Table of contents

Our task is to analyse and identify the patterns that determine whether a game succeeds or not. This will allow us to spot potential big winners and plan advertising campaigns.

First of all we'll import all of our libraries and start to inspect the data.

- Please note: there is a lot of installs that are necessary for some features to work, please give the notebook a couple of minutes to load properly.

In [5]:

```
#!pip install --upgrade pip
#!pip install squarify
#!pip install streamlit
#!pip install --upgrade plotly
#!pip install plotly.express
#!pip install --upgrade ipython
```

In [2]:

```
%matplotlib inline
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from scipy import stats as st
import re
from IPython.display import display, HTML
from IPython.display import display_html
import matplotlib as mpl
import squarify
import plotly.express as px
import warnings
from pylab import rcParams


warnings.filterwarnings("ignore")
df = pd.read_csv('./games.csv')
```

In [3]:

```
%matplotlib inline
warnings.filterwarnings("ignore")
rcParams['figure.figsize'] = 10,5
rcParams['font.size'] = 30
sns.set()
np.random.seed(8)

def plot_distribution(inp):
    warnings.filterwarnings("ignore")
    plt.figure()
    ax = sns.distplot(inp)
    plt.axvline(np.mean(inp), color="k", linestyle="dashed", linewidth=5)
    _, max_ = plt.ylim()
    plt.text(
        inp.mean() + inp.mean() / 10,
        max_ - max_ / 10,
        "Mean: {:.2f}".format(inp.mean()),
    )
    return plt.figure;
```

In [4]:

```
def display_side_by_side(dfs:list, captions:list):
    output = ""
    combined = dict(zip(captions, dfs))
    for caption, df in combined.items():
        output += df.style.set_table_attributes("style='display:inline'").set_caption(capti
        output += "\xa0\xa0\xa0"
    display(HTML(output))
```

In [5]:

```
df.shape
```

Out[5]:

```
(16715, 11)
```

In [6]:

```
df.head(15)
```

Out[6]:

	Name	Platform	Year_of_Release	Genre	NA_sales	EU_sales	JP_sales	Other_
0	Wii Sports	Wii	2006.0	Sports	41.36	28.96	3.77	
1	Super Mario Bros.	NES	1985.0	Platform	29.08	3.58	6.81	
2	Mario Kart Wii	Wii	2008.0	Racing	15.68	12.76	3.79	
3	Wii Sports Resort	Wii	2009.0	Sports	15.61	10.93	3.28	
4	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	11.27	8.89	10.22	
5	Tetris	GB	1989.0	Puzzle	23.20	2.26	4.22	
6	New Super Mario Bros.	DS	2006.0	Platform	11.28	9.14	6.50	
7	Wii Play	Wii	2006.0	Misc	13.96	9.18	2.93	
8	New Super Mario Bros. Wii	Wii	2009.0	Platform	14.44	6.94	4.70	
9	Duck Hunt	NES	1984.0	Shooter	26.93	0.63	0.28	
10	Nintendogs	DS	2005.0	Simulation	9.05	10.95	1.93	
11	Mario Kart DS	DS	2005.0	Racing	9.71	7.47	4.13	
12	Pokemon Gold/Pokemon Silver	GB	1999.0	Role-Playing	9.00	6.18	7.20	
13	Wii Fit	Wii	2007.0	Sports	8.92	8.03	3.60	
14	Kinect Adventures!	X360	2010.0	Misc	15.00	4.89	0.24	

In [7]:

```
df.describe()
```

Out[7]:

	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

In [8]:

```
display(df.dtypes)
```

```
Name              object
Platform          object
Year_of_Release  float64
Genre             object
NA_sales         float64
EU_sales         float64
JP_sales         float64
Other_sales      float64
Critic_Score     float64
User_Score        object
Rating            object
dtype: object
```

In [9]:

```
display("duplicates", df.duplicated().sum())
```

```
'duplicates'

0
```

In [10]:

```
for (columnName, columnData) in df.iteritems():
    if columnData.isna().sum() != 0:
        print("{} has {} NaNs which is {:,.0f}% of the data".format(columnName, columnData.


df[df.columns[df.isnull().any()]].isnull().sum() * 100 / df.shape[0]
```

```
Name has 2 NaNs which is 0% of the data
Year_of_Release has 269 NaNs which is 2% of the data
Genre has 2 NaNs which is 0% of the data
Critic_Score has 8578 NaNs which is 51% of the data
User_Score has 6701 NaNs which is 40% of the data
Rating has 6766 NaNs which is 40% of the data
```

Out[10]:

```
Name                 0.011965
Year_of_Release      1.609333
Genre                0.011965
Critic_Score        51.319174
User_Score          40.089740
Rating              40.478612
dtype: float64
```

After a quick inspection, we have no duplicates at the start.
next we change the column names to lower case for convenience.
We'll remove the 'name', 'year_of_release' and 'genre' Na' columns as theres no way to find their true value and their entry count is 2% or less. We will change some of the columns float type to int type and to datetime.

In [11]:

```
#Renaming Columns:
df.columns = df.columns.str.lower()
#Lowercase everything:
df['name'] = df['name'].str.lower()
df['platform'] = df['platform'].str.lower()
df['genre'] = df['genre'].str.lower()
df['rating'] = df['rating'].str.lower()
```

In [12]:

```
#Removing Na's:
df.dropna(subset=['name', 'year_of_release', 'genre'], inplace=True)
```

In [13]:

```
#Partial Type conversion:
df['year_of_release'] = df['year_of_release'].astype(int)
df['critic_score'] = pd.to_numeric(df['critic_score'], errors="ignore")
df['user_score'] = pd.to_numeric(df['user_score'], errors="coerce")
```

We will make a copy of the original dataframe so we could compare it with the original after processing

Next we decide with what values to fill in Na's.

In [14]:

```
#New Df:
df2 = df.copy(deep=True)
```

In [15]:

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

Out[15]:

```
count    7463.000000
mean        7.126330
std         1.499447
min         0.000000
25%         6.400000
50%         7.500000
75%         8.200000
max         9.700000
Name: user_score, dtype: float64
```

In [16]:

```
display('Mean', df['user_score'].mean(),'Mode',df['user_score'].mode()[0],'Median',df['user
```

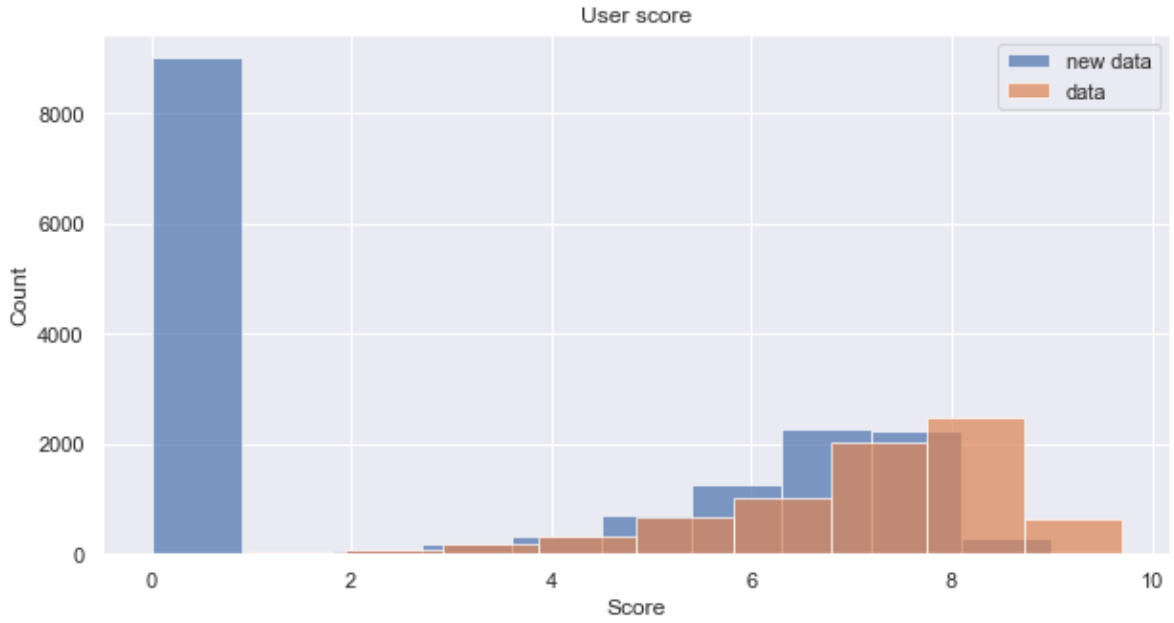
```
'Mean'

7.1263298941444955

'Mode'

7.8

'Median'

7.5
```

In [17]:

```
df2['user_score'].fillna(0, inplace=True)
df2['user_score'] = df2['user_score'].astype(int)
```

In [18]:

```
df2['user_score'].hist(label='new data',alpha = 0.7)
df['user_score'].hist(label='data', alpha = 0.7)
plt.title('User score')
plt.ylabel('Count')
plt.xlabel('Score')
plt.legend();
```



In [19]:

```
df['critic_score'].describe()
```

Out[19]:

```
count    7983.000000
mean       68.994363
std        13.920060
min        13.000000
25%        60.000000
50%        71.000000
75%        79.000000
max        98.000000
Name: critic_score, dtype: float64
```

In [20]:

```
display('Mean', df['critic_score'].mean(),'Mode',df['critic_score'].mode()[0],'Median',df['
```

