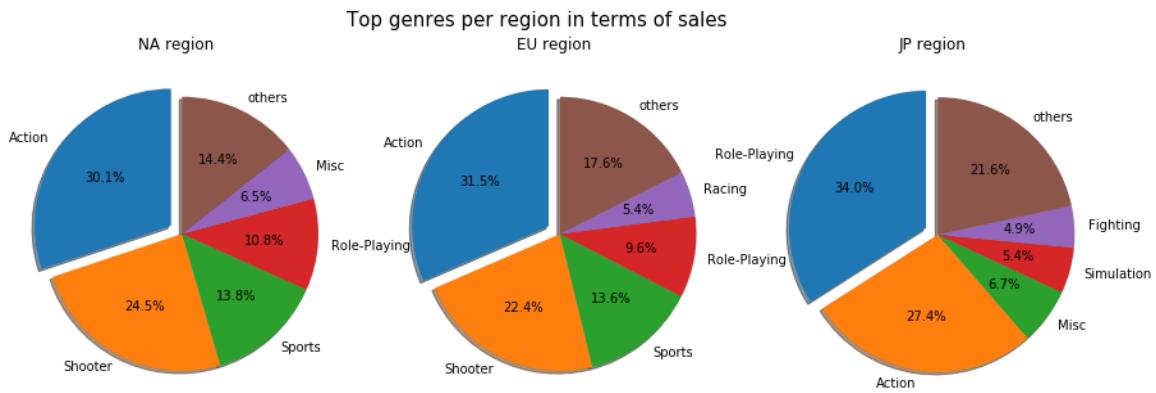
```
eu_top_genres = df_final.pivot_table(index='genre', values='eu_sales', aggfunc=[
'sum']).reset_index()
eu_top_genres.columns = ['genre','total_eu_sales']
eu_top_genres['market_share'] = eu_top_genres['total_eu_sales']/(eu_top_genres[
'total_eu_sales']).sum()
eu_top_genres = eu_top_genres.sort_values(by='total_eu_sales', ascending=False).
reset_index(drop=True)

others_sales = eu_top_genres.loc[5:,'total_eu_sales'].sum()
others_share = eu_top_genres.loc[5:,'market_share'].sum()

eu_top_genres = eu_top_genres[0:5].append({'genre':'others', 'total_eu_sales':ot
hers_sales, 'market_share':others_share}, ignore_index=True)
```

In [48]:

```
jp_top_genres = df_final.pivot_table(index='genre', values='jp_sales', aggfunc=[
'sum']).reset_index()
jp_top_genres.columns = ['genre','total_jp_sales']
jp_top_genres['market_share'] = jp_top_genres['total_jp_sales']/(jp_top_genres[
'total_jp_sales']).sum()
jp_top_genres = jp_top_genres.sort_values(by='total_jp_sales', ascending=False).
reset_index(drop=True)

others_sales = jp_top_genres.loc[5:,'total_jp_sales'].sum()
others_share = jp_top_genres.loc[5:,'market_share'].sum()

jp_top_genres = jp_top_genres[0:5].append({'genre':'others', 'total_jp_sales':ot
hers_sales, 'market_share':others_share}, ignore_index=True)
```

In [49]:

```
fig, (ax1,ax2,ax3) = plt.subplots(1,3,figsize=(15,5)) #ax1,ax2,ax3 refer to our
 3 pies
# 1,2 denotes 1 row, 2 columns - if you want to stack vertically, it would be 2,
1

fig.suptitle('Top genres per region in terms of sales', fontsize=15)


labels = na_top_genres['genre']
sizes = na_top_genres['market_share']
explode = (0.1, 0, 0, 0, 0, 0)  # only "explode" the 1st slice

ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
ax1.set_title('NA region');


labels = eu_top_genres['genre']
sizes = eu_top_genres['market_share']
explode = (0.1, 0, 0, 0, 0, 0)  # only "explode" the 1st slice

ax2.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax2.axis('equal');  # Equal aspect ratio ensures that pie is drawn as a circle.
ax2.set_title('EU region')


labels = jp_top_genres['genre']
sizes = jp_top_genres['market_share']
explode = (0.1, 0, 0, 0, 0, 0)  # only "explode" the 1st slice

ax3.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax3.axis('equal');  # Equal aspect ratio ensures that pie is drawn as a circle.
ax3.set_title('JP region');
```



Action games are the most popular in both the NA and EU regions, while role-playing games prevail in Japan (with action games on the second place).

