Анализ продажи игр

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 (например, Xbox или PlayStation). Вам нужно выявить определяющие успешность игры закономерности. Это позволит сделать ставку на потенциально популярный продукт и спланировать рекламные кампании.

Перед вами данные до 2016 года. Представим, что сейчас декабрь 2016 г., и вы планируете кампанию на 2017-й. Нужно отработать принцип работы с данными. Неважно, прогнозируете ли вы продажи на 2017 год по данным 2016-го или же 2027-й — по данным 2026 года.

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, например, «Для взрослых», «Для детей младшего возраста» или «Для подростков».

Данные представлены в файле */datasets/games.csv*

Таблица `games` :

- *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.

Данные за 2016 год могут быть неполными.

Оглавление

1. Открытие и изучение данных

In [1]:
```
# For better figure's quality
%config InlineBackend.figure_format = 'retina'
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats as st
```

In [2]:
```
# workaround to use praktikum file system as well as local windows system\n",
```

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

In [3]:
```
games.head(10)
```

Out[3]:

	Name	Platform	Year_of_Release	Genre	NA_sales	EU_sales	JP_sales	Other_sales	Critic_Score	User_Score	Rating
0	Wii Sports	Wii	2006.0	Sports	41.36	28.96	3.77	8.45	76.0	8	E
1	Super Mario Bros.	NES	1985.0	Platform	29.08	3.58	6.81	0.77	NaN	NaN	NaN
2	Mario Kart Wii	Wii	2008.0	Racing	15.68	12.76	3.79	3.29	82.0	8.3	E
3	Wii Sports Resort	Wii	2009.0	Sports	15.61	10.93	3.28	2.95	80.0	8	E
4	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	11.27	8.89	10.22	1.00	NaN	NaN	NaN
5	Tetris	GB	1989.0	Puzzle	23.20	2.26	4.22	0.58	NaN	NaN	NaN
6	New Super Mario Bros.	DS	2006.0	Platform	11.28	9.14	6.50	2.88	89.0	8.5	E
7	Wii Play	Wii	2006.0	Misc	13.96	9.18	2.93	2.84	58.0	6.6	E
8	New Super Mario Bros. Wii	Wii	2009.0	Platform	14.44	6.94	4.70	2.24	87.0	8.4	E
9	Duck Hunt	NES	1984.0	Shooter	26.93	0.63	0.28	0.47	NaN	NaN	NaN

Данные прочитаны, разделитель указан верный.

Проверим общие показатели датафрейма.

In [4]:
```
games.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
```

Все названия колонок нужно привести к нижнему регистру. Пользовательский рейтинг хранится в формате строки, а не float, также год выпуска можно хранить не в числовом виде, а в виде строки или формата даты-времени. Есть пропущенные строки, посмотрим подробнее.

In [5]:
```
games.isna().sum()
```

Out[5]:
```
Name                  2
Platform              0
Year_of_Release     269
Genre                 2
NA_sales              0
EU_sales              0
JP_sales              0
Other_sales           0
Critic_Score       8578
User_Score         6701
Rating             6766
dtype: int64
```

Пропуски содержаться в датах релиза и рейтингах. Есть пропущенные значение в названии игр, посмотрим внимательные.

In [6]:
```
games[games['Name'].isna()]
```

Out[6]:

	Name	Platform	Year_of_Release	Genre	NA_sales	EU_sales	JP_sales	Other_sales	Critic_Score	User_Score	Rating

	Name	Platform	Year_of_Release	Genre	NA_sales	EU_sales	JP_sales	Other_sales	Critic_Score	User_Score	Rating
659	NaN	GEN	1993.0	NaN	1.78	0.53	0.00	0.08	NaN	NaN	NaN
14244	NaN	GEN	1993.0	NaN	0.00	0.00	0.03	0.00	NaN	NaN	NaN

В ходе предобработки данных предстоит решить, что делать с этими пропусками. Так как таких записей всего лишь 2, можно заменить здесь пропуски на название `unknown`.

Проверим, есть ли полные дубликаты в датафрейме.

In [7]:
```
games.duplicated().sum()
```

Out[7]:
```
0
```

Дубликатов нет.

Вывод раздела 1

Мы прочитали исходные данные, определили возможные проблемы с типами данных, пропусками в параметрах и названии колонок.