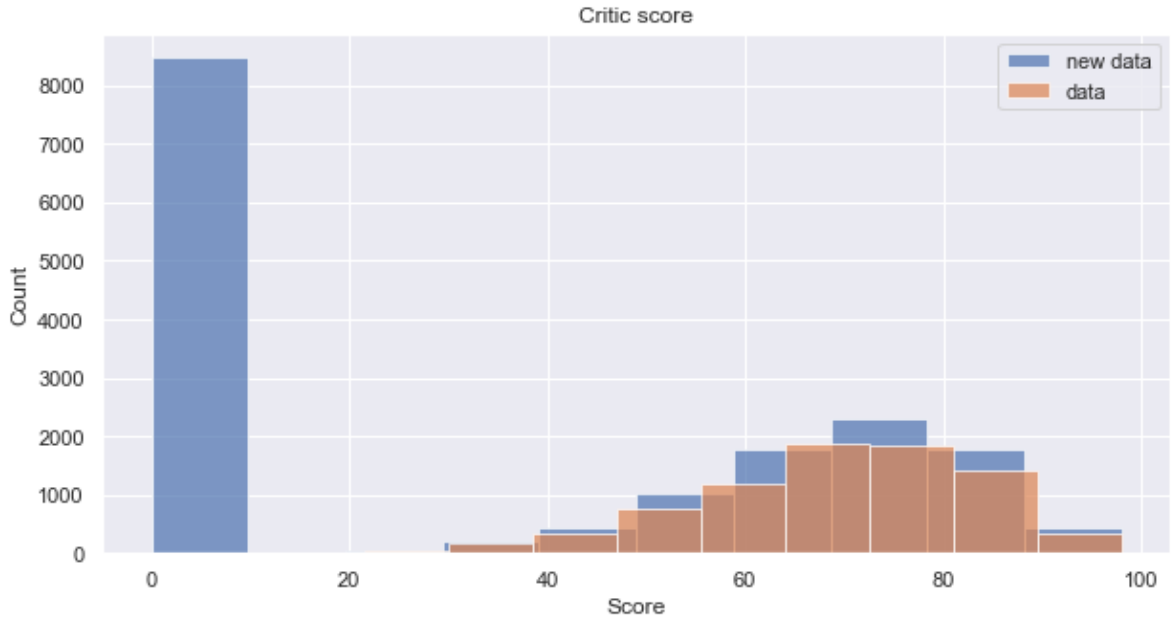
```
'Mean'

68.99436302142053

'Mode'

70.0

'Median'

71.0
```

In [21]:

```
df2['critic_score'].fillna('0',inplace=True)
df2['critic_score'] = df2['critic_score'].astype(int)
```

In [22]:

```
df2['critic_score'].hist(label='new data',alpha = 0.7)
df['critic_score'].hist(label='data',alpha = 0.7)
plt.title('Critic score')
plt.ylabel('Count')
plt.xlabel('Score')
plt.legend();
```



In [23]:

```
df['rating'].describe()
```

Out[23]:

```
count     9768
unique       8
top          e
freq      3921
Name: rating, dtype: object
```

In [24]:

```
df2['rating'].fillna('unknown', inplace = True)
```

In [25]:

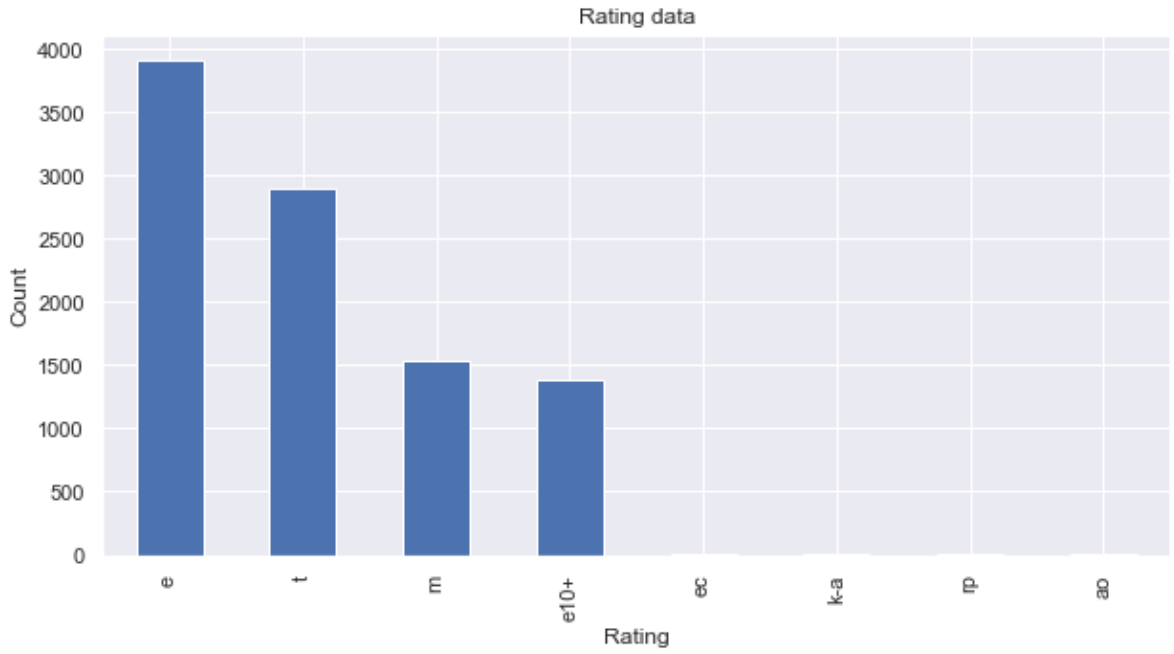
```
df2['rating'].isna().sum()
```

Out[25]:

```
0
```

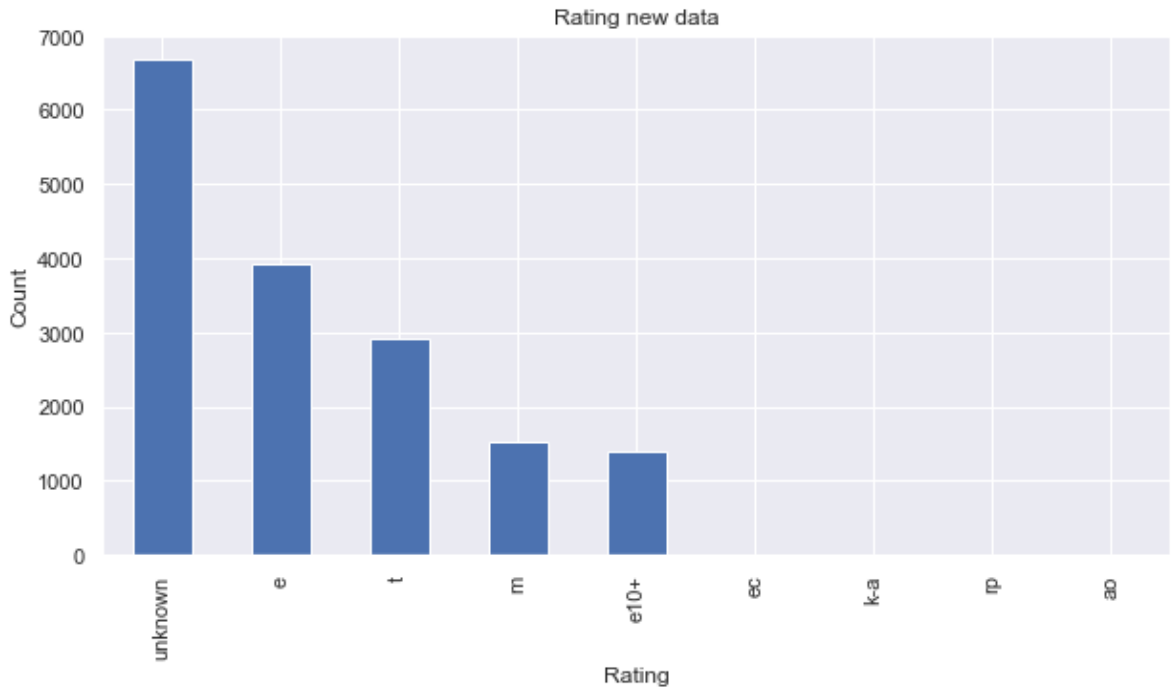
In [26]:

```
df['rating'].value_counts().plot(kind='bar')
plt.title('Rating data')
plt.ylabel('Count')
plt.xlabel('Rating');
```



In [27]:

```
df2['rating'].value_counts().plot(kind='bar')
plt.title('Rating new data')
plt.ylabel('Count')
plt.xlabel('Rating');
```



In [28]:

```
def profit_sum(row):
    return row['na_sales'] + row['eu_sales'] + row['jp_sales'] + row['other_sales']

df2['total_sales'] = df2.apply(profit_sum, axis=1)
```

In [29]:

```
df2.head()
```

Out[29]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	c
0	wii sports	wii	2006	sports	41.36	28.96	3.77	8.45	
1	super mario bros.	nes	1985	platform	29.08	3.58	6.81	0.77	
2	mario kart wii	wii	2008	racing	15.68	12.76	3.79	3.29	
3	wii sports resort	wii	2009	sports	15.61	10.93	3.28	2.95	
4	pokemon red/pokemon blue	gb	1996	role-playing	11.27	8.89	10.22	1.00	

In [30]:

```
df2.dtypes
```

Out[30]:

```
name                object
platform            object
year_of_release      int32
genre               object
na_sales           float64
eu_sales           float64
jp_sales           float64
other_sales        float64
critic_score         int32
user_score           int32
rating              object
total_sales        float64
dtype: object
```

Summary

The data is clean, i couldnt find anything to groupby with so i used the ffill method to fill in the missing data, it seems that way the data would stay normalized and the only thing that changes is the quantity of each parameter. The other option was to fill all missing data with '0' or 'none' as it seems impossible to find which value is supposed to be present.
We kept a clean dataframe before the processing for future reference, all the data types were changed to the appropriate types.
There are no duplicates found. Added the total sales column for each game.

