Проект

Вступление

Краткое описание проекта

Вы работаете в интернет-магазине «Стримчик», который продаёт по всему миру компьютерные игры. Из открытых источников доступны исторические данные о продажах игр, оценки пользователей и экспертов, жанры и платформы. Доступны данные до 2016 года.

Цель проекта

Нужно выявить определяющие успешность игры закономерности. Это позволит сделать ставку на потенциально популярный продукт и спланировать рекламные кампании.

Описание данных

Столбцы:

- Name — название игры
- Platform — платформа
- Year_of_Release — год выпуска
- Genre — жанр игры
- NA_sales — продажи в Северной Америке (миллионы проданных копий)
- EU_sales — продажи в Европе (миллионы проданных копий)
- JP_sales — продажи в Японии (миллионы проданных копий)
- Other_sales — продажи в других странах (миллионы проданных копий)
- Critic_Score — оценка критиков (максимум 100)
- User_Score — оценка пользователей (максимум 10)
- Rating — рейтинг от организации ESRB (англ. Entertainment Software Rating Board). Эта ассоциация определяет рейтинг компьютерных игр и присваивает им подходящую возрастную категорию.

Обозначения

- Слово - столбец из датафрейма
- Слово - значение входящее в один из столбцов датафрейма
- Слово - географические обьекты
- Слово - компании и огранизации
- Слово - внешние ссылки
- Слово - важная информация

Содержание

Подключение датафрейма и нужных библиотек

Подключение нужных библиотек

```
!pip install missingno
import pandas as pd
import missingno as msno
import matplotlib.pyplot as plt
import scipy.stats as st
```

```
Collecting missingno
  Downloading missingno-0.5.1-py3-none-any.whl (8.7 kB)
Requirement already satisfied: numpy in /opt/conda/lib/python3.9/site-
packages (from missingno) (1.21.1)
Requirement already satisfied: scipy in /opt/conda/lib/python3.9/site-
packages (from missingno) (1.9.1)
Requirement already satisfied: seaborn in
/opt/conda/lib/python3.9/site-packages (from missingno) (0.11.1)
Requirement already satisfied: matplotlib in
/opt/conda/lib/python3.9/site-packages (from missingno) (3.3.4)
Requirement already satisfied: kiwisolver>=1.0.1 in
/opt/conda/lib/python3.9/site-packages (from matplotlib->missingno)
(1.4.4)
Requirement already satisfied: pillow>=6.2.0 in
/opt/conda/lib/python3.9/site-packages (from matplotlib->missingno)
(8.4.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!
=2.1.6,>=2.0.3 in /opt/conda/lib/python3.9/site-packages (from
matplotlib->missingno) (2.4.7)
Requirement already satisfied: python-dateutil>=2.1 in
/opt/conda/lib/python3.9/site-packages (from matplotlib->missingno)
(2.8.1)
Requirement already satisfied: cycler>=0.10 in
/opt/conda/lib/python3.9/site-packages (from matplotlib->missingno)
(0.11.0)
Requirement already satisfied: six>=1.5 in
/opt/conda/lib/python3.9/site-packages (from python-dateutil>=2.1-
>matplotlib->missingno) (1.16.0)
Requirement already satisfied: pandas>=0.23 in
/opt/conda/lib/python3.9/site-packages (from seaborn->missingno)
(1.2.4)
Requirement already satisfied: pytz>=2017.3 in
/opt/conda/lib/python3.9/site-packages (from pandas>=0.23->seaborn-
>missingno) (2021.1)
Installing collected packages: missingno
Successfully installed missingno-0.5.1
```

Подключение датафрейма

```
df = pd.read_csv("/datasets/games.csv")
```

Ознакомление с данными

Информация о датафрейме

```
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
```

Пример из датафрейма

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

```
                      Name Platform  Year_of_Release         
Genre  \
0                 Wii Sports      Wii           2006.0        Sports

1          Super Mario Bros.      NES           1985.0      Platform

2             Mario Kart Wii      Wii           2008.0        Racing

3          Wii Sports Resort      Wii           2009.0        Sports

4   Pokemon Red/Pokemon Blue       GB           1996.0  Role-Playing

5                     Tetris       GB           1989.0        Puzzle

6      New Super Mario Bros.       DS           2006.0      Platform

7                   Wii Play      Wii           2006.0          Misc
```

```
8  New Super Mario Bros. Wii      Wii           2009.0      Platform

9                  Duck Hunt      NES           1984.0       Shooter


   NA_sales  EU_sales  JP_sales  Other_sales  Critic_Score User_Score
Rating
0     41.36     28.96      3.77         8.45          76.0          8
E
1     29.08      3.58      6.81         0.77           NaN        NaN
NaN
2     15.68     12.76      3.79         3.29          82.0        8.3
E
3     15.61     10.93      3.28         2.95          80.0          8
E
4     11.27      8.89     10.22         1.00           NaN        NaN
NaN
5     23.20      2.26      4.22         0.58           NaN        NaN
NaN
6     11.28      9.14      6.50         2.88          89.0        8.5
E
7     13.96      9.18      2.93         2.84          58.0        6.6
E
8     14.44      6.94      4.70         2.24          87.0        8.4
E
9     26.93      0.63      0.28         0.47           NaN        NaN
NaN
```

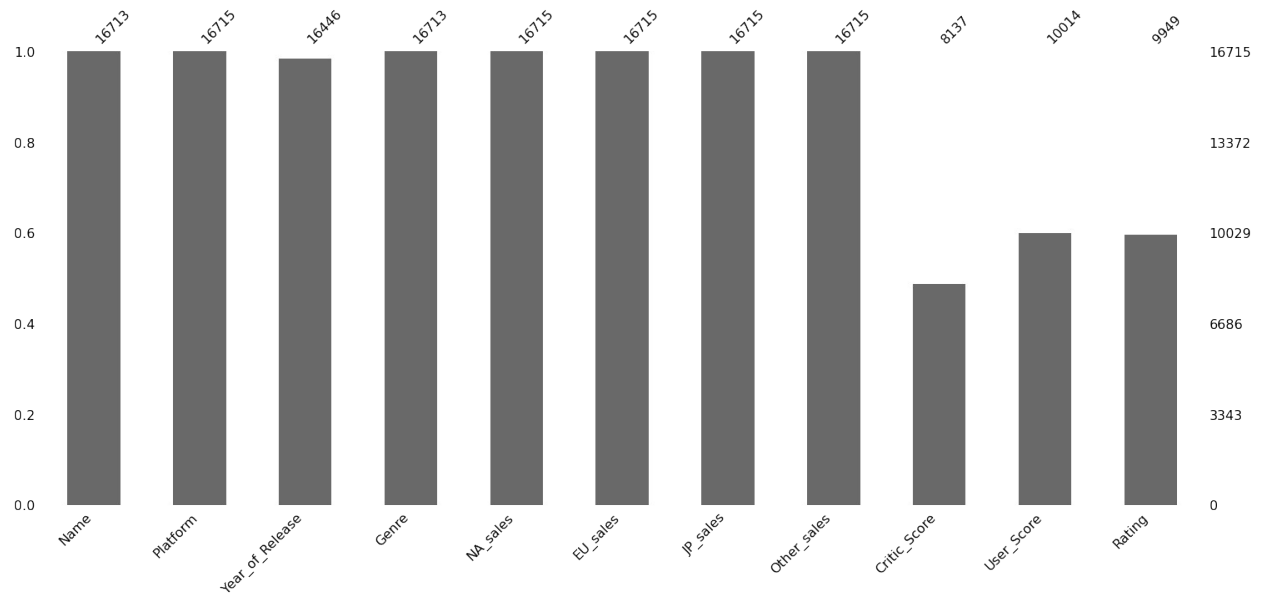
Названия столбцов датафрейма

```
list(df.columns)
```

```
['Name',
 'Platform',
 'Year_of_Release',
 'Genre',
 'NA_sales',
 'EU_sales',
 'JP_sales',
 'Other_sales',
 'Critic_Score',
 'User_Score',
 'Rating']
```