2. Предобработка данных

Приведем все названия колонок к нижнему регистру.

In [8]:
```
games.columns = [x.lower() for x in games.columns]
```

In [9]:
```
games.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
```

Как мы уже увидели в предыдущем разделе рейтинг среди игроков хранится в виде строки, посмотрим какие значения встречаются.

In [10]:
```
games['user_score'].value_counts()
```

Out[10]:
```
tbd    2424
7.8     324
8       290
8.2     282
8.3     254
       ...
1.3       2
9.6       2
0.7       2
0         1
9.7       1
Name: user_score, Length: 96, dtype: int64
```

In [11]:
```
games[games['user_score'] == 'tbd'].head(5)
```

Out[11]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating
119	Zumba Fitness	Wii	2010.0	Sports	3.45	2.59	0.0	0.66	NaN	tbd	E
301	Namco Museum: 50th Anniversary	PS2	2005.0	Misc	2.08	1.35	0.0	0.54	61.0	tbd	E10+
520	Zumba Fitness 2	Wii	2011.0	Sports	1.51	1.03	0.0	0.27	NaN	tbd	T

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating
645	uDraw Studio	Wii	2010.0	Misc	1.65	0.57	0.0	0.20	71.0	tbd	E
657	Frogger's Adventures: Temple of the Frog	GBA	NaN	Adventure	2.15	0.18	0.0	0.07	73.0	tbd	E

Практически 20% от всех присутствующих значений рейтинга среди пользователей занимают значения *tbd*. Это значение показывает, что рейтинг еще не определен и ожидается для игр в том источнике, который использовался для получения данных для анализа. Такое значение встречается для, в том числе, старых игр (от 2005 года, например). Скорее всего, рейтинг среди пользователей в таких случаях не определен и не будет определен в дальнейшем, либо сбор данных от пользователей для этой игры прекращен. Такие значения можно заменить на **пропуски**, так как в действительно для анализа нужны именно данные в наличии, кроме того, в этом столбце нужны значения в численном формате для возможности анализа. Альтернативным подходом можно считать выделения таких значений в отдельный столбец.

In [12]:
```
# Updated
games['user_score'] = games['user_score'].replace('tbd', np.nan)
```

In [13]:
```
games.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       7590 non-null   object
 10  rating           9949 non-null   object
dtypes: float64(6), object(5)
memory usage: 1.4+ MB
```

In [14]:
```
games['user_score'].value_counts()
```

Out[14]:
```
7.8    324
8      290
8.2    282
8.3    254
8.5    253
      ...
1.3      2
9.6      2
0.7      2
0        1
9.7      1
Name: user_score, Length: 95, dtype: int64
```

Теперь оставшиеся значения можно преобразовать в числовой формат.

In [15]:
```
games['user_score'] = games['user_score'].astype(float)
```

In [16]:
```
games.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       7590 non-null   float64
 10  rating           9949 non-null   object
```

```
dtypes: float64(7), object(4)
memory usage: 1.4+ MB
```

In [17]:
```
games.sample(5)
```

Out[17]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating
5956	Kurohyou: Ryu ga Gotoku Shinshou	PSP	2010.0	Adventure	0.00	0.00	0.29	0.00	NaN	NaN	NaN
14142	FabStyle	3DS	2011.0	Strategy	0.00	0.00	0.03	0.00	NaN	NaN	NaN
15986	Hiiro no Kakera Portable	PSP	2008.0	Adventure	0.00	0.00	0.02	0.00	NaN	NaN	NaN
14185	Need for Speed (2015)	PC	2016.0	Racing	0.00	0.03	0.00	0.00	NaN	NaN	NaN
6731	SBK Superbike World Championship	PS3	2008.0	Racing	0.12	0.11	0.00	0.02	59.0	6.1	E10+

Разбор пропусков.

In [18]:
```
games.isna().sum()
```

Out[18]:
```
name                 2
platform             0
year_of_release    269
genre                2
na_sales             0
eu_sales             0
jp_sales             0
other_sales          0
critic_score      8578
user_score        9125
rating            6766
dtype: int64
```

В изначальных данных много пропущенных значений в различных рейтингах, которые, скорее всего, вызваны тем, что открытые источники, которые использовались для получения данных для анализа, не имеют информации о части игр, либо они выходили на рынках, данных по которым нет у этих источников.