Data Analysis

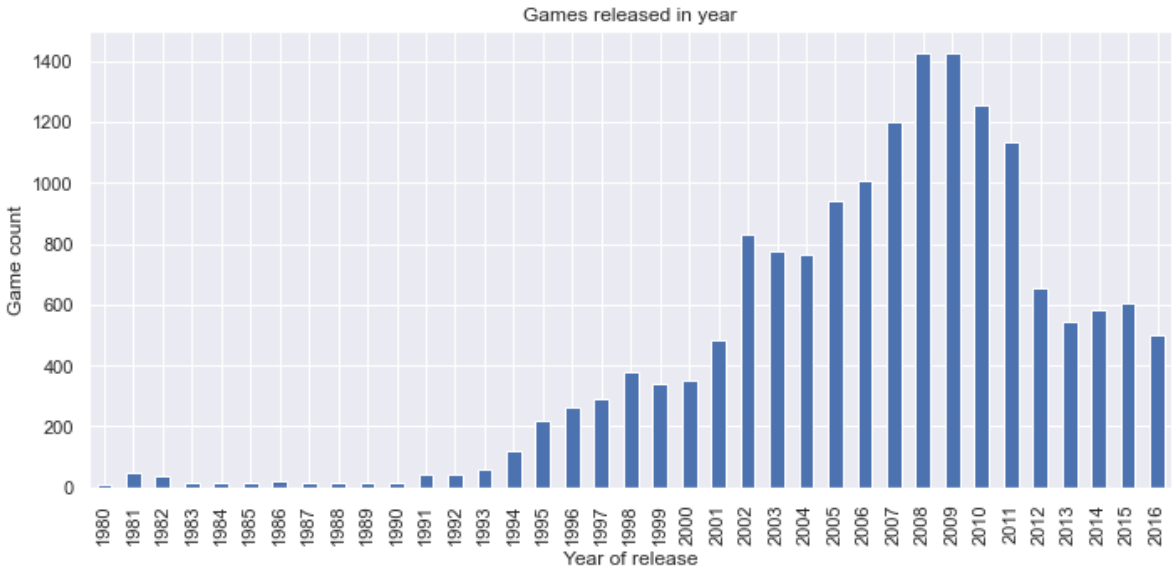
We'll explore the data and take a deeper look at some characteristics it has to offer.
we will start with number of games released each year

In [31]:

```
dff = df2.groupby('year_of_release')['name'].count().reset_index()
```

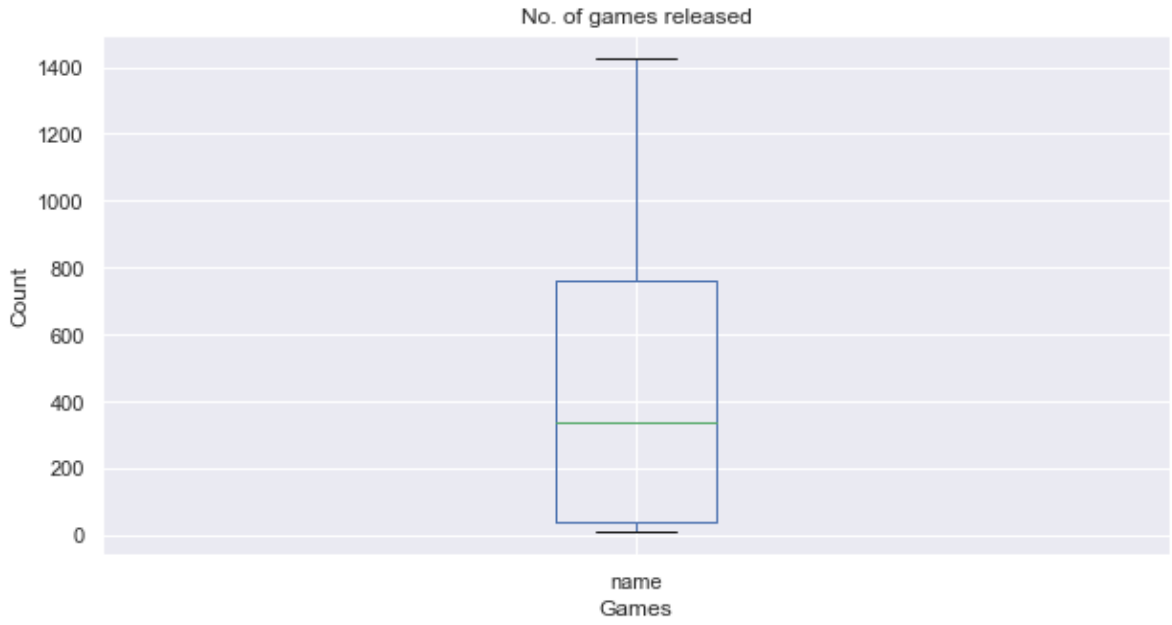
In [32]:

```
#this is the first chart i did:
df2.groupby('year_of_release')['name'].count().plot(kind='bar', title='Games released in ye
plt.xlabel('Year of release')
plt.ylabel('Game count')
plt.tight_layout()
```



In [33]:

```
df2.groupby('year_of_release')['name'].count().plot(kind='box', grid=True, title='No. of ga
plt.xlabel('Games')
plt.ylabel('Count');
```

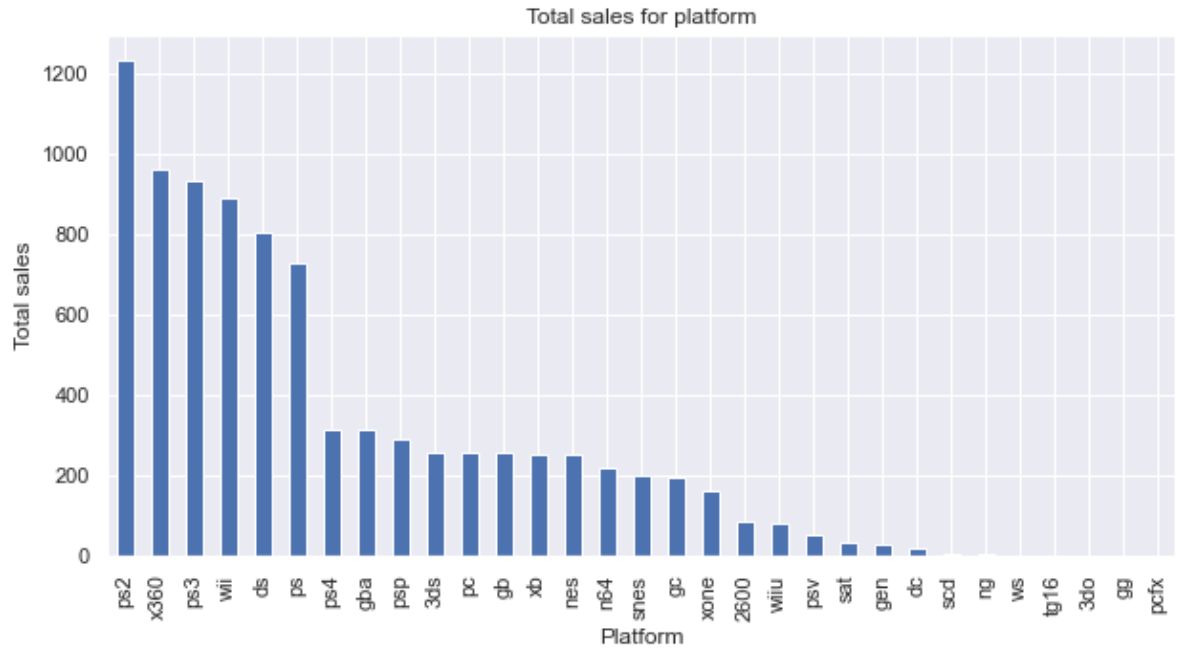


Since 1994 the number of games became higher then before and as the years went by the numbers grew even higher(obviously due to technology evolving and games being more popular). the peak was around 2008 - 2009 with the largest amount, near 1400 games. By that data we can see that data prior to 1994 is not significant as the game count at those years didnt reach the count of 50.

Next we'll compare between the platforms and sees which ones had the most sales.

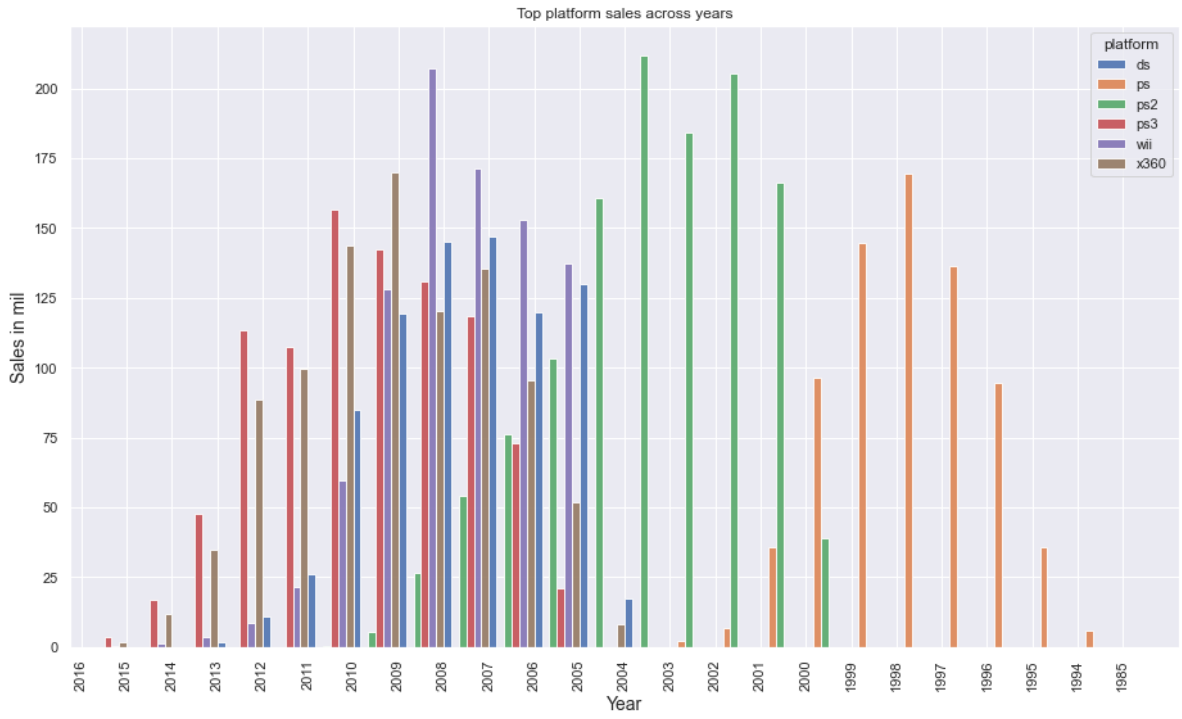
In [34]:

```
df2.groupby('platform')['total_sales'].sum().sort_values(ascending=False).plot(kind='bar',
plt.xlabel('Platform')
plt.ylabel('Total sales');
```



