Выделение определяющие успешность игр закономерности для планирования рекламной компании "Стримчик".

Входные данные до 2016 года. Интересующий момент времени - декабрь 2016 г., и планируется кампанию на 2017-й.
В наборе данных попадается аббревиатура ESRB (Entertainment Software Rating Board) — это ассоциация, определяющая
возрастной рейтинг компьютерных игр. ESRB оценивает игровой контент и присваивает ему подходящую возрастную категорию:

- «Для взрослых»
- «Для детей младшего возраста»
- «Для подростков».

План

1. Получение файлов с данными и изучиние общей информации

- Описание данных
- Получение данных и изучение общей информации

2. Подготовка данных

- Приведите данные к корректным наименованиям
- Исправление пропусков и выбросов в данных
- Добавление информации

3. Анализ данных

- Анализ игр и продаж.
- Выделение периода для прогноза на 2017 год.
- Анализ по платформам
- Анализ по жанрам

4. Составление портрета пользователя каждого региона

- Топ-5 платформ
- Топ-5 жанров
- Влияние рейтинга ESRB на продажи

5. Проверка гипотез

---

1. Получение файлов с данными и изучение общей информации

Описание данных

Таблица **users** (информация о пользователях):

- **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 (Entertainment Software Rating Board).
  - `E` - Everyone
  - `EC` - Early childhood
  - `E10+` - Everyone 10 and older
  - `M` - Mature (> 17 y o)
  - `T` - Tean (13 - 19 y o)
  - `AO` - Adults only 18 +
  - `RP` - Rating Pending
  - `K-A` - Kids to Adults

Данные за 2016 год могут быть неполными.

Получение данных и изучение общей информации

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as st
import numpy as np
import os
from pathlib import Path
import urllib
import plotly.express as px
import scipy.stats as st

pd.options.display.float_format = '{:.3f}'.format
%config InlineBackend.figure_format='retina'
```

In [2]:
```
def get_file(name, url):
    if not os.path.exists(name):
        print(name, 'не найден. Будет загружен из сети')
        try:
            _ = urllib.request.urlretrieve(url, name)
        except:
            print('Проблема с загрузкой из сети')

files = { 'games': ('datasets/games.csv', 'https://code.s3.yandex.net/datasets/games.cs

Path('datasets').mkdir(parents=True, exist_ok=True)
for key in files:
    get_file(*files[key])
```

In [3]:
```
data = pd.read_csv('datasets/games.csv')
data.info()
display(data.sample(3))
data.describe().T
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16715 entries, 0 to 16714
Data columns (total 11 columns):
Name               16713 non-null object
Platform           16715 non-null object
Year_of_Release    16446 non-null float64
Genre              16713 non-null object
NA_sales           16715 non-null float64
EU_sales           16715 non-null float64
JP_sales           16715 non-null float64
Other_sales        16715 non-null float64
Critic_Score       8137 non-null float64
User_Score         10014 non-null object
Rating             9949 non-null object
dtypes: float64(6), object(5)
memory usage: 1.4+ MB
```

	Name	Platform	Year_of_Release	Genre	NA_sales	EU_sales	JP_sales	Other_sales	Critic_Scor
5855	Lumines II	PSP	2006.000	Puzzle	0.120	0.120	0.000	0.070	na
407	Star Wars Episode III: Revenge of the Sith	PS2	2005.000	Action	1.470	1.390	0.030	0.430	60.00
8848	Dark Rift	N64	1997.000	Fighting	0.120	0.030	0.000	0.000	na

Out[3]:

	count	mean	std	min	25%	50%	75%	max
Year_of_Release	16446.000	2006.485	5.877	1980.000	2003.000	2007.000	2010.000	2016.000
NA_sales	16715.000	0.263	0.814	0.000	0.000	0.080	0.240	41.360
EU_sales	16715.000	0.145	0.503	0.000	0.000	0.020	0.110	28.960
JP_sales	16715.000	0.078	0.309	0.000	0.000	0.000	0.040	10.220
Other_sales	16715.000	0.047	0.187	0.000	0.000	0.010	0.030	10.570

	count	mean	std	min	25%	50%	75%	max
Critic_Score	8137.000	68.968	13.938	13.000	60.000	71.000	79.000	98.000

In [4]:
```
data.duplicated().sum()
```

Out[4]:
```
0
```

Вывод

Входные данние имеют пропуски - в большей мере относится к данным о рейтинге и отзывов критиков и игроков.
В меньшей мере к остальным характеристикам. Год выпуска хранится в качестве дробного значения. Также увидели,
что лишь меньшенство рассматриваемых игр были выпущены позднее 2010 года. Дубликатов в данных нет.

---

2. Подготовка данных

Приведите данные к корректным наименованиям

In [5]:
```
def describe_updown(data, list_cols=False):
    '''Поиск значений low_iqr и up_iqr для солбцов в list_cols DataFrame data
    и занесений значений в DataFrame метода .describe()'''
    def get_lowest_uppest(col):
        '''Получение нижнего и верхнего "усов" данных - то,
        что будет добавлено в DataFrame метода .describe()'''
        col_info = dict(col.describe())
        if col_info.get('75%', None) is None:
            return None
        iqr = col_info['75%'] - col_info['25%']
        lowest = col_info['25%'] - 1.5 * iqr
        lowest = lowest if lowest >= 0 else 0
        uppest = col_info['75%'] + 1.5 * iqr
        return lowest, uppest

    if list_cols is False:
        list_cols = data.columns
    cols_to_add = []
    descr = pd.DataFrame(data[list_cols].describe())
    lowers, uppers = [], []
    for item in list_cols:
        temp = get_lowest_uppest(data[item])
        if temp:
            cols_to_add.append(item)
            lowers.append(temp[0])
            uppers.append(temp[1])
    to_add = pd.DataFrame([lowers, uppers], index=(('low_iqr', 'up_iqr')))
    to_add.columns = cols_to_add
    descr = descr.append(to_add)
    return descr

def emptiness_col(data, col):
    to_print = data[col].isna().sum() / data.shape[0]
    print(col, 'column contains {: .3%} empty values'.format(to_print))
```