The NA and EU regions are similar: shooter games come second in both regions, the difference in percentage is not very big - around 7-8%. Third are sport games. Let's see which platform has the most sports games.

In [50]:

```
df_final[df_final['genre']=='Sports'].groupby('platform')['genre'].count().sort_values(ascending=False).head(3)
```

Out[50]:

```
platform
PS3     63
PS4     46
X360    44
Name: genre, dtype: int64
```

'PS3' has the most sports games and it explains why it occupies third and second places in the NA and EU regions, respectively.

The JP region differs: more than 30% of games are role-playing. Let's check which platform has the most role-playing games.

In [51]:

```
df_final[df_final['genre']=='Role-Playing'].groupby('platform')['genre'].count().sort_values(ascending=False).head(3)
```

Out[51]:

```
platform
PSV    85
3DS    80
PS3    61
Name: genre, dtype: int64
```

As we expected, 'PSV' and '3DS' platforms release the most role-playing games, that's why they are the biggest in Japan in terms of sales.

The rest of genres that we summarized under "others" takes up a little less than a quarter of all sales, so we can conclude that different players prefer different kinds of games, so the variety of genres is appreciated by users.

ESRB ratings per region

Finally, let's analyze the ESRB ratings in terms of sales in every region.

In [52]:

```
na_ratings = df_final.pivot_table(index='rating', values='na_sales', aggfunc=['s
um']).reset_index()
na_ratings.columns = ['rating','total_na_sales']
na_ratings['market_share'] = na_ratings['total_na_sales']/(na_ratings['total_na_
sales']).sum()
na_ratings = na_ratings.sort_values(by='total_na_sales', ascending=False).reset_
index(drop=True)

others_sales = na_ratings.loc[4:,'total_na_sales'].sum()
others_share = na_ratings.loc[4:,'market_share'].sum()

na_ratings = na_ratings[0:4].append({'rating':'others', 'total_na_sales':others_
sales, 'market_share':others_share}, ignore_index=True)
```

In [53]:

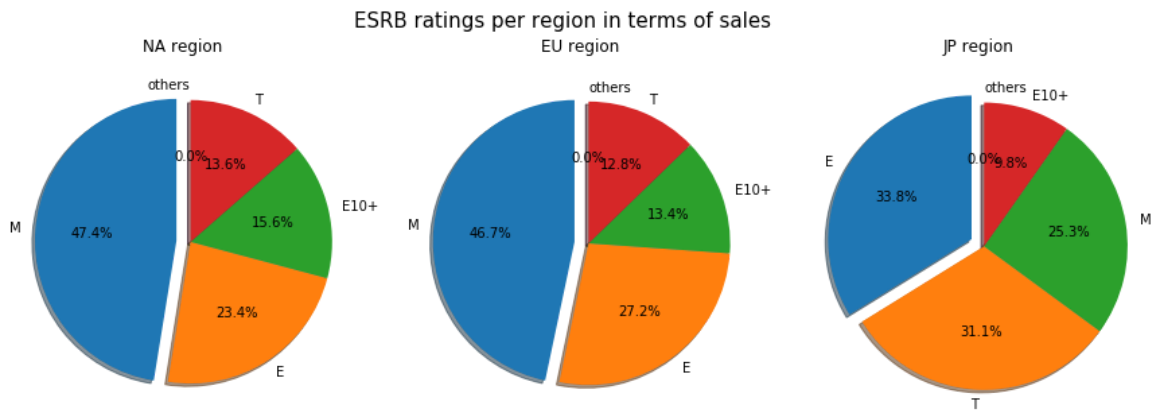
```
eu_ratings = df_final.pivot_table(index='rating', values='eu_sales', aggfunc=['s
um']).reset_index()
eu_ratings.columns = ['rating','total_eu_sales']
eu_ratings['market_share'] = eu_ratings['total_eu_sales']/(eu_ratings['total_eu_
sales']).sum()
eu_ratings = eu_ratings.sort_values(by='total_eu_sales', ascending=False).reset_
index(drop=True)

others_sales = eu_ratings.loc[4:,'total_eu_sales'].sum()
others_share = eu_ratings.loc[4:,'market_share'].sum()

eu_ratings = eu_ratings[0:4].append({'rating':'others', 'total_eu_sales':others_
sales, 'market_share':others_share}, ignore_index=True)
```