In [35]:

```
plat = df2.query('platform == "ps2" or platform == "x360" or platform == "ps3" or platform
grp_plat = plat.pivot_table(index= 'year_of_release',columns='platform',values='total_sales
plt.ylabel('Sales in mil',size=14)
plt.xlabel('Year',size=14)
plt.tight_layout()
plt.show()
```



We found out the 6 top selling platforms that had the biggest sales overall(by a margin!), and by the bar chart above we can see that the lifespan of a console is about 10 years, from that we can take the PS series as an example that they overlap for a short period as a newer version comes out it takes 3 - 4 years for a complete transition, which is the platforms relevance period, if we make a educated guess this happens due to technology improving.
It also seems that like with the PS example, Ps2 was released after the Ps1 was about 7 years old and then you have about 3 years for the switch to occur. On a side note this happens even now in real life (As Ps4 was released at November 15,2013 and after 7 years the Ps5 was released this month (November 2020). My estimate is that in about 3 years or so we wont see any Ps4 sales anymore. 😉

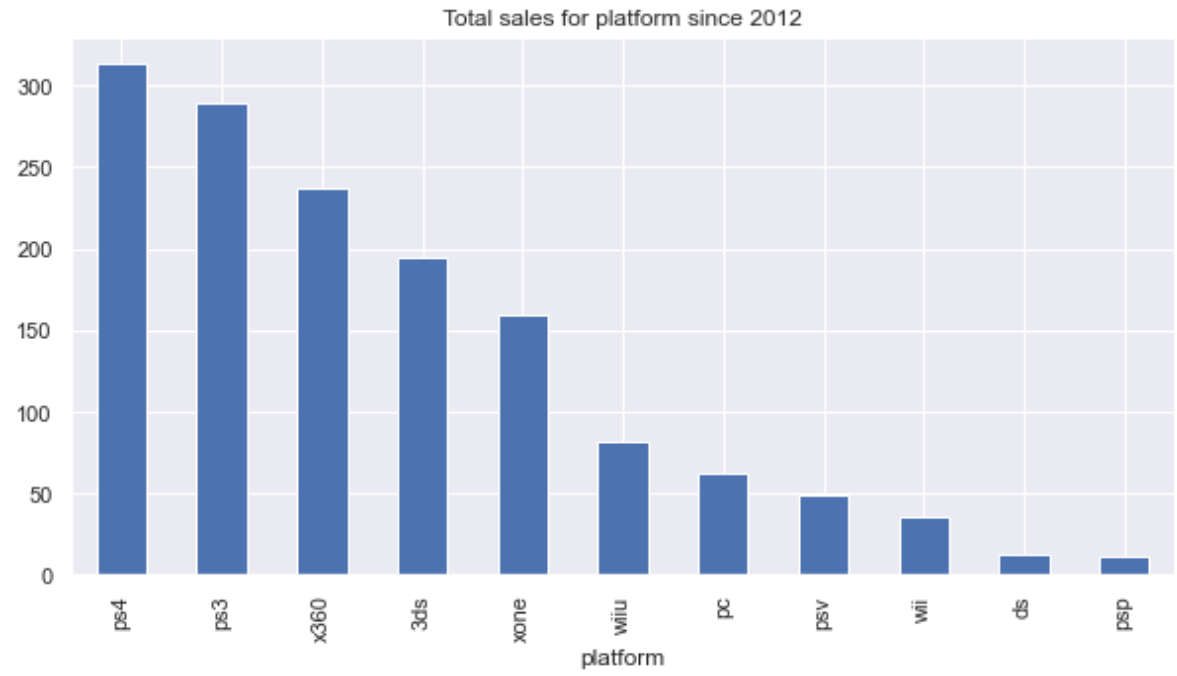
With those findings we should take the period of about 4 years, from 2012 and upwards.

In [36]:

```
df2 = df2.query('year_of_release >= 2012')
```

In [37]:

```
df2.groupby('platform')['total_sales'].sum().sort_values(ascending=False).plot(kind='bar',
```



well use z-score to determine to how far the value is from the mean value(std), to see which ones have the highest sale record.

In [38]:

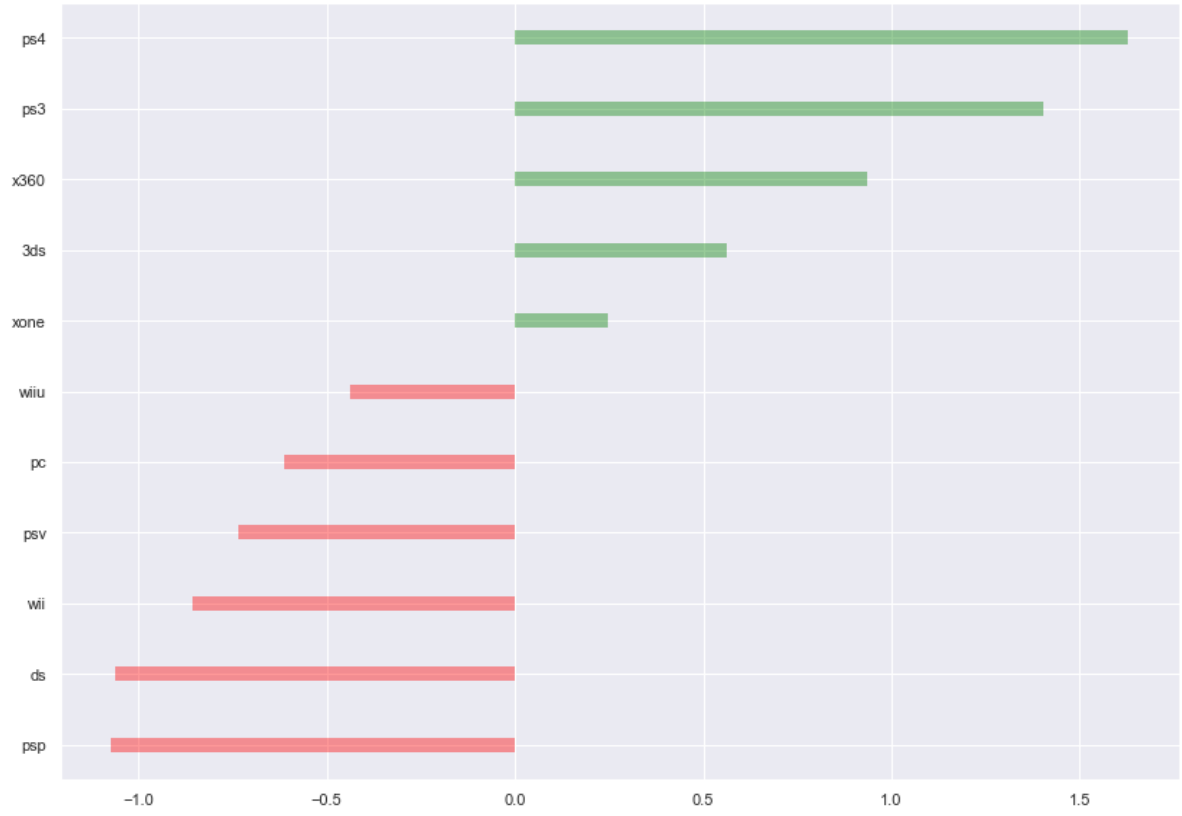
```
df_sales = df2.groupby('platform')['total_sales'].sum().sort_values().reset_index()
df_sales['sales_z'] = (df_sales['total_sales']- df_sales['total_sales'].mean())/df_sales['t
df_sales['color'] = ['red' if x<0 else 'green' for x in df_sales['sales_z']]
df_sales.tail()
```

Out[38]:

	platform	total_sales	sales_z	color
6	xone	159.32	0.248111	green
7	3ds	194.61	0.562873	green
8	x360	236.54	0.936860	green
9	ps3	288.79	1.402894	green
10	ps4	314.14	1.628998	green

In [39]:

```
plt.figure(figsize=(14,10))
plt.hlines(y= df_sales.platform, xmin=0,xmax=df_sales.sales_z,color=df_sales.color,alpha=0.
```



In [40]:

```
df_tree = df2.groupby('platform')['total_sales'].sum().sort_values().reset_index()
df_tree = df_tree.query('total_sales != 0')
```

In [41]:

```
sizes = df_tree['total_sales'].values.tolist()
labels = df_tree.apply(lambda x: str(x[0])+"\n"+"$"+str(round(x[1])),axis = 1)
```