```
    if to_print == 0:
        print('There is no empty values in', col)
    elif to_print < .02:
        print('Discard all empty values')
        data = data.dropna(subset=[col])
    else:
        print('Can\'t discard them')
    return data

def del_anomal_values(data, info_descr, list_cols):
    for col in list_cols:
        low, up = info_descr[col]['low_iqr'], info_descr[col]['up_iqr']
        data = data[(low < data[col]) & (data[col] < up)]
    return data
```

Исправление пропусков и выбросов в данных

In [6]:
```
print('#' * 50)
for col in data.columns:
    data = emptiness_col(data, col)
    print('#' * 50)
```

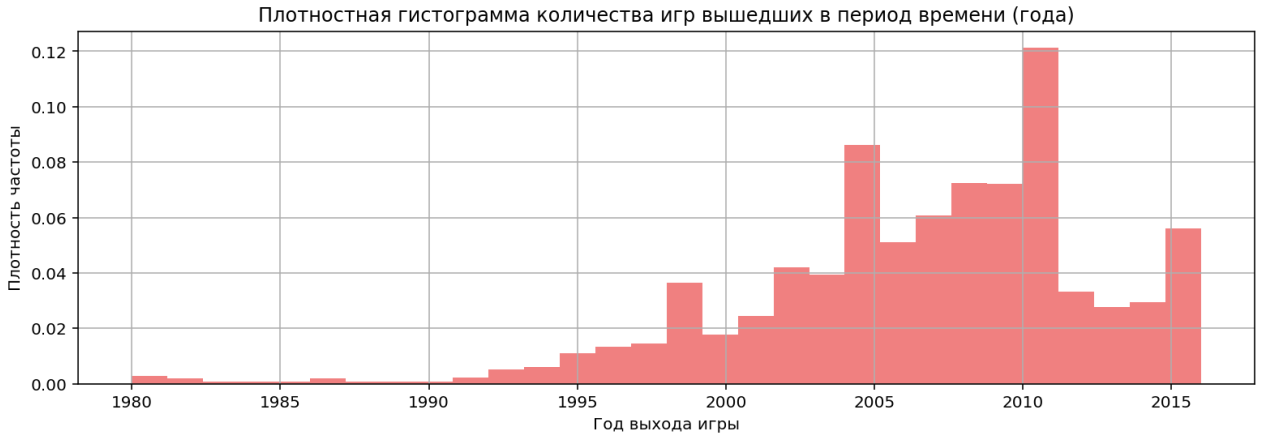
```
##################################################
Name column contains  0.012% empty values
Discard all empty values
##################################################
Platform column contains  0.000% empty values
There is no empty values in Platform
##################################################
Year_of_Release column contains  1.610% empty values
Discard all empty values
##################################################
Genre column contains  0.000% empty values
There is no empty values in Genre
##################################################
NA_sales column contains  0.000% empty values
There is no empty values in NA_sales
##################################################
EU_sales column contains  0.000% empty values
There is no empty values in EU_sales
##################################################
JP_sales column contains  0.000% empty values
There is no empty values in JP_sales
##################################################
Other_sales column contains  0.000% empty values
There is no empty values in Other_sales
##################################################
Critic_Score column contains  51.453% empty values
Can't discard them
##################################################
User_Score column contains  40.167% empty values
Can't discard them
##################################################
Rating column contains  40.598% empty values
Can't discard them
##################################################
```

In [7]:
```
data.columns = [col.lower() for col in data.columns]
```

year_of_release

In [8]:
```
data.loc[:, 'year_of_release'] = data.year_of_release.astype(np.uint16)
```

In [9]:
```
# fig, axises = plt.subplots(1, 2)

data.year_of_release.plot(kind='hist', density=True, bins=30,\
                          grid=True, figsize=(13, 4), color='lightcoral')# , #ax=axises
plt.title('Плотностная гистограмма количества игр вышедших в период времени (года)')
plt.xlabel('Год выхода игры')
plt.ylabel('Плотность частоты')
plt.show()
```



platform

In [10]:
```
data.groupby('platform')['name'].count().plot(kind='bar', figsize=(15, 5), grid=True, c
plt.title('Диаграмма количества игр вышедших на соответствующие платформы')
plt.xlabel('Платформа')
plt.ylabel('Количество продагных игр')
plt.show()
```

genre

In [11]:
```
ax = data.groupby('genre')['name'].count().plot(kind='bar', figsize=(15, 5), grid=True,
plt.title('Диаграмма кол-ва игр вышедших по жанрам')
plt.xlabel('Жанр')
plt.ylabel('Количество проданых игр (млн копий)')
plt.show()
```

critic_score

In [12]:
```
data.loc[:, 'critic_score'] = data.critic_score.fillna(
    data.name.map(
        data.groupby('name').critic_score.first()
    )
)
```

In [13]:
```
print('critic_score column contains {: .3%} empty values'.format(data.critic_score.isna
```

```
critic_score column contains  45.743% empty values
```

In [14]:
```
data.loc[:, 'critic_score'] = data.critic_score.fillna(-1).astype(np.int8)
```

user_score

In [15]:
```
data.loc[:, 'user_score'] = data.user_score.fillna(
    data.name.map(
        data.groupby('name').user_score.first()
    )
)
```

In [16]:
```
print('user_score column contains {: .3%} empty values'.format(data.user_score.isna().s
```