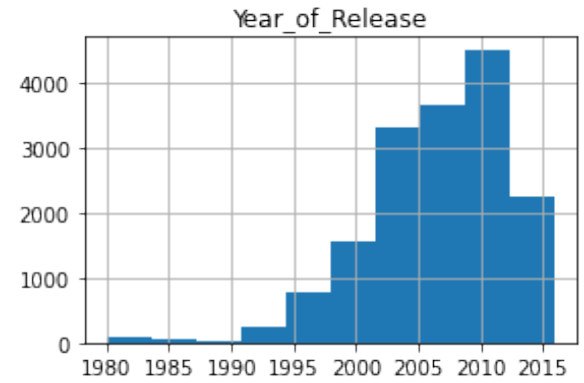
Кол-во пропусков в каждом столбце

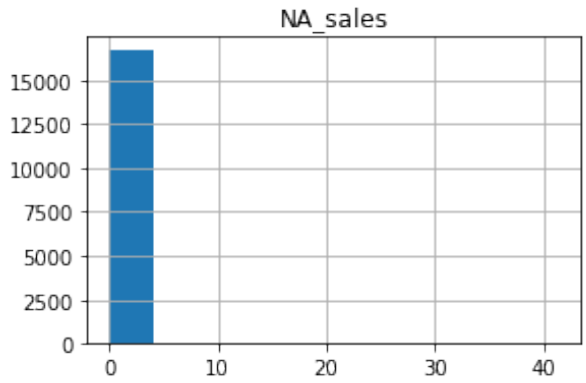
```
msno.bar(df)
plt.show()
```

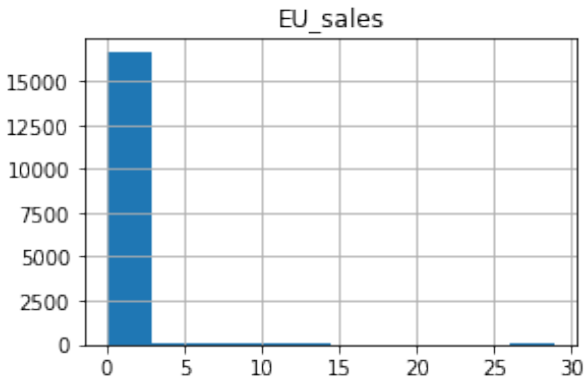


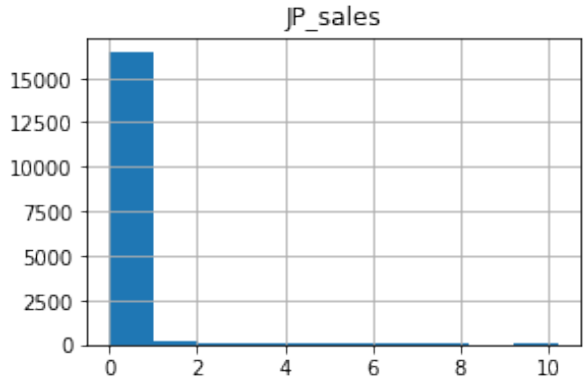
Гистограммы для некоторых столбцов

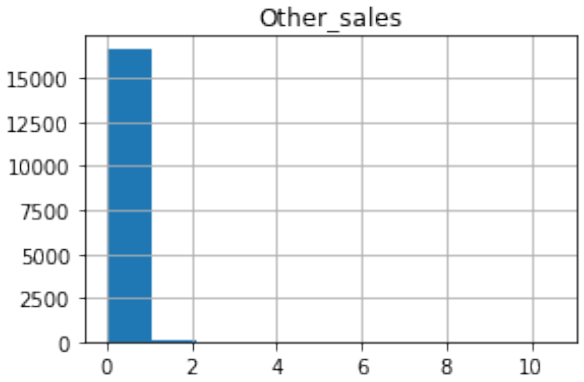
```
df.hist(figsize=(10, 10))
plt.show()
```

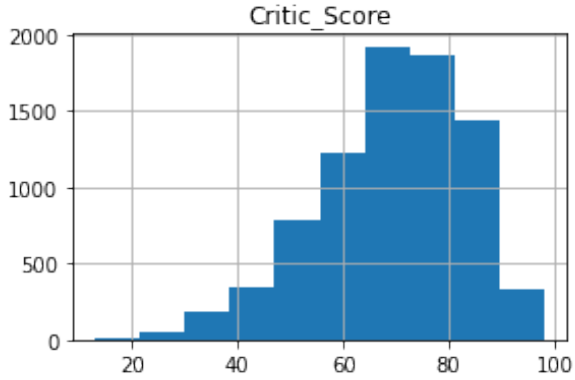












Явные дубликаты строк

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

```
0
```

```
df.duplicated(subset=['Name', 'Platform', 'Year_of_Release']).sum()
```

```
2
```

```
df.drop_duplicates(subset=['Name', 'Platform', 'Year_of_Release'],
inplace=True)
```

```
df.duplicated(subset=['Name', 'Platform', 'Year_of_Release']).sum()
```

```
0
```

Вывод из обзора данных

- **В данных присутствуют пропуски, в столбцах 'Name', 'Year_of_Release' и 'Genre' их малое кол-во, а в столбцах 'Critic_Score', 'User_Score' и 'Rating' кол-во пропусков немного меньше или равно половине от общего кол-ва данных.**
- **В столбцах 'Year_of_Release' и 'Critic_Score' стоит изменить тип данных на целочисленный, а в столбце 'User_Score' - на вещественный.**
- **В столбцах 'NA_sales', 'EU_sales', 'JP_sales', 'Other_sales' присутствуют выбросы, из-за которых, их гистограммы выглядят неинформативно.**
- **Явные дубликаты строк, сгрупированные по году выпуска, имени, платформе, удалены.**

Предобработка данных

Корректировка названий столбцов

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

Обработка столбца 'year_of_release'

```
df['year_of_release'] = df['year_of_release'].fillna(0).astype('int')
```

Заполним пропуски нулями, т.к. это не нужная нам информация, и изменим тип данных на целочисленный.

Обработка столбца 'name'

```
df.dropna(subset=['name'], inplace=True)
```

Обработка столбца 'genre'

```
df['genre'] = df['genre'].fillna('unknown')
```

Обработка столбца 'rating'

```
#rating
df['rating'] = df['rating'].fillna('undef')
print(df['rating'].value_counts())
df['rating'].value_counts().plot(kind='bar', title='Распределение
рейтингов')
plt.show()
```

```
undef    6764
E        3989
T        2961
M        1563
E10+     1420
```

```
EC          8
RP          3
K-A         3
AO          1
Name: rating, dtype: int64
```


```
df['rating'] = df['rating'].replace('K-A', 'E').replace('AO',
'M').replace('RP', 'undef')

df = df[df['rating'] != 'EC']
```

Заменим рейтинг 'K-A' на 'E', т.к. 'K-A' это прошлое название рейтинга 'E', 'AO' на рейтинг 'M', ввиду их схожести и малочисленности рейтинга 'AO', 'RP' на значение-заглушку, т.к. это по сути тоже пропуски, а строки с рейтингом 'EC' удалим, потому что они составляют пренебрежимо малую долю в данных. (информация из интернета: источник)

Обработка столбца 'user_score'

```
#user_score
print(df['user_score'].unique())
df.loc[df['user_score'] == 'tbd', 'user_score'] = float('NaN')
df['user_score'] = df['user_score'].astype('float32')
```

```
['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']
```

Заметим что 'tbd' это еще одно обозначение пропусков, а также поменяем тип данных на правильный.

Обработка столбца 'critic_score'

```
#critic_score
df['critic_score'] = df['critic_score'].fillna(-5).astype('int32')
```

Возможные причины пропусков в данных

Пропуски в столбце 'rating' могут быть связаны с тем, что их выствыляет именно АМЕРИКАНСКАЯ ассоциация 'ESRB', определяющая возрастной рейтинг компьютерных игр: она оценивает игры только в США и Канаде (информация из интернета: источник), поэтому, можно предположить, что все игры без рейтинга (с пропусками) были выпущенны где-то в других регионах. Пропуски в других столбцах скорее возникли в результате технических ошибок (например при выгрузке данных).