Оставим пропуски в рейтингах `critic_score`, `user_score` и `rating` как есть, так как их нельзя заменить каким-либо подходящим значением без искажения данных для анализа. Также этих строчек достаточно много и в других колонках есть полезная информация, а значит удалять их из рассмотрения нельзя.

Вызывают интерес 2 записи с пропусками значений в названии игры и жанре, посмотрим на них.

In [19]:
```
games[games['name'].isna()]
```

Out[19]:

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

Так как таких записей всего 2 от всей выборки и дата выпуска игры очень давняя, **эти записи можно просто удалить**. Это не исказит анализ.

In [20]:
```
games.dropna(subset=['name'], inplace=True)
```

In [21]:
```
games.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16713 entries, 0 to 16714
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   name             16713 non-null  object 
 1   platform         16713 non-null  object 
 2   year_of_release  16444 non-null  float64
 3   genre            16713 non-null  object 
 4   na_sales         16713 non-null  float64
 5   eu_sales         16713 non-null  float64
 6   jp_sales         16713 non-null  float64
 7   other_sales      16713 non-null  float64
```

```
 8   critic_score     8137 non-null   float64
 9   user_score       7590 non-null   float64
 10  rating           9949 non-null   object
dtypes: float64(7), object(4)
memory usage: 1.5+ MB
```

Остались пропуски в столбце с датой выхода игры.

In [22]:
```
games[games['year_of_release'].isna()]
```

Out[22]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating
183	Madden NFL 2004	PS2	NaN	Sports	4.26	0.26	0.01	0.71	94.0	8.5	E
377	FIFA Soccer 2004	PS2	NaN	Sports	0.59	2.36	0.04	0.51	84.0	6.4	E
456	LEGO Batman: The Videogame	Wii	NaN	Action	1.80	0.97	0.00	0.29	74.0	7.9	E10+
475	wwe Smackdown vs. Raw 2006	PS2	NaN	Fighting	1.57	1.02	0.00	0.41	NaN	NaN	NaN
609	Space Invaders	2600	NaN	Shooter	2.36	0.14	0.00	0.03	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...
16373	PDC World Championship Darts 2008	PSP	NaN	Sports	0.01	0.00	0.00	0.00	43.0	NaN	E10+
16405	Freaky Flyers	GC	NaN	Racing	0.01	0.00	0.00	0.00	69.0	6.5	T
16448	Inversion	PC	NaN	Shooter	0.01	0.00	0.00	0.00	59.0	6.7	M
16458	Hakuouki: Shinsengumi Kitan	PS3	NaN	Adventure	0.01	0.00	0.00	0.00	NaN	NaN	NaN
16522	Virtua Quest	GC	NaN	Role-Playing	0.01	0.00	0.00	0.00	55.0	5.5	T

269 rows × 11 columns

In [23]:
```
games[games['name'] == 'LEGO Batman: The Videogame']
```

Out[23]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating
397	LEGO Batman: The Videogame	X360	2008.0	Action	2.04	1.02	0.0	0.32	76.0	7.9	E10+
456	LEGO Batman: The Videogame	Wii	NaN	Action	1.80	0.97	0.0	0.29	74.0	7.9	E10+
460	LEGO Batman: The Videogame	DS	2008.0	Action	1.75	1.01	0.0	0.29	72.0	8.0	E10+
1519	LEGO Batman: The Videogame	PS3	2008.0	Action	0.72	0.39	0.0	0.19	75.0	7.7	E10+
1538	LEGO Batman: The Videogame	PSP	NaN	Action	0.57	0.44	0.0	0.27	73.0	7.4	E10+
1553	LEGO Batman: The Videogame	PS2	2008.0	Action	0.72	0.03	0.0	0.52	77.0	8.9	E10+
12465	LEGO Batman: The Videogame	PC	2008.0	Action	0.02	0.03	0.0	0.01	80.0	7.8	E10+

In [24]:
```
games[games['name'] == 'FIFA Soccer 2004']
```

Out[24]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating
377	FIFA Soccer 2004	PS2	NaN	Sports	0.59	2.36	0.04	0.51	84.0	6.4	E
2606	FIFA Soccer 2004	XB	2003.0	Sports	0.24	0.49	0.00	0.05	82.0	8.2	E
12029	FIFA Soccer 2004	GC	2003.0	Sports	0.05	0.01	0.00	0.00	83.0	6.2	E
13086	FIFA Soccer 2004	GBA	2003.0	Sports	0.04	0.01	0.00	0.00	82.0	7.9	E

Часто игры выходят сразу на нескольких платформах, поэтому разумно будет предположить, что пропущенные значения для конкретной платформы по дате выпуска можно заменить на дату выхода на другой платформе.