```
user_score column contains  37.704% empty values
```

In [17]:
```
# data.loc[data.user_score == 'tbd', ['user_score']] = '-.2'
# data.loc[data.user_score.isna(), ['user_score']] = '-.1'

data.loc[(data.user_score.isna()) | (data.user_score == "tbd"), ['user_score']] = '-.1'
```

In [18]:
```
data.loc[:, 'user_score'] = (data.user_score.astype(float) * 10).astype(np.int8)
```

rating

In [19]:
```
data.groupby('rating')['name'].count().plot(kind='bar', figsize=(15, 5), grid=True, col
plt.title('Диаграмма кол-ва игр вышедших по рейтингу ESRB')
plt.xlabel('Рейтинг ESRB')
plt.ylabel('Количество проданых игр (млн копий)')
plt.show()
```

In [20]:
```
data.groupby('rating')['name'].count()
```

Out[20]:
```
rating
AO         1
E       3921
E10+    1393
EC         8
K-A        3
M       1536
RP         1
T       2905
Name: name, dtype: int64
```

In [21]:
```
data.loc[:, 'rating'] = data.rating.fillna(
    data.name.map(
        data.groupby('name').rating.first()
    )
)
```

In [22]:
```
print('rating column contains {: .3%} empty values'.format(data.rating.isna().sum() / d
```

```
rating column contains  38.063% empty values
```

In [23]:
```
data.loc[:, 'rating'] = data.rating.fillna('unknown')
```

Добавление информации

In [24]:
```
data.loc[:, 'total_sales'] = data.na_sales + data.eu_sales + data.jp_sales + data.other
sales_cols = ['na_sales', 'eu_sales', 'jp_sales', 'other_sales', 'total_sales']
sales_info = describe_updown(data, sales_cols).T
sales_info
```

Out[24]:

	count	mean	std	min	25%	50%	75%	max	low_iqr	up_iqr
na_sales	16444.000	0.264	0.818	0.000	0.000	0.080	0.240	41.360	0.000	0.600
eu_sales	16444.000	0.146	0.507	0.000	0.000	0.020	0.110	28.960	0.000	0.275
jp_sales	16444.000	0.078	0.311	0.000	0.000	0.000	0.040	10.220	0.000	0.100
other_sales	16444.000	0.048	0.188	0.000	0.000	0.010	0.030	10.570	0.000	0.075
total_sales	16444.000	0.536	1.559	0.000	0.060	0.170	0.470	82.540	0.000	1.085

Вывод

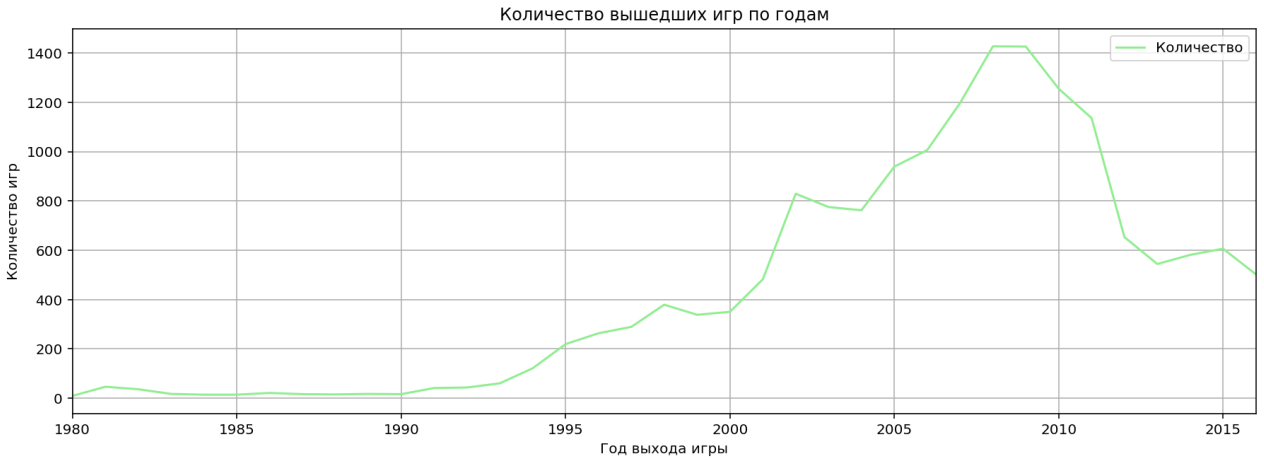
Были обработаны пропуски в данных (менее 2% пропусков были удалены),
В столбцах `critic_score` , `user_score` , `rating` пропуски достигали до 50%,
С помощью анализа смежных платформ (по названию игры) была заполнена мизерная часть данных, остальные пропуски были заоплнены маркерами (`-1` и `unknown`)

---

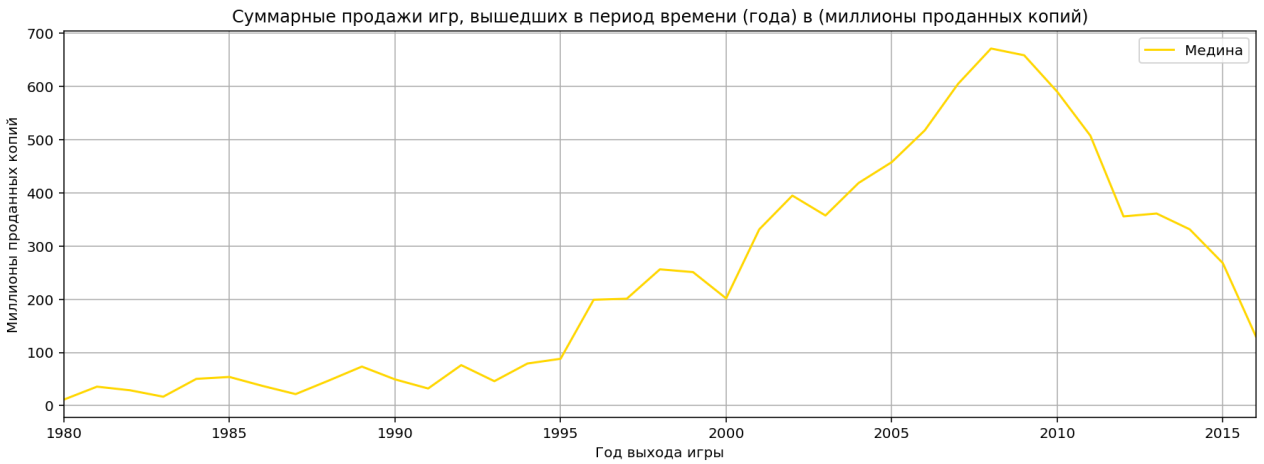
3. Анализ данных

Анализ игр и продаж.

In [25]:
```
ax = data.groupby('year_of_release').count().reset_index().plot(figsize=(15, 5), color=
                                                    x='year_of_release', y='total_sales', l