In [54]:

```
jp_ratings = df_final.pivot_table(index='rating', values='jp_sales', aggfunc=['s
um']).reset_index()
jp_ratings.columns = ['rating','total_jp_sales']
jp_ratings['market_share'] = jp_ratings['total_jp_sales']/(jp_ratings['total_jp_
sales']).sum()
jp_ratings = jp_ratings.sort_values(by='total_jp_sales', ascending=False).reset_
index(drop=True)

others_sales = jp_ratings.loc[4:,'total_jp_sales'].sum()
others_share = jp_ratings.loc[4:,'market_share'].sum()

jp_ratings = jp_ratings[0:4].append({'rating':'others', 'total_jp_sales':others_
sales, 'market_share':others_share}, ignore_index=True)
```

In [55]:

```
fig, (ax1,ax2,ax3) = plt.subplots(1,3,figsize=(15,5)) #ax1,ax2,ax3 refer to our
 3 pies
# 1,2 denotes 1 row, 2 columns - if you want to stack vertically, it would be 2,
1

fig.suptitle('ESRB ratings per region in terms of sales', fontsize=15)


labels = na_ratings['rating']
sizes = na_ratings['market_share']
explode = (0.1, 0, 0, 0, 0)  # only "explode" the 1st slice

ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
ax1.set_title('NA region');


labels = eu_ratings['rating']
sizes = eu_ratings['market_share']
explode = (0.1, 0, 0, 0, 0)  # only "explode" the 1st slice

ax2.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax2.axis('equal');  # Equal aspect ratio ensures that pie is drawn as a circle.
ax2.set_title('EU region')


labels = jp_ratings['rating']
sizes = jp_ratings['market_share']
explode = (0.1, 0, 0, 0, 0)  # only "explode" the 1st slice

ax3.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax3.axis('equal');  # Equal aspect ratio ensures that pie is drawn as a circle.
ax3.set_title('JP region');
```



As for the ESRB rating, the tendencies in the NA and EU regions are the same, while Japan shows again a different picture.

The NA and EU:

Games with rating "M" (Mature 18+) are sold the most - almost a half of all market -, second are games with rating "E" (Everyone) (a little under 30% of all sales), third are games with rating "E10+" (Everyone 10+) - around 15% of all sales, and forth are games with rating "T" (Teens 17+) - also around 15% of all sales. The rest of ratings occupy less than 1% of all sales (EC - probably it was abandoned as there is no description on the official website and RP - ratings pending).

Japan:

Games with rating "E" (Everyone) are sold the most - a little more than 30% of all-, second are games with rating "T" (teens 17+) - also around 30% of all sales, third are games with rating "M" (Mature 18+) - around 25% of all sales, and forth are games with rating "E10+" - 10% of all sales. The rest of ratings occupy less than 1% of all sales (EC - probably it was abandoned as there is no description on the official website and RP - ratings pending).

This gives us an idea of the age distribution of the target audience in the gaming industry per region.

Summarizing this section, a typical user in each region has the following characteristics:

- NA region: a Mature person (18+) playing mostly action games on the "X360" platform;
- EU region: a Mature person (18+) playing mostly action games on the "PS4" platform;
- JP region: a person of any age playing mostly role-playing games on the "3DS" platform.

Statistical hypotheses testing

Average user ratings of the Xbox One and PC platforms are the same

Step 1: the null and alternative hypotheses

H0: The means of two statistical populations are equal. In our case it means that the average user score of the "XOne" and "PC" platforms is the same.

H1: The means of two statistical populations are not equal. In our case it means that the average user score of the "XOne" and "PC" platforms differs, although we do not specify here which one is more.

Step 2: Set the criteria for a decision

In behavioral science, the level of significance is typically set at 5% and we will choose this criteria as well. When the probability of obtaining a sample mean is less than 5% if the null hypothesis were true, then we reject the value stated in the null hypothesis.

Step 3: Compute the test statistic

In order to test our hypothesis that the means of two statistical populations are equal based on samples taken from them, we will apply the method `scipy.stats.ttest_ind()`.