In [42]:

```
plt.figure(figsize=(15,9))
squarify.plot(sizes=sizes, label=labels);
```

In [43]:

```
df_pivot = df2.pivot_table(index='year_of_release', columns='platform', values='total_sales
df_pivot
```

Out[43]:

platform	3ds	ds	pc	ps3	ps4	psp	psv	wii	wiiu	x360	xone
year_of_release											
2012	51.36	11.01	23.22	107.36	0.00	7.69	16.19	21.71	17.56	99.74	0.00
2013	56.57	1.54	12.38	113.25	25.99	3.14	10.59	8.59	21.65	88.58	18.96
2014	43.76	0.00	13.28	47.76	100.00	0.24	11.90	3.75	22.03	34.74	54.07
2015	27.78	0.00	8.52	16.82	118.90	0.12	6.25	1.14	16.35	11.96	60.14
2016	15.14	0.00	5.25	3.60	69.25	0.00	4.25	0.18	4.60	1.52	26.15

In [44]:

```
df_pivot.shift(+1).head()
```

Out[44]:

platform	3ds	ds	pc	ps3	ps4	psp	psv	wii	wiiu	x360	xone
year_of_release											
2012	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013	51.36	11.01	23.22	107.36	0.00	7.69	16.19	21.71	17.56	99.74	0.00
2014	56.57	1.54	12.38	113.25	25.99	3.14	10.59	8.59	21.65	88.58	18.96
2015	43.76	0.00	13.28	47.76	100.00	0.24	11.90	3.75	22.03	34.74	54.07
2016	27.78	0.00	8.52	16.82	118.90	0.12	6.25	1.14	16.35	11.96	60.14

In [45]:

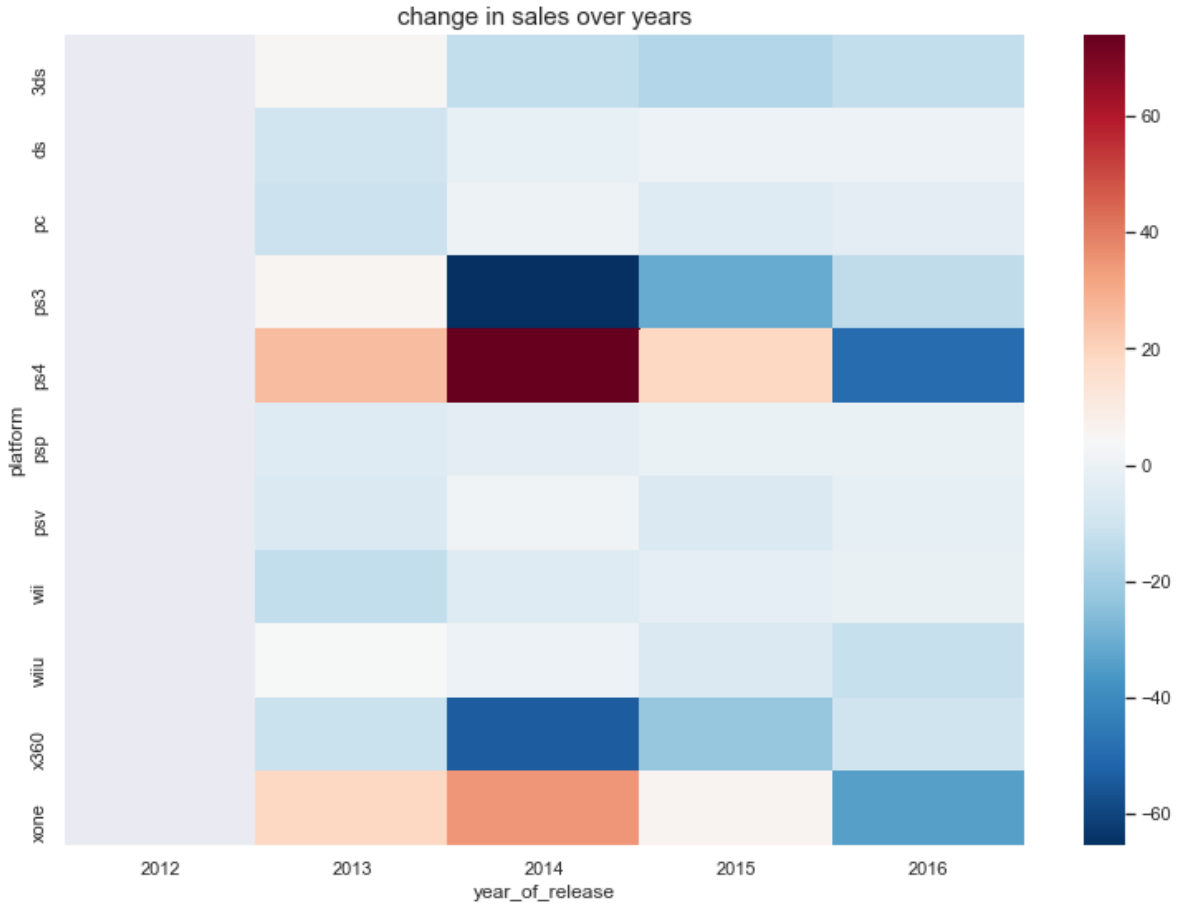
```
dynamics = df_pivot-df_pivot.shift(+1)
dynamics
```

Out[45]:

platform	3ds	ds	pc	ps3	ps4	psp	psv	wii	wiiu	x360	xone
year_of_release											
2012	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013	5.21	-9.47	-10.84	5.89	25.99	-4.55	-5.60	-13.12	4.09	-11.16	18.96
2014	-12.81	-1.54	0.90	-65.49	74.01	-2.90	1.31	-4.84	0.38	-53.84	35.11
2015	-15.98	0.00	-4.76	-30.94	18.90	-0.12	-5.65	-2.61	-5.68	-22.78	6.07
2016	-12.64	0.00	-3.27	-13.22	-49.65	-0.12	-2.00	-0.96	-11.75	-10.44	-33.99

In [46]:

```
plt.figure(figsize=(13,9))
sns.heatmap(dynamics.T, cmap='RdBu_r').set_title('change in sales over years', fontdict={'s
```



In [47]:

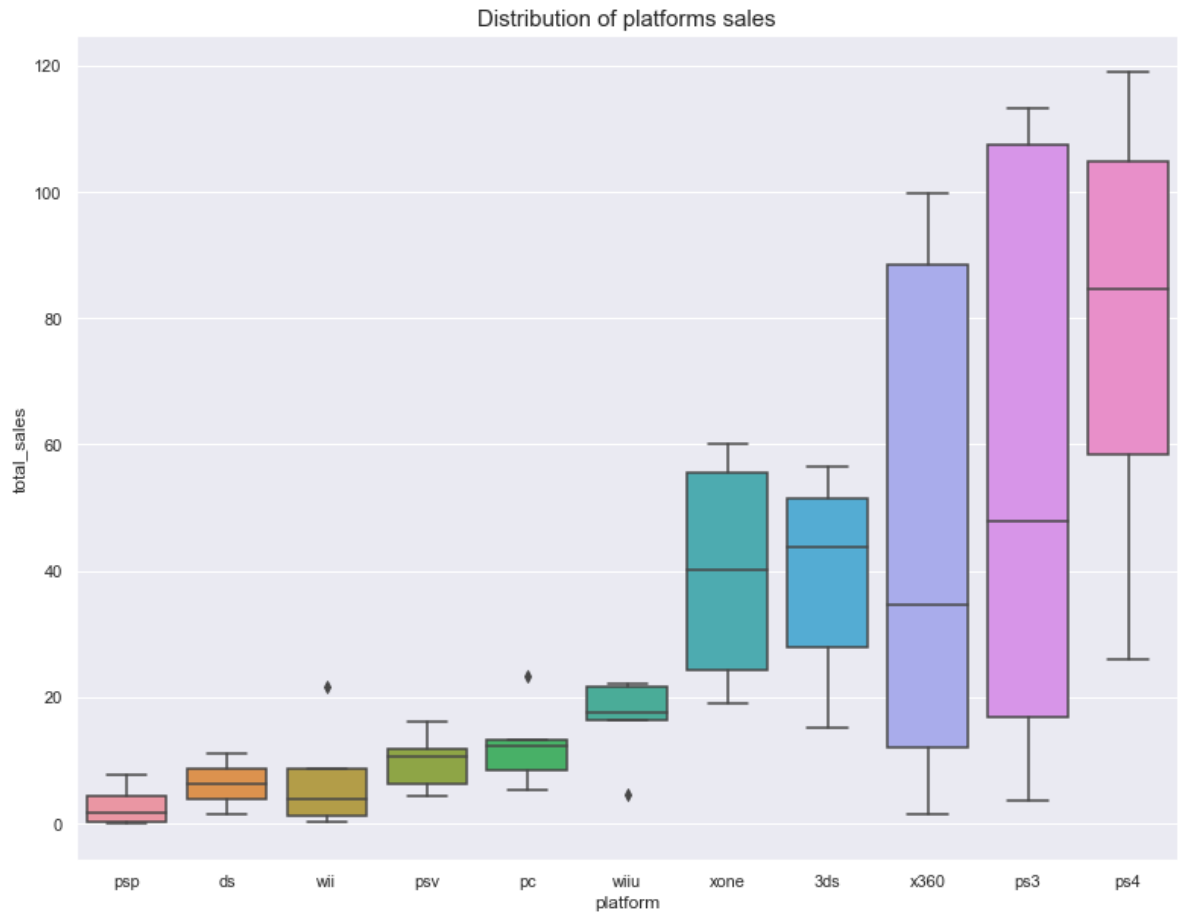
```
grouped = df2.groupby(['platform','year_of_release'])['total_sales'].sum().reset_index()
ordered = grouped.groupby(['platform'])['total_sales'].sum().sort_values().reset_index()['p
ordered
```

Out[47]:

```
0      psp
1       ds
2      wii
3      psv
4       pc
5     wiiu
6     xone
7      3ds
8     x360
9      ps3
10     ps4
Name: platform, dtype: object
```

In [48]:

```
plt.figure(figsize=(13,10))
sns.boxplot(x='platform',y='total_sales', data=grouped, order=ordered).set_title('Distribut
```



By both the charts and the tables it is noticeble that some platforms like: ds, gba, gc, ps2, psp, psv, Wii, and xb are or declining in sales or stopped selling at all. on the other hand platforms like: ps4, ps3,ps2 , pc, xone, wiiu, 3ds some of those platforms have the biggest sales across the charts, its seems that

PS2, X360, ps3 have the biggest sales although some of them are declining as the years go by, theres an exception for the pc platform though it seems it goes up and down.
our potentially profitable platforms are: Ps4,ps3, xone and pc.