plt.title('Количество вышедших игр по годам')
plt.xlabel('Год выхода игры')
plt.ylabel('Количество игр')
plt.show()
```



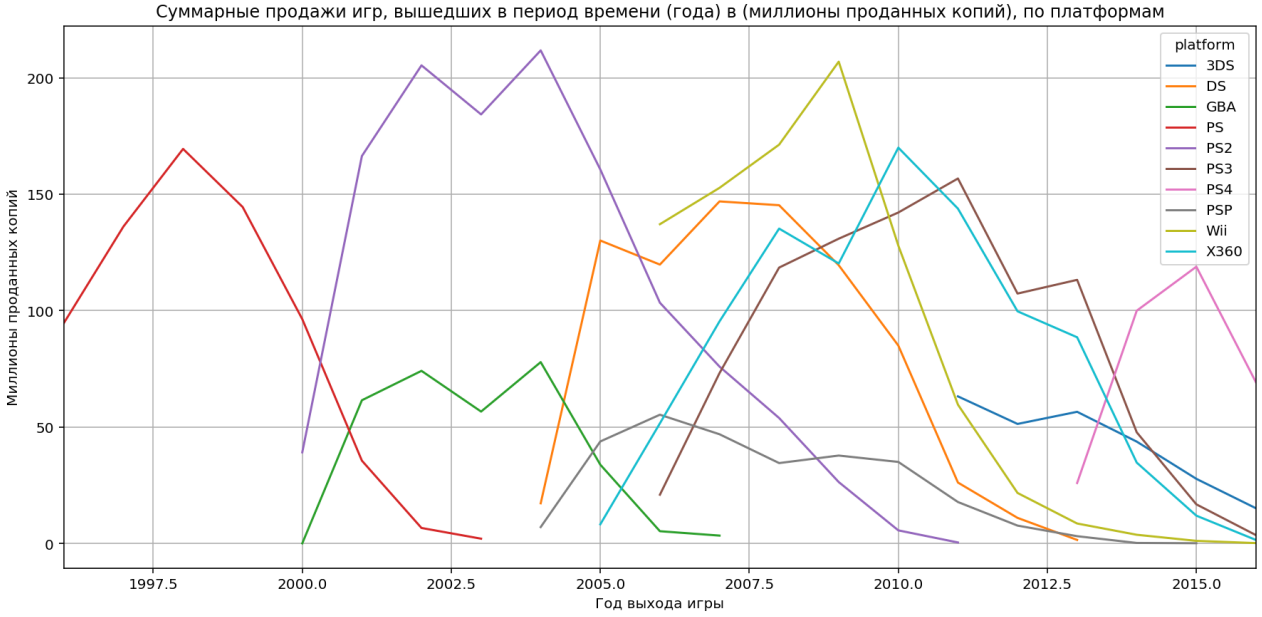
In [26]:
```
data.groupby('year_of_release').sum().reset_index().plot(grid=True, figsize=(15, 5), co
                                                    x='year_of_release', y='total_sales', l

plt.title('Суммарные продажи игр, вышедших в период времени (года) в (миллионы проданны
plt.xlabel('Год выхода игры')
plt.ylabel('Миллионы проданных копий')
plt.show()
```



Данные ранее 95 года можно не рассматривать, из-за нерентабельности в наст момент (очень маленькие продажи, что значит, что мало покупателей, которые даже просто "помнят" игры из дестства)

In [27]:
```
data = data.query('year_of_release > 1995')
```

In [28]:
```
top_platforms_sales = (data
    .groupby('platform').sum()
    .sort_values(by='total_sales', ascending=False).head(10)
    .reset_index()['platform']
)
top_platforms_sales = list(top_platforms_sales)