Создадим вспомогательный датафрейм с временами выхода игр.

In [25]:
```
games_dates = games.groupby('name')['year_of_release'].max()
```

А также вспомогательную функцию.

In [26]:
```
def fill_dates(row):
    if pd.isnull(row['year_of_release']):
        name = row['name']
        return games_dates[name]
    else:
        return row['year_of_release']
```

In [27]:
```
games['year_of_release'] = games.apply(fill_dates, axis=1)
```

In [28]:
```
games['year_of_release'].isna().sum()
```

Out[28]: 146

In [29]:
```
games[games['name'] == 'FIFA Soccer 2004']
```

Out[29]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating
377	FIFA Soccer 2004	PS2	2003.0	Sports	0.59	2.36	0.04	0.51	84.0	6.4	E
2606	FIFA Soccer 2004	XB	2003.0	Sports	0.24	0.49	0.00	0.05	82.0	8.2	E
12029	FIFA Soccer 2004	GC	2003.0	Sports	0.05	0.01	0.00	0.00	83.0	6.2	E
13086	FIFA Soccer 2004	GBA	2003.0	Sports	0.04	0.01	0.00	0.00	82.0	7.9	E

Количество пропусков уменьшилось, но остались те игры, для которых даты выхода нет совсем.

In [30]:
```
games[games['year_of_release'].isna()]
```

Out[30]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating
475	wwe Smackdown vs. Raw 2006	PS2	NaN	Fighting	1.57	1.02	0.00	0.41	NaN	NaN	NaN
627	Rock Band	X360	NaN	Misc	1.93	0.33	0.00	0.21	92.0	8.2	T
657	Frogger's Adventures: Temple of the Frog	GBA	NaN	Adventure	2.15	0.18	0.00	0.07	73.0	NaN	E
805	Rock Band	Wii	NaN	Misc	1.33	0.56	0.00	0.20	80.0	6.3	T
1142	Rock Band	PS3	NaN	Misc	0.99	0.41	0.00	0.22	92.0	8.4	T
...	...	...	...	...	...	...	...	...	...	...	...
16277	Homeworld Remastered Collection	PC	NaN	Strategy	0.00	0.01	0.00	0.00	86.0	8.2	E10+
16288	Shorts	DS	NaN	Platform	0.01	0.00	0.00	0.00	NaN	NaN	E10+
16348	Agarest Senki: Re-appearance	PS3	NaN	Role-Playing	0.00	0.00	0.01	0.00	NaN	NaN	NaN
16458	Hakuouki: Shinsengumi Kitan	PS3	NaN	Adventure	0.01	0.00	0.00	0.00	NaN	NaN	NaN
16522	Virtua Quest	GC	NaN	Role-Playing	0.01	0.00	0.00	0.00	55.0	5.5	T

146 rows × 11 columns

In [31]:
```
games[games['name'] == 'Rock Band']
```

Out[31]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating
627	Rock Band	X360	NaN	Misc	1.93	0.33	0.0	0.21	92.0	8.2	T
805	Rock Band	Wii	NaN	Misc	1.33	0.56	0.0	0.20	80.0	6.3	T
1142	Rock Band	PS3	NaN	Misc	0.99	0.41	0.0	0.22	92.0	8.4	T
1840	Rock Band	PS2	NaN	Misc	0.71	0.06	0.0	0.35	82.0	6.8	T

Пропуски в этих случаях в этом стобце нельзя заменить на какое-то среднее или медианное значение без искажения анализа, также остальные колонки в данных заполнены данными, поэтому просто удалить их будет неправильно.

Поэтому оставим эти пропуски в `year_of_release` без изменений.