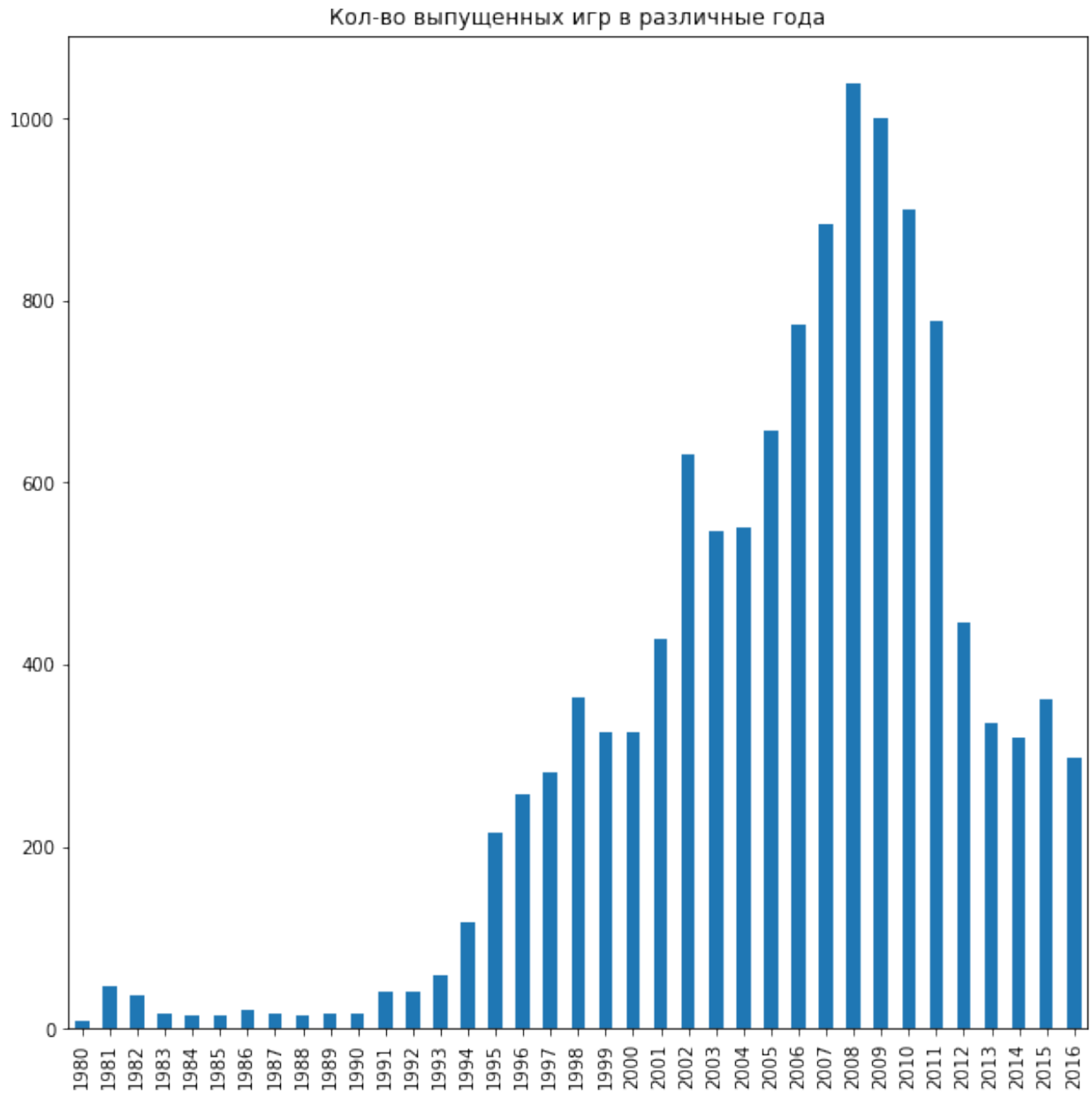
Добавление нового столбца

```
df['all_sales'] = df[['eu_sales','na_sales','other_sales',
'jp_sales']].sum(axis='columns')
```

Исловательский анализ данных

Кол-во выпущенных игр в различные года

```
pd.Series(
    data=[len(df[df['year_of_release']==year]['name'].unique()) for
year in sorted(df['year_of_release'].unique()) if year!=0],
    index=[year for year in sorted(df['year_of_release'].unique()) if
year != 0]
).plot(
    kind='bar',
    figsize=(10,10),
    title='Кол-во выпущенных игр в различные года')
plt.show()
```



Не все периоды времени будут актуальны для исследования.

Продажи 5-ти самых прибыльных платформ в различные года

```
for platform in df[df['year_of_release']!=0].groupby('platform')
['all_sales'].sum().sort_values(ascending=False).head().index:
    df[(df['platform'] == platform) & (df['year_of_release'] !=
0)].groupby('year_of_release')['all_sales'].sum().plot(
        kind='bar', title=f'Продажи платформы {platform} в различные
года.'
    )
    plt.show()
```


Продажи платформы PS2 в различные года.
200
175
150
125
100
75
50
25
0
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
year_of_release


Продажи платформы X360 в различные года.
160
140
120
100
80
60
40
20
0
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
year_of_release


Продажи платформы PS3 в различные года.
160
140
120
100
80
60
40
20
0
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
year_of_release


Продажи платформы Wii в различные года.
200
175
150
125
100
75
50
25
0
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
year_of_release

Игры лидирующие по продажам за все время

```
df.groupby('name')
['all_sales'].sum().sort_values(ascending=False).head()

name
Wii Sports              82.54
Grand Theft Auto V      56.58
Super Mario Bros.       45.31
Tetris                  35.84
Mario Kart Wii          35.52
Name: all_sales, dtype: float64
```

За какой характерный срок появляются новые и исчезают старые платформы?

По вышепоказанным графикам, можно предположить, что этот срок равен 10-тью годам; также важно уточнить, что этот срок не влияет на платформу 'PC', т.к. платформа не устаревает, из-за возможности обновлять ее комплектующие.

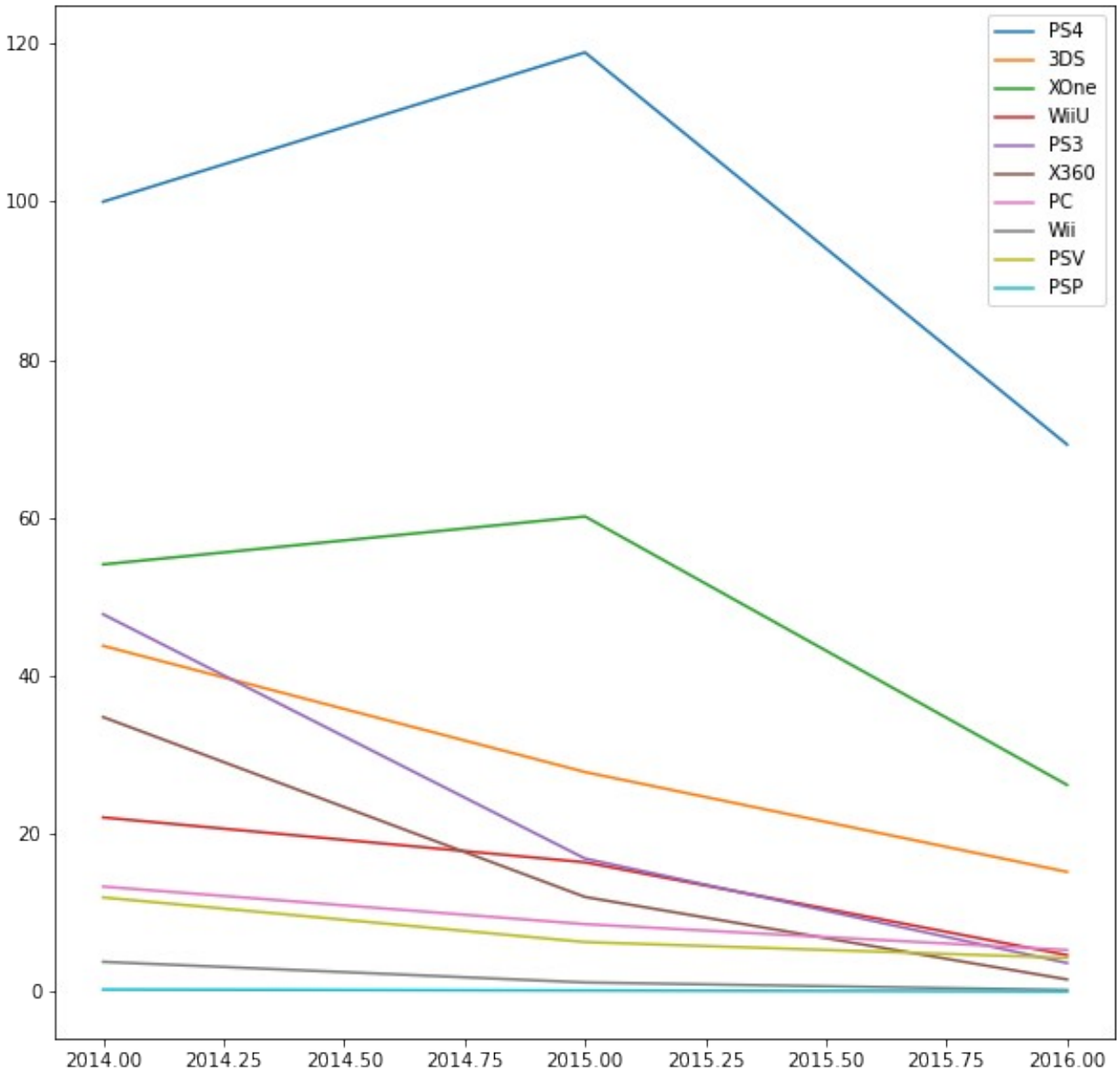
Определение актуального периода для анализа

Учитывая что компьютерные игры, относительно новая и динамично меняющийся индустрия, нужно взять не большой временной интервал, начиная с 2014 года.

```
df_actual = df[df['year_of_release'] >= 2014]
```

Перспективные платформы

```
for platform in df_actual['platform'].unique():

pd.Series(data=[df_actual[(df_actual['platform']==platform)&(df_actual
['year_of_release']==year)]['all_sales'].sum() \
                for year in
sorted(df_actual['year_of_release'].unique())],
        index=sorted(df_actual['year_of_release'].unique()),
        name=platform
    ).plot(
        figsize=(10,10),
        legend=True
    )
```




```
potential_platform = ['PS4', 'XOne', '3DS', 'PC']
```

Выберем платформы с наибольшими возможными продажами в 2017 году, учитывая, что данные за 2016 год не полные, и продажи каждой консоли в этом году еще могут вырасти.