(data
    .query('platform in @top_platforms_sales')
    .pivot_table(index='platform', values='total_sales',\
                 columns='year_of_release', aggfunc='sum').T
    .plot(grid=True, figsize=(15, 7))
)

plt.title('Суммарные продажи игр, вышедших в период времени (года) в (миллионы проданны
plt.xlabel('Год выхода игры')
plt.ylabel('Миллионы проданных копий')
plt.show()
```



In [29]:
```
top_platforms_duration = (data
    .pivot_table(index='platform', values='year_of_release', aggfunc=['min', 'max']).re
)

top_platforms_duration.columns = 'name', 'start', 'end'
top_platforms_duration.loc[:, 'duration'] = top_platforms_duration.end - top_platforms_
top_platforms_duration.duration.plot(kind='bar', figsize=(15, 4), grid=True, color='whe
plt.title('Диагрмма "продолжительности жизни" платформ')
plt.xlabel('Величина "продолжительности жизни" платформы в годах')
plt.ylabel('Количество платформ')
plt.show()
```

Большинство платформ живёт максимум 8 лет

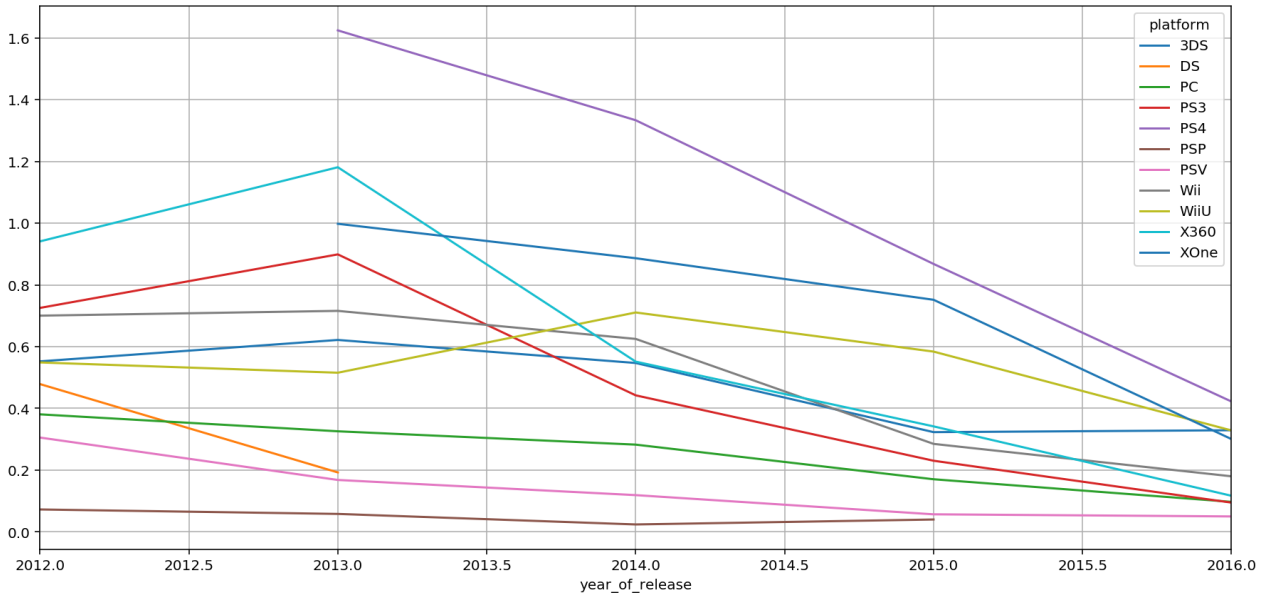
Выделение периода для прогноза на 2017 год

In [30]:
```
data_n = data.query('year_of_release > 2011') # n - newest
```

Выделем промежуток с 2011 года для прогназирования на 2017 год

Анализ по платформам

In [31]:
```
(data_n
    .pivot_table(index='year_of_release', values='total_sales', columns='platform')
    .plot(figsize=(15, 7), grid=True)
)

plt.show()
```



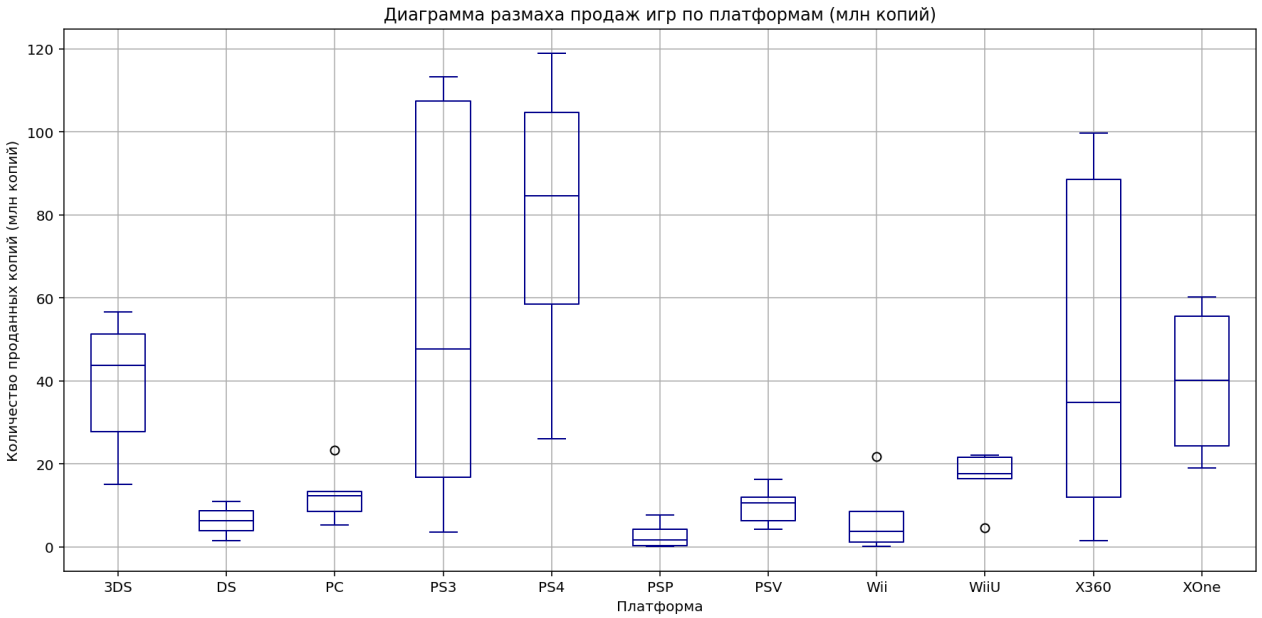
In [32]:
```
data_nn = (data
           .query('year_of_release > 2014')
           .pivot_table(index='year_of_release', values='total_sales', columns='platfor
)