In [32]:
```
games.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16713 entries, 0 to 16714
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   name             16713 non-null  object 
 1   platform         16713 non-null  object 
 2   year_of_release  16567 non-null  float64
 3   genre            16713 non-null  object 
 4   na_sales         16713 non-null  float64
 5   eu_sales         16713 non-null  float64
 6   jp_sales         16713 non-null  float64
 7   other_sales      16713 non-null  float64
 8   critic_score     8137 non-null   float64
 9   user_score       7590 non-null   float64
 10  rating           9949 non-null   object 
dtypes: float64(7), object(4)
memory usage: 1.5+ MB
```

Для дальнейшего анализа нам пригодятся знания о продажах по всему миру, поэтому добавим в датафрейм столбец с ними.

In [33]:
```
games['world_sales'] = games['na_sales'] + games['eu_sales'] + games['jp_sales'] + games['other_sales']
```

In [34]:
```
games.head(10)
```

Out[34]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating	world_sales
0	Wii Sports	Wii	2006.0	Sports	41.36	28.96	3.77	8.45	76.0	8.0	E	82.54
1	Super Mario Bros.	NES	1985.0	Platform	29.08	3.58	6.81	0.77	NaN	NaN	NaN	40.24
2	Mario Kart Wii	Wii	2008.0	Racing	15.68	12.76	3.79	3.29	82.0	8.3	E	35.52
3	Wii Sports Resort	Wii	2009.0	Sports	15.61	10.93	3.28	2.95	80.0	8.0	E	32.77
4	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	11.27	8.89	10.22	1.00	NaN	NaN	NaN	31.38
5	Tetris	GB	1989.0	Puzzle	23.20	2.26	4.22	0.58	NaN	NaN	NaN	30.26
6	New Super Mario Bros.	DS	2006.0	Platform	11.28	9.14	6.50	2.88	89.0	8.5	E	29.80
7	Wii Play	Wii	2006.0	Misc	13.96	9.18	2.93	2.84	58.0	6.6	E	28.91
8	New Super Mario Bros. Wii	Wii	2009.0	Platform	14.44	6.94	4.70	2.24	87.0	8.4	E	28.32
9	Duck Hunt	NES	1984.0	Shooter	26.93	0.63	0.28	0.47	NaN	NaN	NaN	28.31

Вывод раздела 2

В этом разделе мы:

- привели названия столбцов нижнему регистру

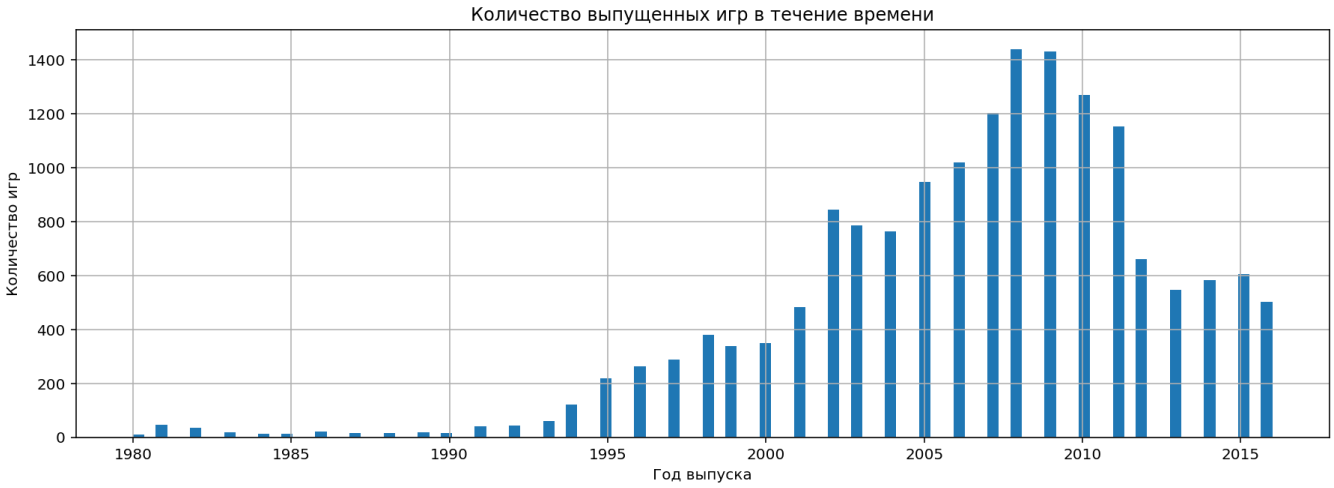
- преобразовали рейтинг пользователь в числовой формат
- обработали пропуски в данных, удалив строки без названия игр, частично заменив значения в пропусках года выпуска, а также оставив пропуски в рейтингах для сохранения полезных данных
- добавили столбец с суммарными продажами по всему миру.