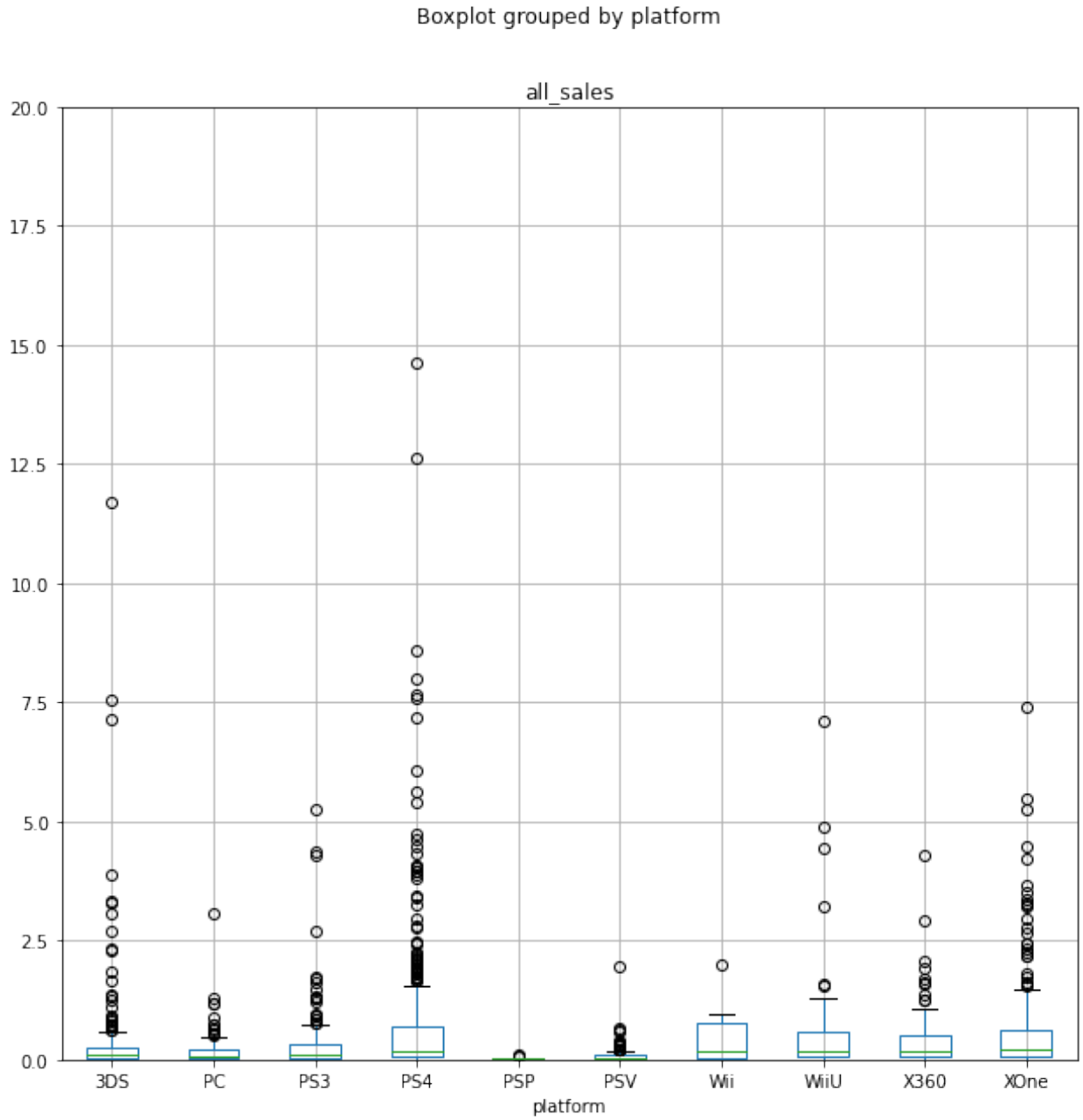
```
falling_platforms = [platform for platform in
df_actual['platform'].unique() if platform not in potential_platform]

pd.Series(
    data=[(2016-df[df['year_of_release']!=0].groupby('platform')
['year_of_release'].min()[platform]) for platform in
falling_platforms],
```

```
    index=falling_platforms
).plot(
    kind='bar',
    title='Срок существования каждой из консолей, чьи продажи падали.'
)
plt.show()
```

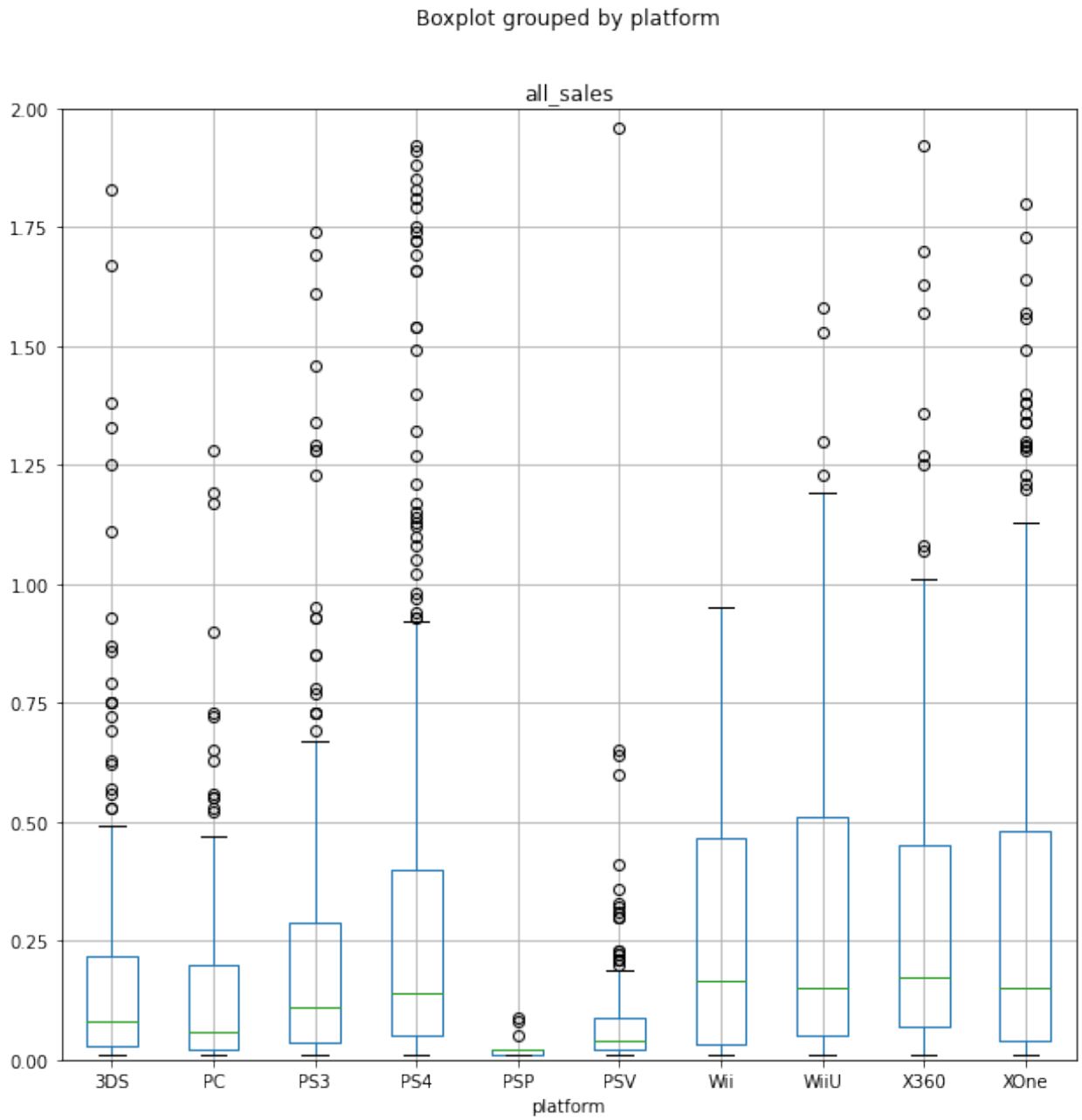
Получается, что платформы ''PS3'', 'X360', 'PSP' уже на рынке более 10 лет, следовательно сильно устарели, игры постепенно перестают на них выходить, и продажи начинают падать, а продажи платформ 'WiiU', 'PSV' практически не падают, а остаются стабильно низкими, скорее всего потому, что они являются портативными консолями, которые не сыскали популярности, как 'PSP', и востребованы в основном в регионе 'JP'.