data_nn.columns = ['name', '2015', '2016']
data_nn.loc[:, 'delta'] = data_nn.apply(lambda row: row['2016'] - row['2015'], axis=1)
data_nn = data_nn.dropna()[['name', '2016', 'delta']].sort_values(by='2016', ascending=
data_nn.columns = ['platform', 'sales', 'growth']
display(data_nn)
```

	platform	sales	growth
0	PS4	0.422	-0.446
1	3DS	0.329	0.006
2	WiiU	0.329	-0.255
3	XOne	0.301	-0.451
4	Wii	0.180	-0.105
5	X360	0.117	-0.225
6	PC	0.097	-0.073
7	PS3	0.095	-0.136
8	PSV	0.050	-0.007

Наиболее привлекательные платформы - PS4 и 3DS, у которых максимальное кол-во продаж и положительная динамика продаж соответственно

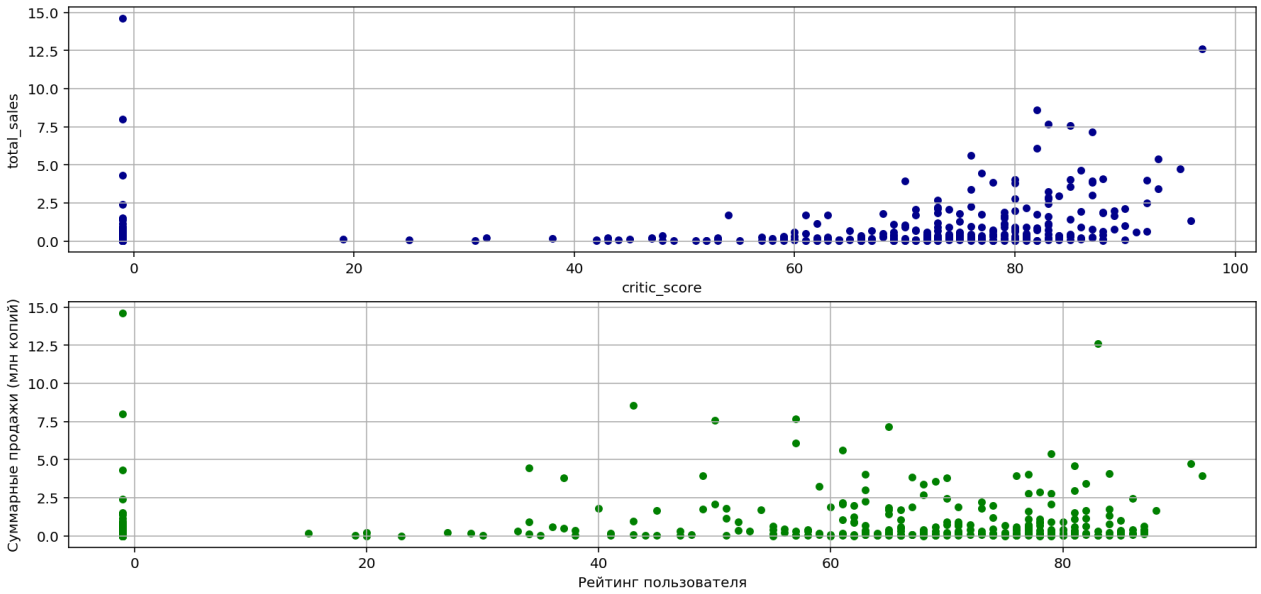
In [33]:
```
(data_n
    .pivot_table(index='year_of_release', values='total_sales', columns='platform', agg
    .plot(kind='box', figsize=(15, 7), grid=True, color='darkblue')
)

plt.title('Диаграмма размаха продаж игр по платформам (млн копий)')
plt.xlabel('Платформа')
plt.ylabel('Количество проданных копий (млн копий)')
plt.show()
```

```
/opt/conda/lib/python3.7/site-packages/numpy/core/_asarray.py:83: VisibleDeprecationWarn
ing:

Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-t
uples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do th
is, you must specify 'dtype=object' when creating the ndarray
```



In [34]:
```
fig, axises = plt.subplots(2, 1)
```

```
ax = (data_n
    .query('platform == "PS4"')
    .plot(kind='scatter', x='critic_score', y='total_sales', figsize=(15, 7), grid=True
)
plt.xlabel('Рейтинг критика')
plt.ylabel('Суммарные продажи (млн копий)')

ax = (data_n
    .query('platform == "PS4"')
    .plot(kind='scatter', x='user_score', y='total_sales', figsize=(15, 7), grid=True,
)
plt.xlabel('Рейтинг пользователя')
plt.ylabel('Суммарные продажи (млн копий)')
plt.show()
```



In [35]:
```
corr_critic = data_n.query('platform == "PS4"').critic_score.corr(data_n.total_sales)
corr_user = data_n.query('platform == "PS4"').user_score.corr(data_n.total_sales)
print('Платформа PS4')
print('Корреляция м-у рейтингом критика и суммарными продажами составляет\t{: .0%}'.for
print('Корреляция м-у рейтингом пользователя и суммарными продажами составляет {: .0%}'
```