The method takes the following parameters:

- `array1`, `array2` are arrays containing the samples. We will use the `monthly_profit` variables we calculated earlier for both plans;
- `equal_var` is an optional parameter that specifies whether or not the variances of the populations should be considered equal. To set this parameter let's test whether the variances of our samples are the same.

In [56]:

```
sample_1 = df_final[df_final['platform'] == 'XOne'].dropna()['user_score']
sample_2 = df_final[df_final['platform'] == 'PC'].dropna()['user_score']
```

In [57]:

```
st.levene(sample_1, sample_2)
```

Out[57]:

```
LeveneResult(statistic=10.413474411034404, pvalue=0.0013628728621040
526)
```

The p-value is very low (much less than 1%) suggests that the populations do not have equal variances. Hence we will set the `equal_var` parameter to False.

In [58]:

```
alpha = .05 # critical statistical significance level
                        # if the p-value is less than alpha, we reject the hypot
hesis

results = st.ttest_ind(
                        sample_1,
                        sample_2,
                        equal_var=False)

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

if (results.pvalue < alpha):
        print("We reject the null hypothesis")
else:
        print("We retain the null hypothesis")
```

```
p-value:  0.17787582723979845
We retain the null hypothesis
```

Step 4: Make a decision

Based on the results of the test statistic we **failed to reach significance**: the decision is to **retain the null hypothesis**. The equality of samples' means is associated with a high probability of occurrence (more than 5%) when the null hypothesis is true. It means that the average user ratings of the Xbox One and PC platforms are indeed the same.

Average user ratings for the Action and Sports genres are the same

Step 1: the null and alternative hypotheses

H0: The means of two statistical populations are equal. In our case it means that the average user score of the "Action" games and "Sports" games is the same.

H1: The means of two statistical populations are not equal. In our case it means that the average user score of the "Action" games and "Sports" games differs, although we do not specify here which one is more.

Step 2: Set the criteria for a decision

In behavioral science, the level of significance is typically set at 5% and we will choose this criteria as well. When the probability of obtaining a sample mean is less than 5% if the null hypothesis were true, then we reject the value stated in the null hypothesis.

Step 3: Compute the test statistic

In order to test our hypothesis that the means of two statistical populations are equal based on samples taken from them, we will apply the method `scipy.stats.ttest_ind()`.

The method takes the following parameters:

- `array1`, `array2` are arrays containing the samples. We will use the `monthly_profit` variables we calculated earlier for both plans;
- `equal_var` is an optional parameter that specifies whether or not the variances of the populations should be considered equal. To set this parameter let's test whether the variances of our samples are the same.

In [59]:

```
sample_3 = df_final[df_final['genre'] == 'Action'].dropna()['user_score']
sample_4 = df_final[df_final['genre'] == 'Sports'].dropna()['user_score']
```

In [60]:

```
st.levene(sample_3, sample_4)
```

Out[60]:

```
LeveneResult(statistic=9.406709133497195, pvalue=0.00223829355636501
97)
```

The p-value is very low (much less than 1%) suggests that the populations do not have equal variances. Hence we will set the `equal_var` parameter to False.

In [61]:

```
alpha = .05 # critical statistical significance level
                        # if the p-value is less than alpha, we reject the hypot
hesis

results = st.ttest_ind(
                        sample_3,
                        sample_4,
                        equal_var=False)

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

if (results.pvalue < alpha):
        print("We reject the null hypothesis")
else:
        print("We retain the null hypothesis")
```

```
p-value:  2.0282047387359358e-09
We reject the null hypothesis
```

Step 4: Make a decision

Based on the results of the test statistic we reach **significance**: the decision is to **reject the null hypothesis**. The equality of samples' means is associated with a low probability of occurrence (less than 5%) when the null hypothesis is true. It means that the average user ratings of the Action and Sports games are not the same.

Conclusion

In this report we have **analyzed video games players' behavior in order to identify patterns that determine whether a game succeeds or not and to test two statistical hypotheses**.

First of all, we have familiarized ourselves with the data by performing the descriptive statistics. Based on that analysis, in the preprocessing step we converted all column names to lower case letters, made sure all variables had appropriate data types, checked for duplicates and filled a part of missing values. For the rest of missing values, we've decided to keep them as they are because we didn't find any appropriate way to fill them all and we don't want to distort our data by false assumptions. Finally, we have calculated total sales for all regions.