Диаграммы размаха

```
df_actual[df_actual['all_sales'] <= 20].boxplot(column='all_sales',
by='platform', figsize=(10, 10))
plt.ylim(0, 20)
plt.show()
```




```
df_actual[df_actual['all_sales'] <= 2].boxplot(column='all_sales',
by='platform', figsize=(10, 10))
plt.ylim(0, 2)
plt.show()
```

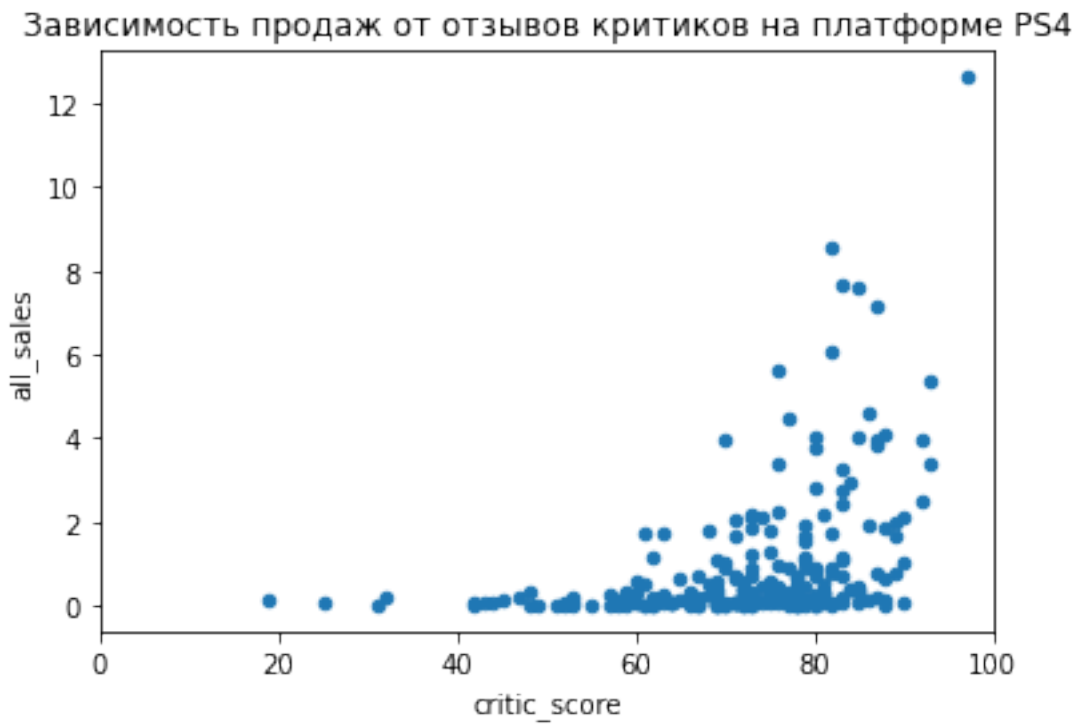


У всех платформ наблюдается множество выбросов, из-за чего продажи на всех платформах скошенны вправо (вверх на диаграмме размаха), это показывает, что независимо от платформы, часто появляются очень популярные игры, с большими продажами. А медианное значение у всех платформ находится в промежутке от 0 до 1/4 млн.

Влияние отзывов критиков и пользователей на продажи

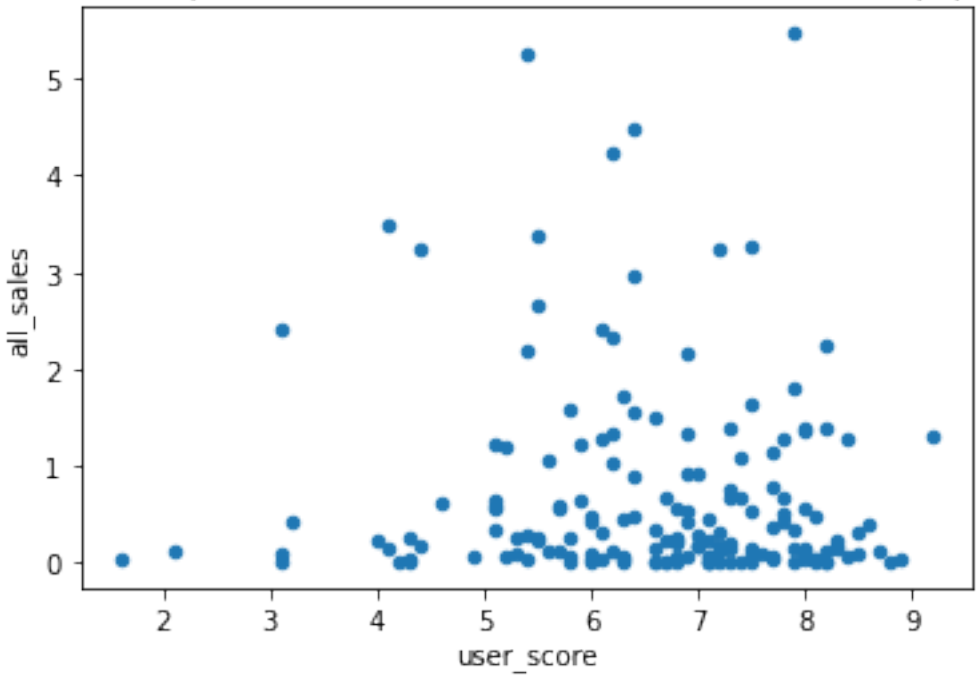
```
for platform in potential_platform:
    df_actual[(df_actual['platform']==platform)].plot(kind='scatter',
x='user_score', y='all_sales', \
```

```
        title=f'Зависимость продаж от отзывов пользователей на
платформе {platform}')
    plt.show()

df_actual[(df_actual['platform']==platform)&(df_actual['critic_score']
!=-5)].plot(kind='scatter', x='critic_score', y='all_sales', \
        title=f'Зависимость продаж от отзывов критиков на платформе
{platform}')
    plt.xlim(0,100)
    plt.show()
    for score in ['critic_score', 'user_score']:
        print(f"{score} - all_sales:
{df_actual.loc[(df_actual['platform']==platform)&(df_actual['critic_sc
ore']!=-
5),'all_sales'].corr(df_actual.loc[(df_actual['platform']==platform)&(
df_actual['critic_score']!=-5), score])}")
    for _ in range(5):
        print()
```

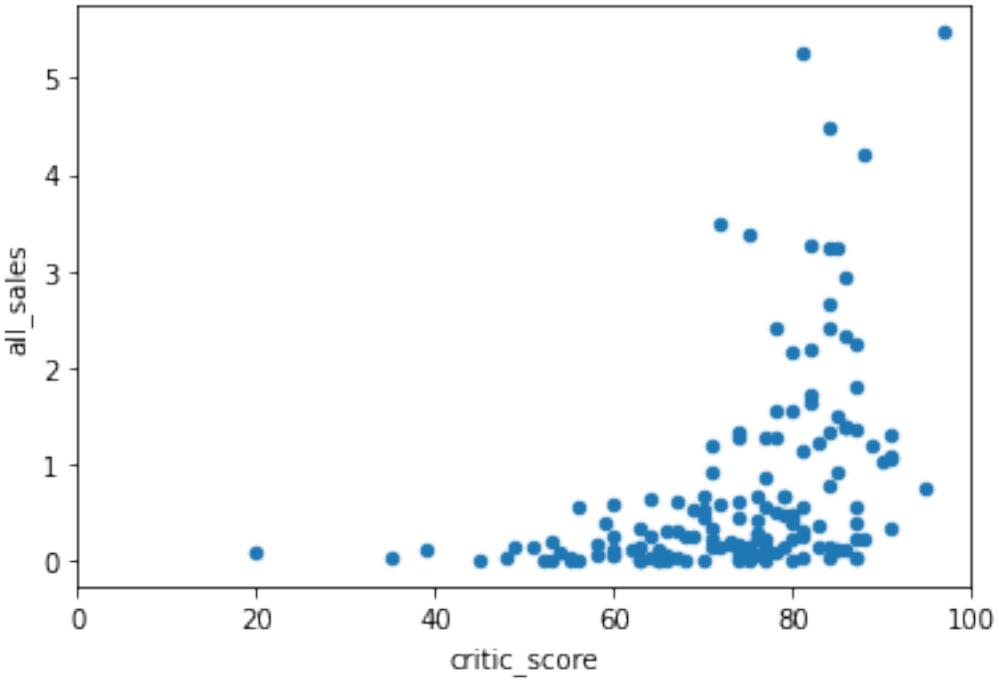



```
critic_score - all_sales: 0.40266141068104083
user_score - all_sales: -0.043185859969767905
```