```
Платформа PS4
Корреляция м-у рейтингом критика и суммарными продажами составляет       22%
Корреляция м-у рейтингом пользователя и суммарными продажами составляет  10%
```

In [36]:
```
for platf in data_nn['platform']:
    data_platf = data_n.query('platform == @platf')
    data_nn.loc[data_nn['platform'] == platf, 'cor_critic'] = round((data_platf
        .critic_score.corr(data_platf.total_sales)) * 100, 3)
    data_nn.loc[data_nn['platform'] == platf, 'cor_user'] = round((data_platf
        .user_score.corr(data_platf.total_sales)) * 100, 3)
data_nn.loc[:, 'is_critic'] = data_nn.cor_critic > data_nn.cor_user
display(data_nn[['platform', 'sales', 'cor_critic', 'cor_user']])
print('В {: .0%} платформ корреляция м-у рейтингом критика и продажами больше,\n'
      'чем корреляция с рейтингом пользователя и продажами больше'\
      .format(data_nn.is_critic.sum() / data_nn.shape[0]))
```

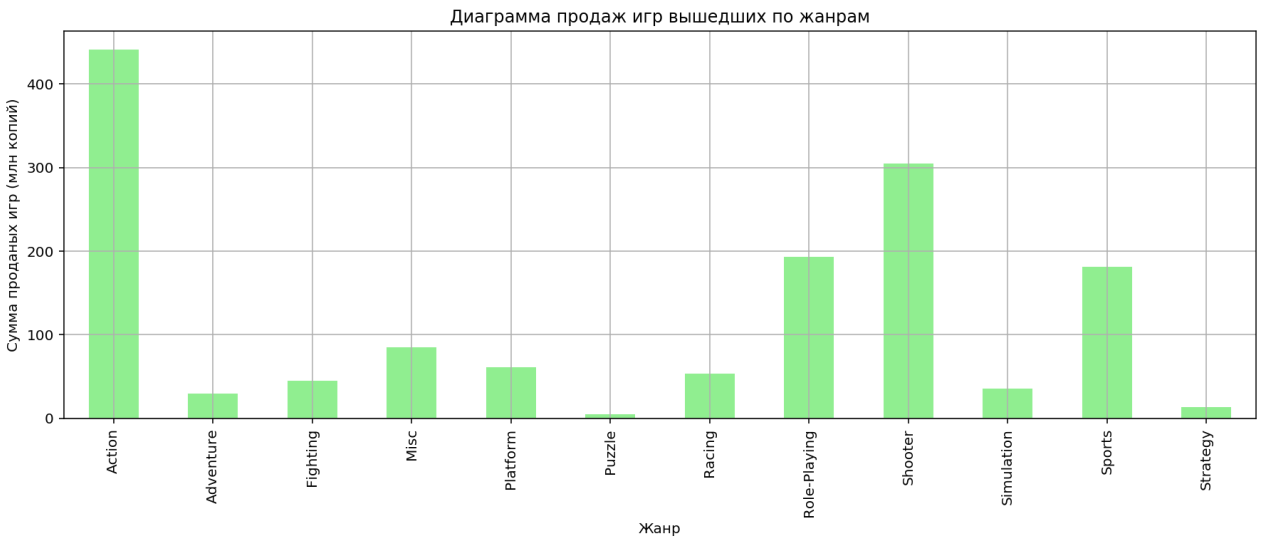
	platform	sales	cor_critic	cor_user

	platform	sales	cor_critic	cor_user
0	PS4	0.422	21.708	10.151
1	3DS	0.329	12.810	12.642
2	WiiU	0.329	25.108	26.676
3	XOne	0.301	19.750	5.998
4	Wii	0.180	40.385	58.230
5	X360	0.117	22.939	8.500
6	PC	0.097	21.075	4.523
7	PS3	0.095	28.372	21.004
8	PSV	0.050	40.218	38.617