Now we'll see how user and proffesional reviews affect our sales on the ps4 platform.

In [49]:

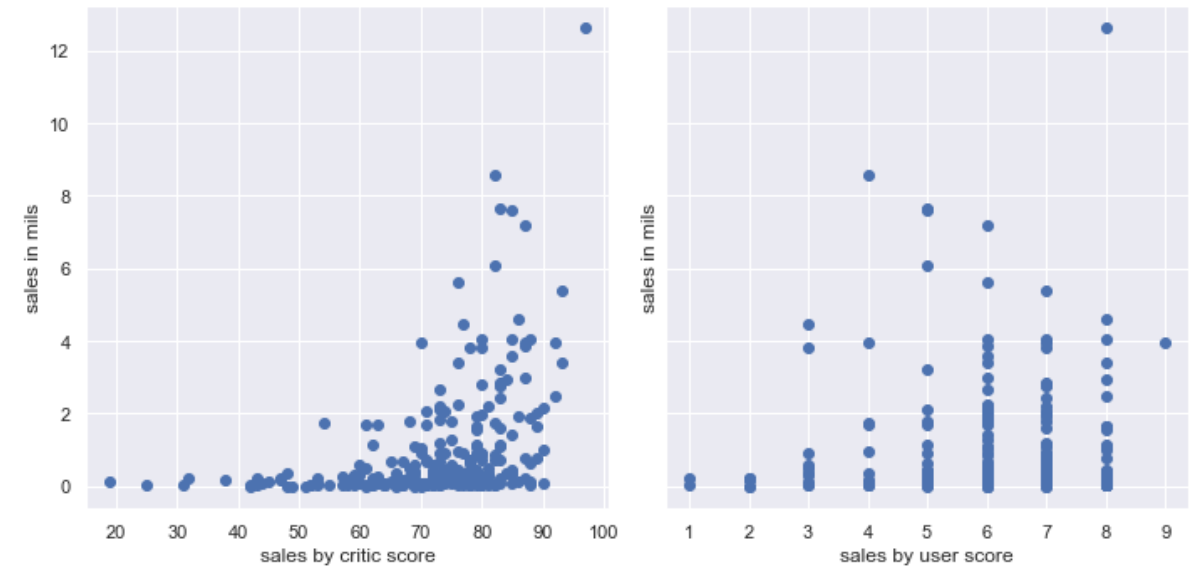
```
df_ps2 = df2.query('platform == "ps4" and critic_score != 0 and user_score != 0')
```

In [50]:

```
display_side_by_side([df_ps2.corr()],['Correlation'])
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10, 5),sharey=True)
axes[0].scatter(x=df_ps2['critic_score'], y = df_ps2['total_sales'])
axes[1].scatter(x=df_ps2['user_score'], y = df_ps2['total_sales'])
axes[0].set(xlabel="sales by critic score")
axes[1].set(xlabel="sales by user score")
axes[0].set(ylabel="sales in mils")
axes[1].set(ylabel="sales in mils")
fig.tight_layout();
```

Correlation

	year_of_release	na_sales	eu_sales	jp_sales	other_sales	critic_score	user
year_of_release	1.000000	-0.261760	-0.191355	-0.122830	-0.232177	-0.019983	0.
na_sales	-0.261760	1.000000	0.714988	0.530480	0.915292	0.414241	-0.
eu_sales	-0.191355	0.714988	1.000000	0.519826	0.935136	0.346044	-0.
jp_sales	-0.122830	0.530480	0.519826	1.000000	0.566734	0.322057	0.
other_sales	-0.232177	0.915292	0.935136	0.566734	1.000000	0.408465	-0.
critic_score	-0.019983	0.414241	0.346044	0.322057	0.408465	1.000000	0.
user_score	0.133063	-0.026950	-0.037972	0.180359	-0.031599	0.562397	1.
total_sales	-0.234400	0.893807	0.951656	0.592214	0.997798	0.405895	-0.



In [51]:

```
list_of_plats = ['ps4','ps3','x360','3ds','xone',"pc"]

def find_corr(platform):
    for i in platform:
        dff = df2.query('platform == @i')
        user = dff.corr()['total_sales']['user_score']
        critic = dff.corr()['total_sales']['critic_score']
        if user > 0 and critic >0:
            print(i ,"[-]" ,user, '-- User score with sales')
            print("-----" , critic, '-- Critic score with sales')
            print("Both Critic and user score have positive correlation with total sales")
            print()
        else:
            print(i ,"[-]" ,user, '-- User score with sales')
            print("-----" , critic, '-- Critic score with sales')
            print('Either user or critic score have a negative or no correlation at all')
            print()
```

In [52]:

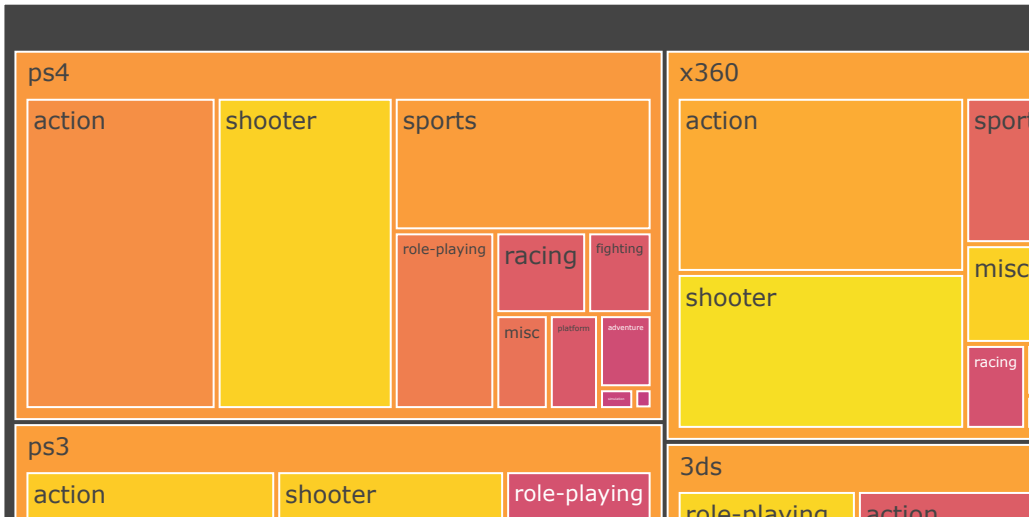
```
find_corr(df2['platform'].unique())
```

```
ps3 [-] 0.2110553734229138 -- User score with sales
----- 0.29897905971421274 -- Critic score with sales
Both Critic and user score have positive correlation with total sales

x360 [-] 0.09522554922085064 -- User score with sales
----- 0.26202840220219736 -- Critic score with sales
Both Critic and user score have positive correlation with total sales

ps4 [-] 0.11257877945234093 -- User score with sales
----- 0.22754960627752624 -- Critic score with sales
Both Critic and user score have positive correlation with total sales

3ds [-] 0.13055327685717946 -- User score with sales
----- 0.15026490592743424 -- Critic score with sales
Both Critic and user score have positive correlation with total sales

ds [-] 0.07869810944153977 -- User score with sales
----- 0.07932569762230195 -- Critic score with sales
Both Critic and user score have positive correlation with total sales

xone [-] 0.1098227536154239 -- User score with sales
----- 0.27483918749445324 -- Critic score with sales
Both Critic and user score have positive correlation with total sales

wiiu [-] 0.28139294472088117 -- User score with sales
----- 0.29812398749349156 -- Critic score with sales
Both Critic and user score have positive correlation with total sales

wii [-] 0.618356296689155 -- User score with sales
----- 0.5381207067838356 -- Critic score with sales
Both Critic and user score have positive correlation with total sales

pc [-] 0.036960316022980645 -- User score with sales
----- 0.20422121304922355 -- Critic score with sales
Both Critic and user score have positive correlation with total sales

psv [-] 0.36954141264059664 -- User score with sales
----- 0.39507986289512054 -- Critic score with sales
Both Critic and user score have positive correlation with total sales

psp [-] 0.2008143919701118 -- User score with sales
----- 0.09150966741463094 -- Critic score with sales
Both Critic and user score have positive correlation with total sales
```

It seems theres a positive correlation between sales and both user and critic scores, although the correlation is small for user scores, the critic scores is about twice that and does affect the games sales. Simply, the better the score the better the game will sell.

In [53]:

```
fig = px.treemap(df2, path=['platform', 'genre'], values='total_sales',color='total_sales',
fig.show()
```



As the figure above is pretty self explanatory, most popular and top selling are the "shooter","action" and "sports" on our leading platforms.
while the least profitable ones are the: "racing", "fighting" and "adventure"..

Region profiles

Lets make profiles for each region.