Зависимость продаж от отзывов пользователей на платформе XOne

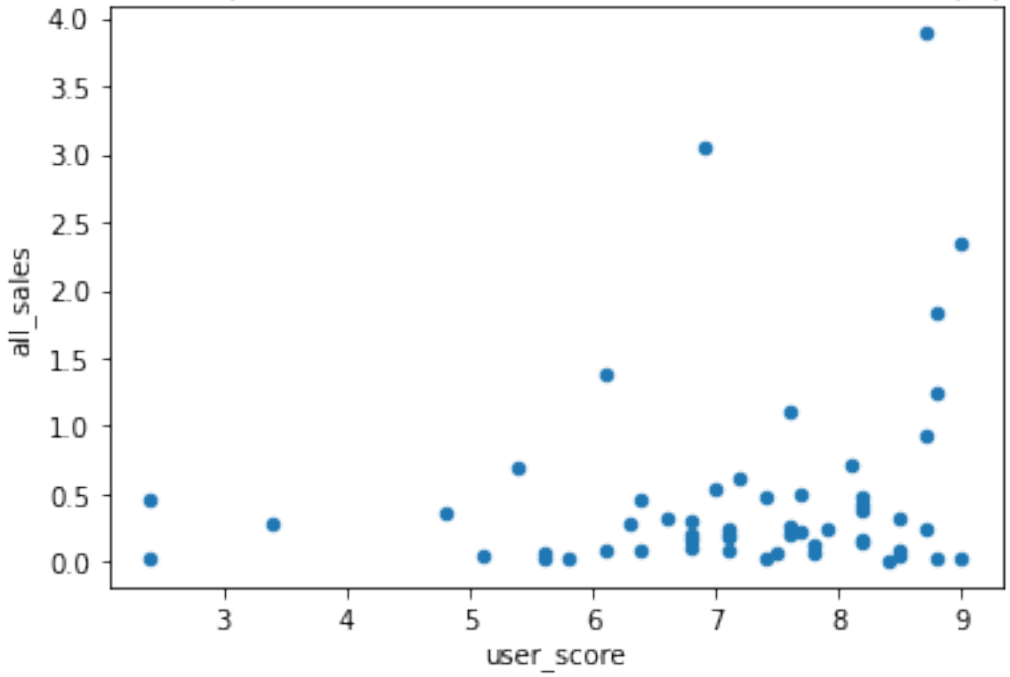


Зависимость продаж от отзывов критиков на платформе XOne

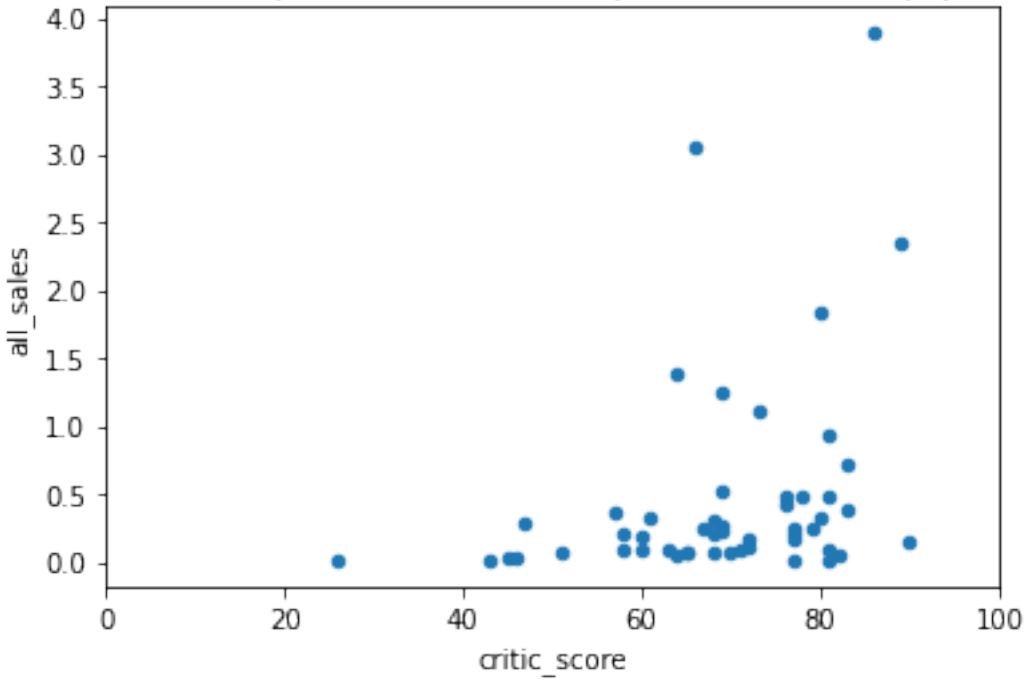



```
critic_score - all_sales: 0.42867694370333226
user_score - all_sales: -0.09173434626358005
```

Зависимость продаж от отзывов пользователей на платформе 3DS



Зависимость продаж от отзывов критиков на платформе 3DS



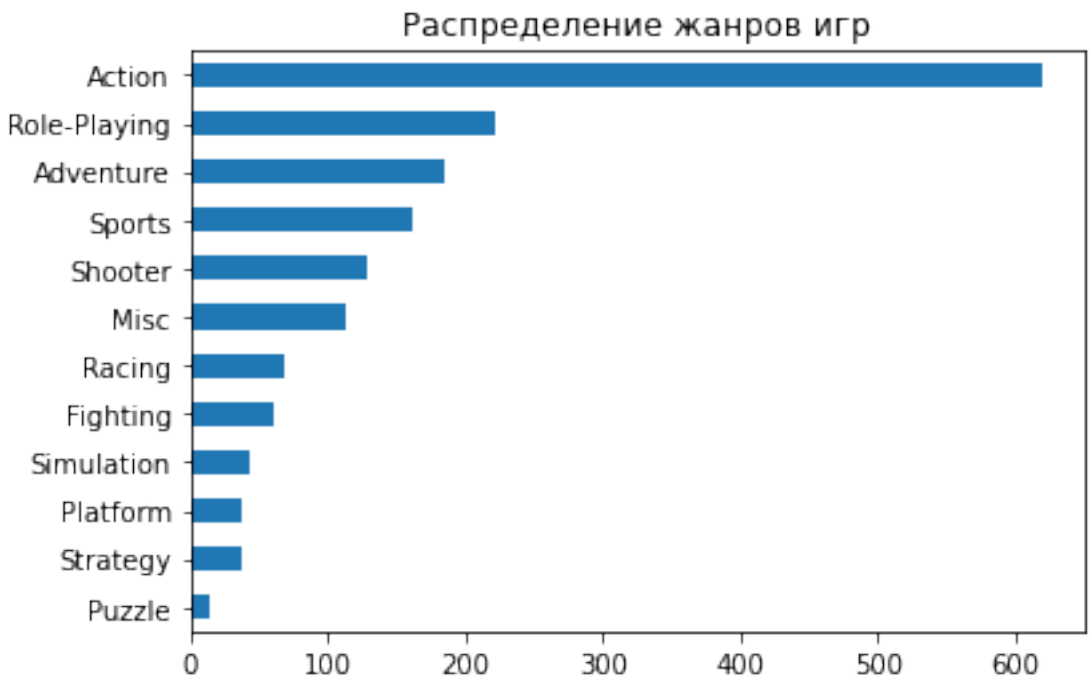
```
critic_score - all_sales: 0.31411749286905105
user_score - all_sales: 0.27984988973168434
```


```
critic_score - all_sales: 0.17463413769350036
user_score - all_sales: -0.08997391075956201
```

Корреляции продаж с рейтингами пользователей нет, однако в случае с рейтингами критиков, можно заметить, что при высокой оценке, начинают появлятся игры с высокими продажами, следовательно, игры с высокой оценкой критиков часто продаются лучше, чем игры с более низкой оценкой.

Влияние жанров

```
df_actual['genre'].value_counts().sort_values().plot(kind='barh',
title='Распределение жанров игр')
plt.show()
```



Самые популярные жанры Action, Role-Playing, Adventure.

```
df_actual.groupby('genre')
['all_sales'].median().sort_values().plot(kind='barh',
title='Медианная прибыль одной игры различных жанров')
plt.show()
```

Самые прибыльные жанры Shooter, Sports, Platform.