```
В  78% платформ корреляция м-у рейтингом критика и продажами больше,
чем корреляция с рейтингом пользователя и продажами больше
```

Анализ по жанрам

In [37]:
```
ax = data_n.groupby('genre')['total_sales'].sum().plot(kind='bar', figsize=(15, 5), gri
plt.title('Диаграмма продаж игр вышедших по жанрам')
plt.xlabel('Жанр')
plt.ylabel('Сумма проданых игр (млн копий)')
plt.show()
```



Имеют место жанры с максимальными (`Action` , `Shooter`) и минимальными (`Puzzle` , `Strategy`) продажами

4. Составление портрета пользователя каждого региона

Топ-5 платформ

In [38]:
```
data_top_pl = data_n.groupby('platform')[['na_sales', 'eu_sales', 'jp_sales']].median()
for col in ['na_sales', 'eu_sales', 'jp_sales']:
    display(data_top_pl[col].sort_values(ascending=False).head(5).reset_index())
```

	platform	na_sales

	platform	na_sales
0	X360	0.170
1	XOne	0.120
2	WiiU	0.110
3	PS4	0.060
4	PS3	0.050

	platform	eu_sales
0	X360	0.100
1	PS4	0.080
2	XOne	0.070
3	WiiU	0.070
4	PC	0.060

	platform	jp_sales
0	3DS	0.060
1	PSV	0.030
2	PSP	0.030
3	PS3	0.030
4	PS4	0.010

Могли убедиться в том, что пользователи из Северной Америки покупаеют игры чаще остальных рассметриваемых территорий

Топ-5 жанров

In [39]:
```
data_n.groupby('genre')['total_sales'].sum().sort_values(ascending=False).reset_index()
```

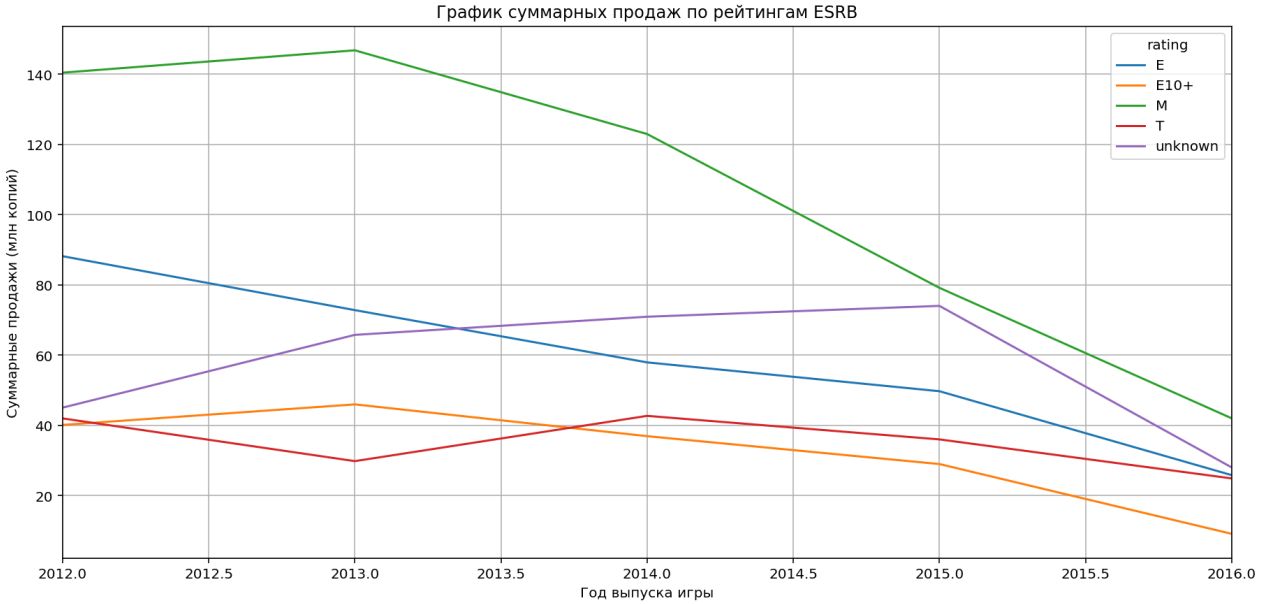
Out[39]:

	genre	total_sales
0	Action	441.120
1	Shooter	304.730
2	Role-Playing	192.800
3	Sports	181.070
4	Misc	85.040

Влияние рейтинга ESRB на продажи

In [40]:
```
(data_n
    .pivot_table(index='rating', columns='year_of_release', values='total_sales', aggfu
    .plot(figsize=(15, 7), grid=True)
)
```

```
plt.title('График суммарных продаж по рейтингам ESRB')
plt.xlabel('Год выпуска игры')
plt.ylabel('Суммарные продажи (млн копий)')
plt.show()
```



5. Проверка гипотез

Проверка гипотез будет сделана на основе Центральной Предельной Теоремы (ЦПТ), которая гласит, что

выборочные средние распределены нормально вокруг истинного среднего генеральной совокупности

Следовательно, выбрав уровень значимости в 5%, сможем понять, можем ли мы оставить (не отбрасывать)

нулевую гипотезу, либо же примем (не отвергнем) альтернативную гипотезу.

In [41]:
```
alpha = .05
```

Рейтинги различных платформ

Примем за нулевую гипотезу (H_0) следующее утверждение

Средние пользовательские рейтинги платформ Xbox One и PC одинаковые

Следовательно, аналогичная гипотеза (H_1) примет следующий вид

Средние пользовательские рейтинги платформ Xbox One и PC не одинаковы

In [42]:
```
hyps01 = (data
            .pivot_table(index='year_of_release', columns='platform', values='user_scor
            .dropna()
)

result = st.ttest_ind(hyps01['PC'], hyps01['XOne'])
if result.pvalue < alpha:
    print('Данные не позволяют иметь место нулевой гипотезе, принимаем альтернативную')
else:
    print('Не отклоняем нулевую гипотезу')
```

```
Не отклоняем нулевую гипотезу
```

Рейтинги различных жанров

Примем за нулевую гипотезу (H_0) следующее утверждение

`Средние пользовательские рейтинги жанров Action и Sports равны`

Следовательно, аналогичная гипотеза (H_1) примет следующий вид

`Средние пользовательские рейтинги жанров Action и Sports не равны`

In [43]:
```
hyps01 = (data
            .pivot_table(index='year_of_release', columns='genre', values='user_score')
            .dropna()
)

result = st.ttest_ind(hyps01['Action'], hyps01['Sports'])
if result.pvalue < alpha:
    print('Данные не позволяют иметь место нулевой гипотезе, принимаем альтернативную')
else:
    print('Не отклоняем нулевую гипотезу')
```

```
Не отклоняем нулевую гипотезу
```

6. Общий вывод

Вывод

Были проанализированные данные от 16 года для планирования рекламной компании "Стримчик" на 2017 год.
По условию не сказано, где находится магазин "Стримчик", но если делать предположение, что это сеть, на
несколько континентах, то можно утверждать, что платформы `XOne` , `PS4` , `PC` , `WiiU` , `X360` , `PS3` - те платформы, игры на которые следует закупить на 2017 год.