3. Анализ данных

Количество выпущенных игр

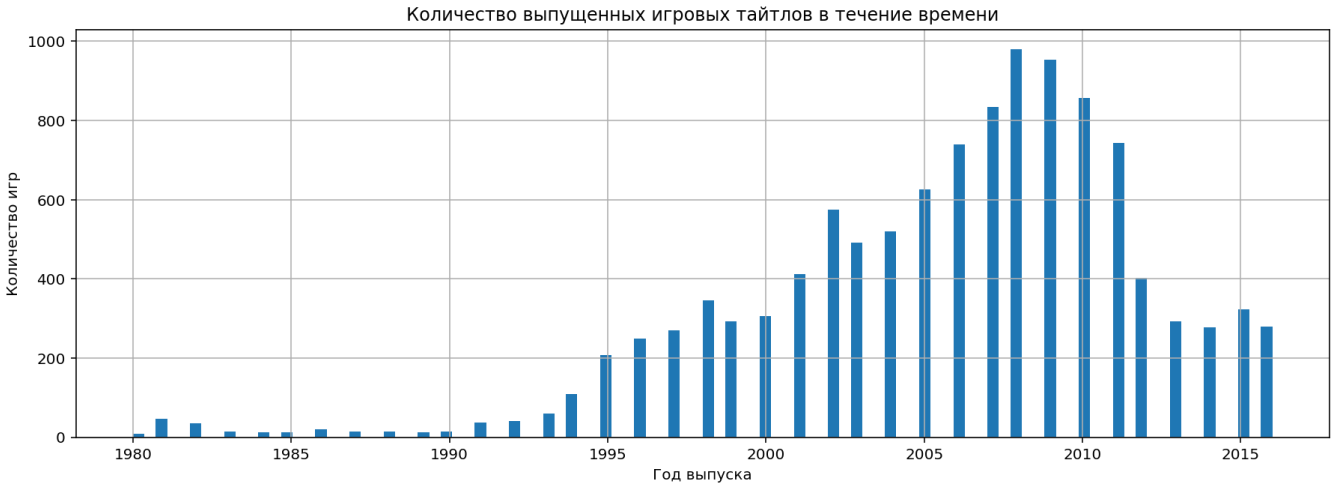
Посмотрим, сколько игр выпускалось в разные годы.

In [35]:
```
plt.figure(figsize=(15,5));
games['year_of_release'].hist(bins=100);
plt.title('Количество выпущенных игр в течение времени');
plt.xlabel('Год выпуска');
plt.ylabel('Количество игр');
```



Часто игры выпускаются сразу для нескольких платформ и в один год. Если смотреть только по названиям игр, то картина будет немного иная.

In [36]:
```
plt.figure(figsize=(15,5));
games.groupby('name')['year_of_release'].min().hist(bins=100);
plt.title('Количество выпущенных игровых тайтлов в течение времени');
plt.xlabel('Год выпуска');
plt.ylabel('Количество игр');
```



Характер распределения не меняется, только сократилось общее число игр.

Проверим распределение выпуска игр по годам.

In [37]:
```
games.groupby('year_of_release')['name'].count().describe()
```

Out[37]:
```
count      37.000000
mean      447.756757
std       455.963900
```

```
min         9.000000
25%        36.000000
50%       339.000000
75%       765.000000
max      1440.000000
Name: name, dtype: float64
```

Как видно, для дальнейшего анализа не нужны данные по очень старым играм до 1991 года, так как игр для анализа слишком мало. Отфильтруем датафрейм.