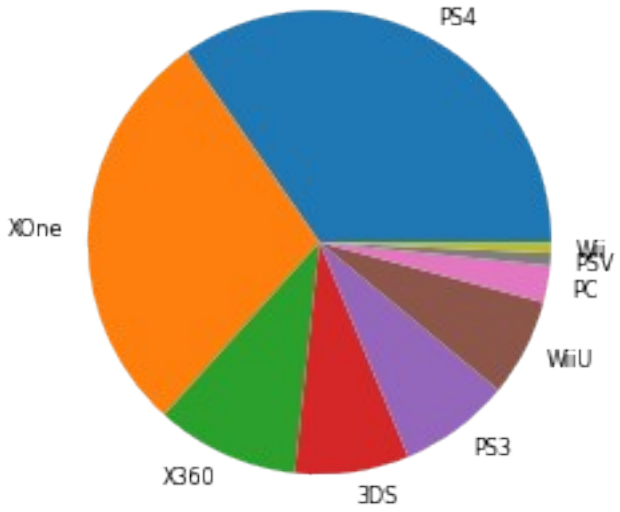
Портрет пользователей в разных регионах

```
def user_portrait_by_region(region):
        df_actual.groupby('platform')
[f'{region.lower()}_sales'].sum().sort_values(ascending=False).plot(
            kind='pie', fontsize=8, ylabel='', title='Круговая
диаграмма продаж с разбивкой по платформам')
        plt.show()
        print(f'Пять платформ лидирующих по продажам в регионе
{region}:')
        print(df_actual.groupby('platform')
[f'{region.lower()}_sales'].sum().sort_values(ascending=False).head(),
'\n')
        df_actual.groupby('genre')
[f'{region.lower()}_sales'].sum().sort_values(ascending=False).plot(
            kind='pie', fontsize=8, ylabel='', title='Круговая
диаграмма продаж с разбивкой по жанрам')
        plt.show()
        print(f'Пять жанров лидирующих по продажам в регионе
{region}:')
        print(df_actual.groupby('genre')
[f'{region.lower()}_sales'].sum().sort_values(ascending=False).head(),
'\n')
        df_actual.groupby('rating')
[f'{region.lower()}_sales'].sum().sort_values(ascending=False).plot(
            kind='pie', fontsize=8, ylabel='', title='Круговая
диаграмма продаж с разбивкой по возрастным рейтингам')
        plt.show()
        print(f'Пять возрастных рейтингов лидирующих по продажам в
регионе {region}:')
        print(df_actual.groupby('rating')
[f'{region.lower()}_sales'].sum().sort_values(ascending=False), '\n')
```

Портрет пользователя из Северной Америки

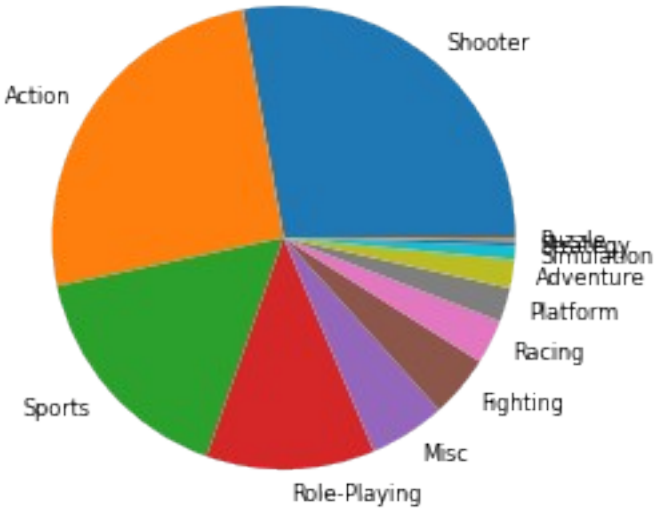
```
user_portrait_by_region('NA')
```

Круговая диаграмма продаж с разбивкой по платформам




```
Пять платформ лидирующих по продажам в регионе NA:
platform
PS4     98.61
XOne    81.27
X360    28.30
3DS     22.64
PS3     22.05
Name: na_sales, dtype: float64
```

Круговая диаграмма продаж с разбивкой по жанрам



```
Пять жанров лидирующих по продажам в регионе NA:
genre
Shooter         79.02
Action          72.53
Sports          46.13
Role-Playing    33.47
Misc            15.05
Name: na_sales, dtype: float64
```

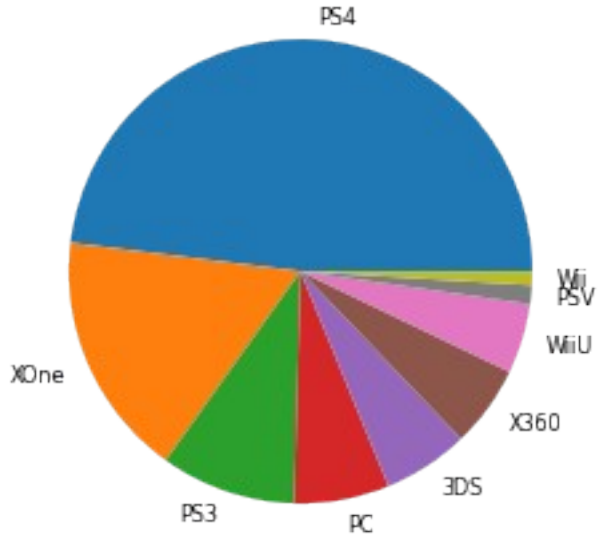
Круговая диаграмма продаж с разбивкой по возрастным рейтингам


```
Пять возрастных рейтингов лидирующих по продажам в регионе NA:
rating
M        96.42
undef    64.72
E        50.74
T        38.95
E10+     33.23
Name: na_sales, dtype: float64
```

Портрет пользователя из Европы

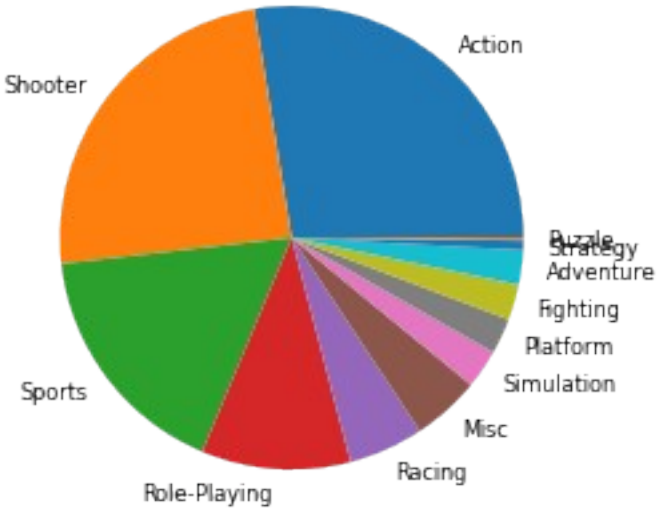
```
user_portrait_by_region('EU')
```

Круговая диаграмма продаж с разбивкой по платформам




```
Пять платформ лидирующих по продажам в регионе EU:
platform
PS4     130.04
XOne     46.25
PS3      25.54
PC       17.97
3DS      16.12
Name: eu_sales, dtype: float64
```

Круговая диаграмма продаж с разбивкой по жанрам



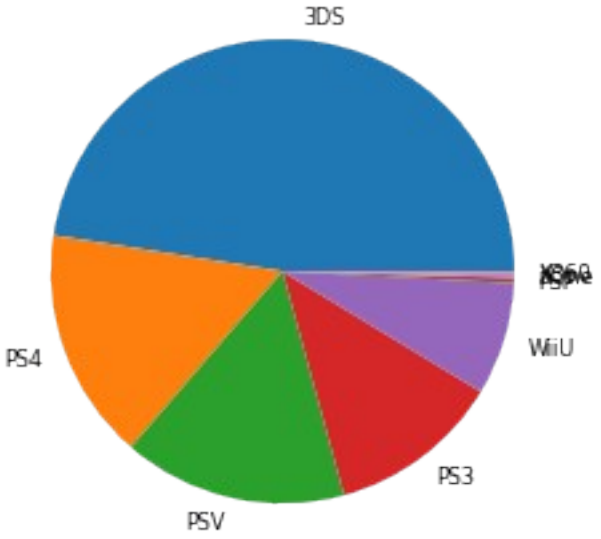
```
Пять жанров лидирующих по продажам в регионе EU:
genre
Action          74.68
Shooter         65.52
Sports          45.73
Role-Playing    28.17
Racing          14.13
Name: eu_sales, dtype: float64
```


```
Пять возрастных рейтингов лидирующих по продажам в регионе EU:
rating
M        93.44
undef    58.95
E        58.06
T        34.07
E10+     26.16
Name: eu_sales, dtype: float64
```