In [54]:

```
NA = df2.query("platform != 'ps2' and platform != 'ps3' and platform != 'x360' and platform
EU = df2.query("platform != 'ps2' and platform != 'ps3' and platform != 'x360' and platform
JP = df2.query("platform != 'ps2' and platform != 'ps3' and platform != 'x360' and platform
```

In [55]:

```
display_side_by_side([NA[:5],EU[:5],JP[:5]],['NA Sales','EU Sales','JP Sales'])
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(10, 5),sharey=True)
axes[0].pie(NA['na_sales'][:5], labels=NA['platform'][:5],explode=[0.05]*5,autopct="%.1f%%"
axes[1].pie(EU['eu_sales'][:5], labels=EU['platform'][:5],explode=[0.05]*5,autopct="%.1f%%"
axes[2].pie(JP['jp_sales'][:5], labels=JP['platform'][:5],explode=[0.05]*5,autopct="%.1f%%"
axes[0].set(xlabel="NA")
axes[1].set(xlabel="EU")
axes[2].set(xlabel="JP")
fig.tight_layout();
```

NA Sales

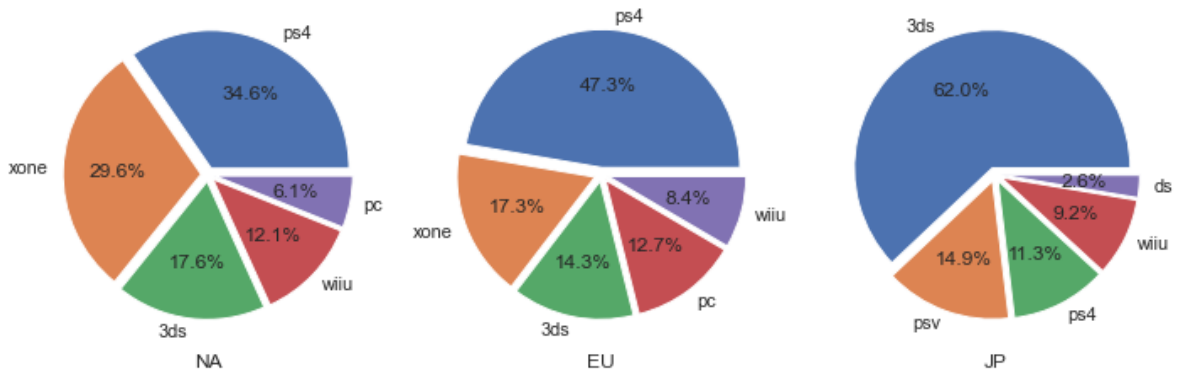
	platform	na_sales
0	ps4	108.740000
1	xone	93.120000
2	3ds	55.310000
3	wiiu	38.100000
4	pc	19.120000

EU Sales

	platform	eu_sales
0	ps4	141.090000
1	xone	51.590000
2	3ds	42.640000
3	pc	37.760000
4	wiiu	25.130000

JP Sales

	platform	jp_sales
0	3ds	87.790000
1	psv	21.040000
2	ps4	15.960000
3	wiiu	13.010000
4	ds	3.720000



These are the top 5 overall selling platforms, Its noticeable that the JP market is the smallest in term of sales and slightly different in variety of platforms, EU is in the middle sharing the platforms mostly with NA.
NA is the biggest in sales preferring PS4 as their leading platform following by Xone and 3ds, The JP region has slightly different taste.

In [56]:

```
NA_genre = df2.groupby('genre')['na_sales'].sum().sort_values(ascending=False).reset_index(
EU_genre = df2.groupby('genre')['eu_sales'].sum().sort_values(ascending=False).reset_index(
JP_genre = df2.groupby('genre')['jp_sales'].sum().sort_values(ascending=False).reset_index(
```

In [57]:

```
display_side_by_side([NA_genre[:5],EU_genre[:5],JP_genre[:5]],['NA Genre','EU Genre','JP Ge
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(10, 5),sharey=True)
axes[0].pie(NA_genre['na_sales'][:5], labels=NA_genre['genre'][:5],explode=[0.05]*5,autopct
axes[1].pie(EU_genre['eu_sales'][:5], labels=EU_genre['genre'][:5],explode=[0.05]*5,autopct
axes[2].pie(JP_genre['jp_sales'][:5], labels=JP_genre['genre'][:5],explode=[0.05]*5,autopct
axes[0].set(xlabel="NA")
axes[1].set(xlabel="EU")
axes[2].set(xlabel="JP")
fig.tight_layout();
```

NA Genre

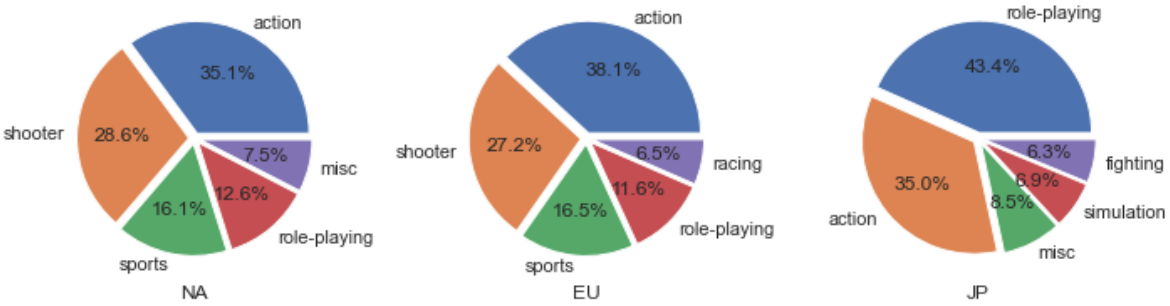
	genre	na_sales
0	action	177.840000
1	shooter	144.770000
2	sports	81.530000
3	role-playing	64.000000
4	misc	38.190000

EU Genre

	genre	eu_sales
0	action	159.340000
1	shooter	113.470000
2	sports	69.090000
3	role-playing	48.530000
4	racing	27.290000

JP Genre

	genre	jp_sales
0	role-playing	65.440000
1	action	52.800000
2	misc	12.860000
3	simulation	10.410000
4	fighting	9.440000



Seems like the genre tastes are pretty close they all have action, shooter and sports among the top 3 favourites, the difference is that JP has a top role-playing favourite. That difference is probably due to a different cultural mindset.

In [58]:

```
NA_rating = df2.groupby('rating')['na_sales'].sum().sort_values(ascending=False).reset_inde
EU_rating = df2.groupby('rating')['eu_sales'].sum().sort_values(ascending=False).reset_inde
JP_rating = df2.groupby('rating')['jp_sales'].sum().sort_values(ascending=False).reset_inde
```

In [59]:

```
display_side_by_side([NA_rating,EU_rating,JP_rating],['NA Rating','EU Rating','JP Rating'])
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(10, 5),sharey=True)
axes[0].pie(NA_rating['na_sales'], labels=NA_rating['rating'],explode=[0.05]*len(NA_rating[
axes[1].pie(EU_rating['eu_sales'], labels=EU_rating['rating'],explode=[0.05]*len(EU_rating[
axes[2].pie(JP_rating['jp_sales'], labels=JP_rating['rating'],explode=[0.05]*len(JP_rating[
axes[0].set(xlabel="NA")
axes[1].set(xlabel="EU")
axes[2].set(xlabel="JP")
fig.tight_layout();
```

NA Rating

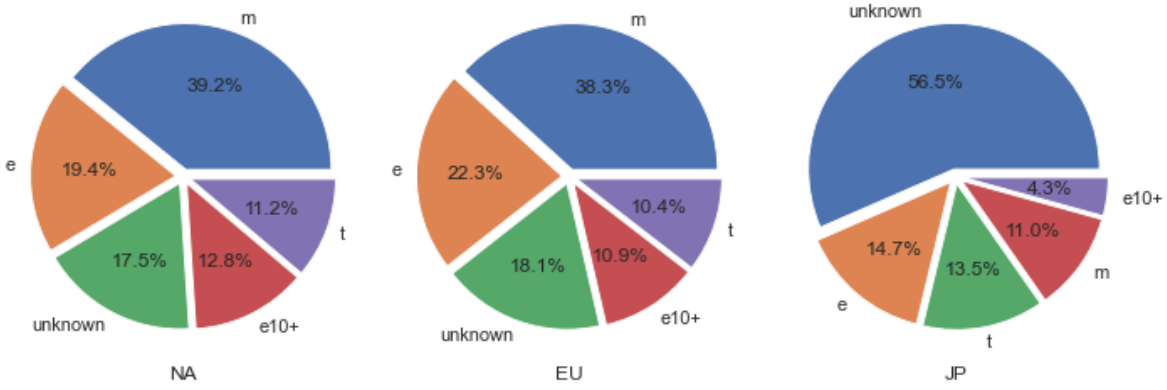
	rating	na_sales
0	m	231.570000
1	e	114.370000
2	unknown	103.310000
3	e10+	75.700000
4	t	66.020000

EU Rating

	rating	eu_sales
0	m	193.960000
1	e	113.030000
2	unknown	91.500000
3	e10+	55.370000
4	t	52.960000

JP Rating

	rating	jp_sales
0	unknown	108.840000
1	e	28.330000
2	t	26.020000
3	m	21.200000
4	e10+	8.190000



It seems that Eu and Na are about the sames in terms of rating for games. but in the JP market people are less prone to buy a 'M' rated games. so the ESRB ratings are only affecting in the JP region

Majority of the NA and EU have a bigger audiance for unrated games, on the other hand JP has more than 50% of audiance for games that are in our 'Uknown' category.