In the following section we have performed an exploratory data analysis and reached the following conclusions:

- Most games in this data set were released around 2009-2012, although the number of games released before the 21st century is not very significant;
- There are 5 platforms that stand out in terms of total sales in all regions: 'PS4', 'PS3', 'X360', '3DS', 'XOne';
- On average it takes around 10 years for new platforms to appear and for the old ones to fade away, so we decided to take data for the past 5 years to build a prognosis for 2017 in order to make sure the chosen platforms are still popular in 2017;
- There are no significant differences in sales for the top 5 platforms. All 5 distributions have long tails of large positive numbers;
- Based on the scatter plots and Pearson coefficients, there is almost no linear connection between sales and user reviews and quite a weak linear connection between game sales and professional reviews for the 'PS4' platform.
- We didn't find any significant differences between the amounts of total sales each platform made for the same games.
- Action games are the most profitable as well as popular. It looks like, generally, more active games (action, shooter, sports) have high sales, while more calm and intellectual games (puzzle, strategy) have low sales.

Next, we analyzed top 5 platforms and genres in each region and then compared them.

We found that in each region different platforms prevail - most games are sold on 'X360' platform in the NA region, on 'PS4' platform in the EU region and on '3DS' in the JP region.

In the NA region, second place belongs to 'PS4' and third - to 'PS3'. In the EU region, second place belongs to 'PS3' and third - to 'X360'. The differences between the 3 platforms are not very big in both regions. Interestingly, the biggest platform of the NA region is almost not represented at all in Japan. Instead, second place belongs to 'PS3' and third - to 'PSV'. Most popular platforms fill up a quarter of all sales in NA and EU regions and almost a half in Japan.

Action games are the most popular in both the NA and EU regions, while role-playing games prevail in Japan (with action games on the second place).

The NA and EU regions are similar: shooter games come second in both regions, the difference in percentage is not very big - around 7-8%. Third are sport games. 'PS3' released the most sports games in the last 5 years and it explains why it occupies the third and second places in the NA and EU regions, respectively.

The JP region differs: more than 30% of games are role-playing. 'PS3' and '3DS' platforms release the most role-playing games, that's why they are the biggest in Japan in terms of sales.

The rest of genres that we summarized under "others" takes up a little less than a quarter of all sales, so we can conclude that different players prefer different kinds of games, so the variety of genres is appreciated by users.

We also checked whether the ESRB ratings affect sales in individual regions and concluded that the tendencies in the NA and EU regions are the same, while Japan shows again a different picture.

The NA and EU:

Games with rating "M" (Mature 18+) are sold the most - almost a half of all market -, second are games with rating "E" (Everyone) (a little under 30% of all sales), third are games with rating "E10+" (Everyone 10+) - around 15% of all sales, and forth are games with rating "T" (Teens 17+) - also around 15% of all sales. The rest of ratings occupy less than 1% of all sales (EC - probably it was abandoned as there is no description on the official website and RP - ratings pending).

Japan:

Games with rating "E" (Everyone) are sold the most - a little more than 30% of all-, second are games with rating "T" (teens 17+) - also around 30% of all sales, third are games with rating "M" (Mature 18+) - around 25% of all sales, and forth are games with rating "E10+" - 10% of all sales. The rest of ratings occupy less than 1% of all sales (EC - probably it was abandoned as there is no description on the official website and RP - ratings pending).

This gives us an idea of the age distribution of the target audience in the gaming industry per region.

Summarizing this section, a typical user in each region has the following characteristics:

- NA region: a Mature person (18+) playing mostly action games on the "X360" platform;
- EU region: a Mature person (18+) playing mostly action games on the "PS4" platform;
- JP region: a person of any age playing mostly role-playing games on the "3DS" platform.

Next step was statistical hypotheses testing. We tested two hypotheses:

1. Average user ratings of the 'Xbox One' and 'PC' platforms are the same.
2. Average user ratings for the Action and Sports genres are the same.

In the first case we have **retained the null** - the average user ratings of the 'Xbox One' and 'PC' platforms are indeed the same.

In the second case we have **rejected the null** - the average user ratings of the Action and Sports games are not the same.