Портрет пользователя из Японии

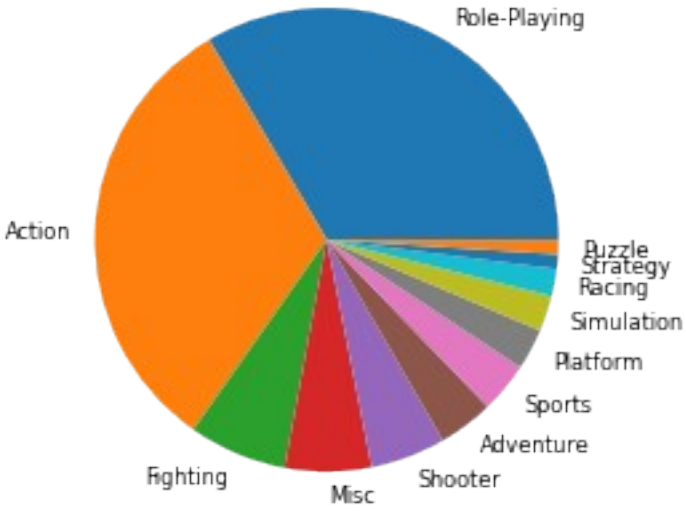
```
user_portrait_by_region('JP')
```

Круговая диаграмма продаж с разбивкой по платформам

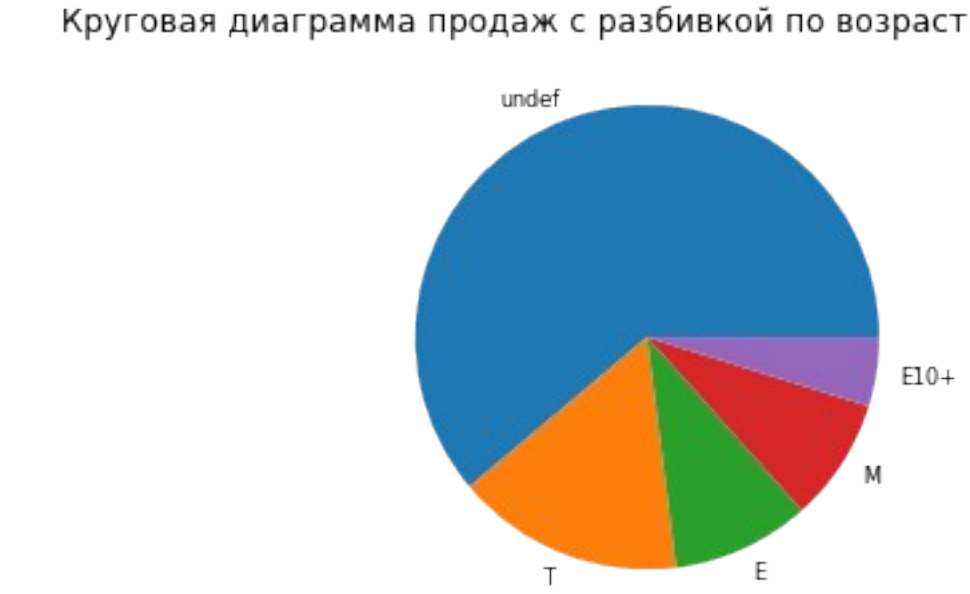



```
Пять платформ лидирующих по продажам в регионе JP:
platform
3DS     44.24
PS4     15.02
PSV     14.54
PS3     11.22
WiiU     7.31
Name: jp_sales, dtype: float64
```

Круговая диаграмма продаж с разбивкой по жанрам



```
Пять жанров лидирующих по продажам в регионе JP:
genre
Role-Playing    31.16
Action          29.58
Fighting         6.37
Misc             5.61
Shooter          4.87
Name: jp_sales, dtype: float64
```




```
Пять возрастных рейтингов лидирующих по продажам в регионе JP:
rating
undef    56.90
T        14.78
E         8.94
M         8.01
E10+      4.46
Name: jp_sales, dtype: float64
```

Вывод из анализа портретов пользователей из различных регионов

- **Во всех регионах популярны 'PS4', '3DS', 'PS3'. В регионах 'NA' и 'EU' особо популярна 'XOne'. В регионе 'EU' также популярна 'PC', в регионе 'JP' - платформа 'WiiU' и 'PSV', а в регионе 'NA' популярна платформа 'X360'.**

- **Во всех регионах популярны жанры 'Action', 'Role-Playing', 'Shooter'. В регионах 'NA' и 'EU' популярен жанр 'Sports'. В регионах 'NA' и 'JP' популярен жанр 'Misc'. В регионе 'EU' популярен жанр 'Racing', а в регионе 'JP' - жанр 'Fighting'.**
- **В регионах 'NA' и 'EU' больше всего играми увлекаются взрослые, а в 'JP' - подростки. Также, если учитывать гипотезу о происхождении пропусков в столбце 'rating', исходи из доли можно предположить что в регионах 'NA' и 'EU' в основном предпочитают игры выпущенные в Северной Америке, а в регионе 'JP' - игры из других регионов, скорее всего отечественного производства.**

Проверка гипотез

Средние пользовательские рейтинги платформ 'Xbox One' и 'PC' одинаковые

```
#нулевая гипотеза - Средние пользовательские рейтинги платформ Xbox One и PC одинаковые
#альтернативная - Средние пользовательские рейтинги платформ Xbox One и PC разные
alpha = .05
result = st.ttest_ind(df_actual.loc[(df_actual['platform'] == 'PC')&(~df_actual['user_score'].isna()), 'user_score'], \
    df_actual.loc[(df_actual['platform'] == 'XOne')&(~df_actual['user_score'].isna()), 'user_score'], equal_var=False)
print('pvalue = {:.1%}'.format(result.pvalue))
if result.pvalue < alpha:
    print('Нулевую гипотезу следуют отбросить')
else:
    print('Нулевую гипотезу отбросить не получится')
```

```
pvalue = 11.6%
Нулевую гипотезу отбросить не получится
```

У нас нет оснований утверждать, что средние пользовательские рейтинги игр, выходивших на платформах 'Xbox One' и 'PC', значимо различаются.

Средние пользовательские рейтинги жанров 'Action' и 'Sports' разные

```
#нулевая гипотеза - Средние пользовательские рейтинги жанров Action и Sports равны..
#альтернативная - Средние пользовательские рейтинги жанров Action и Sports разные.
alpha = .05
```

```
result = st.ttest_ind(df_actual.loc[(df_actual['genre'] ==
'Action')&(~df_actual['user_score'].isna()), 'user_score'], \
    df_actual.loc[(df_actual['genre'] ==
'Sports')&(~df_actual['user_score'].isna()), 'user_score'],
equal_var=False)
print('pvalue = {:.13%}'.format(result.pvalue))
if result.pvalue < alpha:
    print('Нулевую гипотезу следуют отбросить')
else:
    print('Нулевую гипотезу отбросить не получится')
```

```
pvalue = 0.0000000000012%
Нулевую гипотезу следуют отбросить
```

У нас нет оснований утверждать, что средние пользовательские рейтинги игр, принадлежащих к жанрам 'Action' и 'Sports', равны.

Пояснение способа проверки гипотез

Составим гипотезы с упором на то, что в нулевой стоит проверять равенство, затем проверим их с помощью t-теста, с помощью библиотеки scipy, уровень статистической значимости зададим 5-тью процентами, а т.к. нельзя утверждать равенство генеральных выборок, параметр 'equal_var' зададим значением False.

Общий вывод

- Датафрейм представляет собой набор данных о играх продающихся до 2016 года. Названия столбцов были приведены к нижнему регистру. Пропуски в столбце с оцеками критиков, возрастными рейтингами, годами выпуска и названиями жанра были заменены на значения-заглушки, в столбце с оценками пользователей - оставлены, строки с пропусками в столбце с названиями игр - удалены. В некоторых столбцах был изменен тип данных на правильный. В столбце с возрастными рейтингами некоторые малочисленные значения были заменены на более распространенные или были удалены. Выдвинуто предположение о возникновение пропусков в том же столбце, связанное с тем, где, асоциация, определяющая возрастной рейтинг компьютерных игр, предоставляет свои услуги. Был добавлен новый столбец с суммарными продажами для каждой игры.
- Был сделан вывод, что не все данные будут актуальны для анализа. Был выбран срок актуальности платформ в 10 лет. Актуальный период был выбран с 2014 по 2016 года. Были выявлены перспективные платформы, исходя из графика продаж в акутуальный период. Были выявлены причины падения продаж

некоторых платформ. Была выявленна прямая зависимость продаж от рейтинга критиков. Были выявлены самые прибыльные жанры. Для каждого региона был составлен портрет пользователя (самые прибыльные платформы, жанры, возрастные рейтинги). Были проверены некоторые гипотезы.

- В 2017 году следует обратить внимание на игры выходящие на платформах 'PC', 'PS4', 'XOne', '3DS'. На платформах 'PC', 'PS4', 'XOne' стоит обратить внимание на игры в жанрах 'Shooter', 'Action', 'Sports', и имеющих возрастной рейтинг 'M'. А на платформе '3DS' - 'Action', 'Role-Playing', 'Fighting'; возрастной рейтинг - 'T'. Также необходимо выбирать только, те игры, которые перед выпуском, получали высокие оценки критиков.

Пояснение

Платформы 'PC', 'PS4', 'XOne' популярны в регионах 'NA' и 'EU', поэтому, подбирая лучшие жанры и возрастной рейтинг для этих платформ, я опирался на портрет пользователей в этих регионах. Аналогично '3DS' - популярна в регионе 'JP',возрастной рейтинг и жанры подобранны по портрету пользователя.