Hypothesis testing

Now We will test "Average user ratings of the Xbox One and PC platforms are the same":

$$H_0 - XOne \neq Pc$$
$$H_1 - XOne = Pc$$

In [60]:

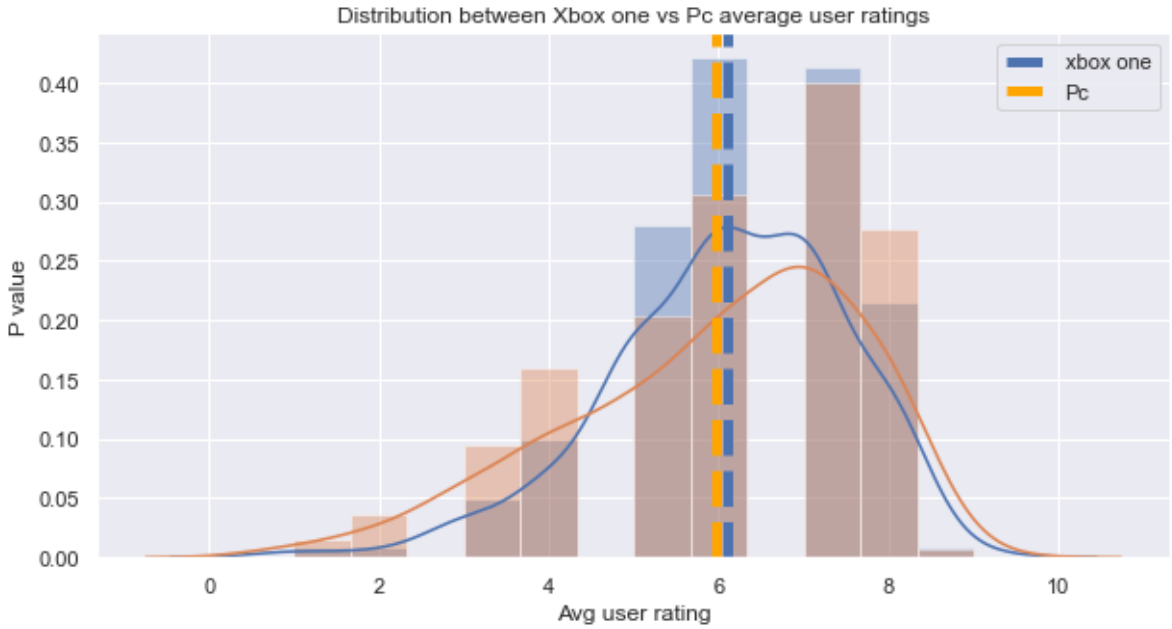
```
%matplotlib inline
warnings.filterwarnings("ignore")
rcParams['figure.figsize'] = 10,5
rcParams['font.size'] = 30
sns.set()
np.random.seed(8)

def plot_distribution(inp):
    warnings.filterwarnings("ignore")
    plt.figure()
    ax = sns.distplot(inp)
    plt.axvline(np.mean(inp), color="k", linestyle="dashed", linewidth=5)
    _, max_ = plt.ylim()
    plt.text(
        inp.mean() + inp.mean() / 10,
        max_ - max_ / 10,
        "Mean: {:.2f}".format(inp.mean()),
    )
    return plt.figure;
```

In [61]:

```
xone = df2.query('platform =="xone" and user_score != 0')
pc = df2.query('platform =="pc" and user_score != 0')
```

In [62]:

```
plt.figure()
ax1 = sns.distplot(xone['user_score']).set_title('Distribution between Xbox one vs Pc avera
ax2 = sns.distplot(pc['user_score'])
plt.axvline(np.mean(xone['user_score']), color='b', linestyle='dashed', linewidth=5, label=
plt.axvline(np.mean(pc['user_score']), color='orange', linestyle='dashed', linewidth=5,labe
ax2.set(xlabel='Avg user rating', ylabel='P value')
plt.legend();
```



We'll run levene's test to check for equality

In [63]:

```
alpha = 0.05

results = st.levene(xone['user_score'], pc['user_score'], center='mean')

print('p-value: ',results.pvalue)
if (results.pvalue < alpha):
    print("We reject the null hypothesis")
else:
    print("We can't reject the null hypothesis")
```

```
p-value:  0.002959821063467202
We reject the null hypothesis
```

In [64]:

```
alpha = 0.05

results = st.ttest_ind(xone['user_score'], pc['user_score'], equal_var=False)

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

if (results.pvalue < alpha):
    print("We reject the null hypothesis")
else:
    print("We can't reject the null hypothesis")
```

```
p-value:  0.35323200542841515
We can't reject the null hypothesis
```

We cant reject the null hypothesis and assume that they are equal.

Next we will check if "Average user ratings for the Action and Sports genres are different.":

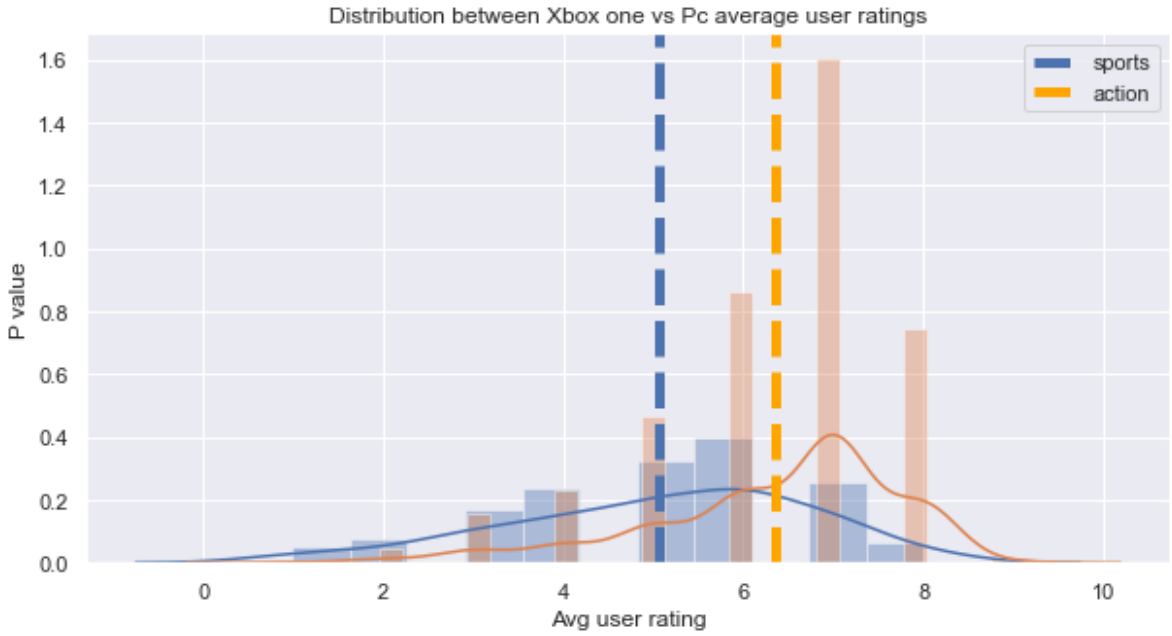
$$H_0 - Action = Sports$$
$$H_1 - Action \neq Sports$$

In [65]:

```
sport = df2.query('genre == "sports" and user_score != 0')
action = df2.query('genre == "action" and user_score != 0')
```

In [66]:

```
plt.figure()
ax1 = sns.distplot(sport['user_score']).set_title('Distribution between Xbox one vs Pc aver
ax2 = sns.distplot(action['user_score'])
plt.axvline(np.mean(sport['user_score']), color='b', linestyle='dashed', linewidth=5, label
plt.axvline(np.mean(action['user_score']), color='orange', linestyle='dashed', linewidth=5,
ax2.set(xlabel='Avg user rating', ylabel='P value')
plt.legend();
```



In [67]:

```
alpha = 0.05

results = st.levene(sport['user_score'], action['user_score'], center='mean')

print('p-value: ',results.pvalue)
if (results.pvalue < alpha):
    print("We reject the null hypothesis")
else:
    print("We can't reject the null hypothesis")
```

```
p-value:  0.0015324700661894543
We reject the null hypothesis
```

We reject the null hypothesis and assume that they are different

In [68]:

```
alpha = 0.05

results = st.ttest_ind(sport['user_score'], action['user_score'], equal_var=False)

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

if (results.pvalue < alpha):
    print("We reject the null hypothesis")
else:
    print("We can't reject the null hypothesis")
```

```
p-value:  2.4900817322539363e-19
We reject the null hypothesis
```

We reject the null hypothesis therefore we assume that there is a significant difference between the means of the two groups.

Overall conclusion

So to try and determine "What makes a game successful" lets summarize our findings: First thing is that the biggest spike in game releases was in 2008-2009 and our optimal period is from 2012 up 2016. Our top selling platforms are ps2, x360, ps3, wii and ds regarding lifetime sales. We can see that successful games, in terms of sales have a positive correlation with critic and user reviews. Also theres impact on sales depending on platforms and genres which vary by region. A games success depends highly on its platform relevance, which in our case is about 4 years time. So for a game to succeed its preferably to be released on a relevant platform (such as ps4 or Xone),getting high user and critic scores will affect its success, Being marketed in the right region with an appropriate rating for that same region is also a key factor. Its important to see how much time a platform predecessor is on the market so we could decide wether we want to keep advertising for that same platform series. For a better profit its important to advertise in the right regions(for example JP has the lowest sales total because the region is smaller in population, and have different cultural preferences). In our case the potentially profitable platforms are ps4, ps3, xone and pc, as they had a good predecessor success. Thus for further longevity of sales its a better decsion to market games related to those platforms. We can also notice that the highest average of games sold is shooter games on those platforms. Rating wise its seems that NA and EU are simillar rating wise, JP region seems to have more than half of its game sold under the 'unknown' category, which we have a missing information about, The JP region is also the smallest of the three regarding population size.

In [ ]:

```
```