In [38]:
```
games_filtered_year = games.query('year_of_release > 1990')
```

In [39]:
```
games_filtered_year.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16346 entries, 0 to 16714
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   name             16346 non-null  object 
 1   platform         16346 non-null  object 
 2   year_of_release  16346 non-null  float64
 3   genre            16346 non-null  object 
 4   na_sales         16346 non-null  float64
 5   eu_sales         16346 non-null  float64
 6   jp_sales         16346 non-null  float64
 7   other_sales      16346 non-null  float64
 8   critic_score     8073 non-null   float64
 9   user_score       7538 non-null   float64
 10  rating           9866 non-null   object 
 11  world_sales      16346 non-null  float64
dtypes: float64(8), object(4)
memory usage: 1.6+ MB
```

In [40]:
```
games_filtered_year.head()
```

Out[40]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating	world_sales
0	Wii Sports	Wii	2006.0	Sports	41.36	28.96	3.77	8.45	76.0	8.0	E	82.54
2	Mario Kart Wii	Wii	2008.0	Racing	15.68	12.76	3.79	3.29	82.0	8.3	E	35.52
3	Wii Sports Resort	Wii	2009.0	Sports	15.61	10.93	3.28	2.95	80.0	8.0	E	32.77
4	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	11.27	8.89	10.22	1.00	NaN	NaN	NaN	31.38
6	New Super Mario Bros.	DS	2006.0	Platform	11.28	9.14	6.50	2.88	89.0	8.5	E	29.80

Данных в отфильтрованном по дате выхода датафрейме все еще достаточноо, только порядка 400 игр (около 2%) были удалены.

In [41]:
```
games = games_filtered_year
```

In [42]:
```
games.groupby('year_of_release')['name'].count().describe()
```

Out[42]:
```
count      26.000000
mean      628.692308
std       429.895408
min        41.000000
25%       301.500000
50%       564.500000
75%       921.500000
max      1440.000000
Name: name, dtype: float64
```

Продажи по платформам

Посмотрим, какие платформы самые популярные по общей выручке.

In [43]:
```
plt.figure(figsize=(15,5));
games.groupby('platform')['world_sales'].sum().sort_values(ascending=False).plot(kind='bar');
plt.title('Общие продажи по всему миру за все время на разных платформах');
plt.xlabel('Платформа');
```

```
plt.ylabel('Млн. проданных копий игр');
plt.grid(True, which='both');
```

Сохраним 10 крупнейших по продажам платформ в списке.

In [44]:
```
biggest_platforms_by_sales = games.groupby('platform')['world_sales'].sum().nlargest(10).index.to_list()
```

In [45]:
```
biggest_platforms_by_sales
```

Out[45]:
```
['PS2', 'X360', 'PS3', 'Wii', 'DS', 'PS', 'PS4', 'GBA', 'PSP', 'PC']
```

Выделим выборку только по крупнейшим платформам.

In [46]:
```
biggest_platforms_games = games.query('platform.isin(@biggest_platforms_by_sales)')
```

Для этих крупнейших платформ построим распределение по годам выпуска игр.

In [47]:
```
biggest_platforms_games.pivot_table(
    index='year_of_release',
    columns='platform',
    values='world_sales',
    aggfunc='sum').plot(style='o-', figsize=(15,5));
plt.title('Общие продажи по всему миру в разные года на разных платформах');
plt.xlabel('Год выхода');
plt.ylabel('Млн. проданных копий игр');
plt.legend(title='Платформа');
plt.grid(True, which='both');
```

Как видно на графике, платформы появляются (есть продажи игр для них), достигают пика и их популярность затем спадает с выходом новых. Особая ситуация с платформой **PC** так как она существует давно и продажи игр не прекращаются, но уступают другим платформам.

Если посмотреть на платформы Sony (PS, PS2, PS3, PS4), то можно отметить регулярные появления новой версии платформы, что **можно назвать "поколениями" платформ. Промежуток между ними примерно 5-7 лет**. Выход последней

платформы задержался на пару лет. Как видно, примерно с выходом PS3 продажи появились и у X360 от Microsoft. Как видно, **период жизни основных платформ - порядка 10 лет**.

Так как по заданию в проекте нам нужны данные для прогнозирования ситуации на рынке после 2016 года, **можно сделать выборку данных для платформ за 10 лет до этого - с 2006 года**, что покажет появление, развитие и закат платформ как минимум в одно "поколение".

In [48]:
```
games_matters = games.query('year_of_release >= 2006')
```

In [49]:
```
len(games_matters)
```

Out[49]: 10416

In [50]:
```
len(games)
```

Out[50]: 16346

In [51]:
```
len(games_matters) / len(games)
```

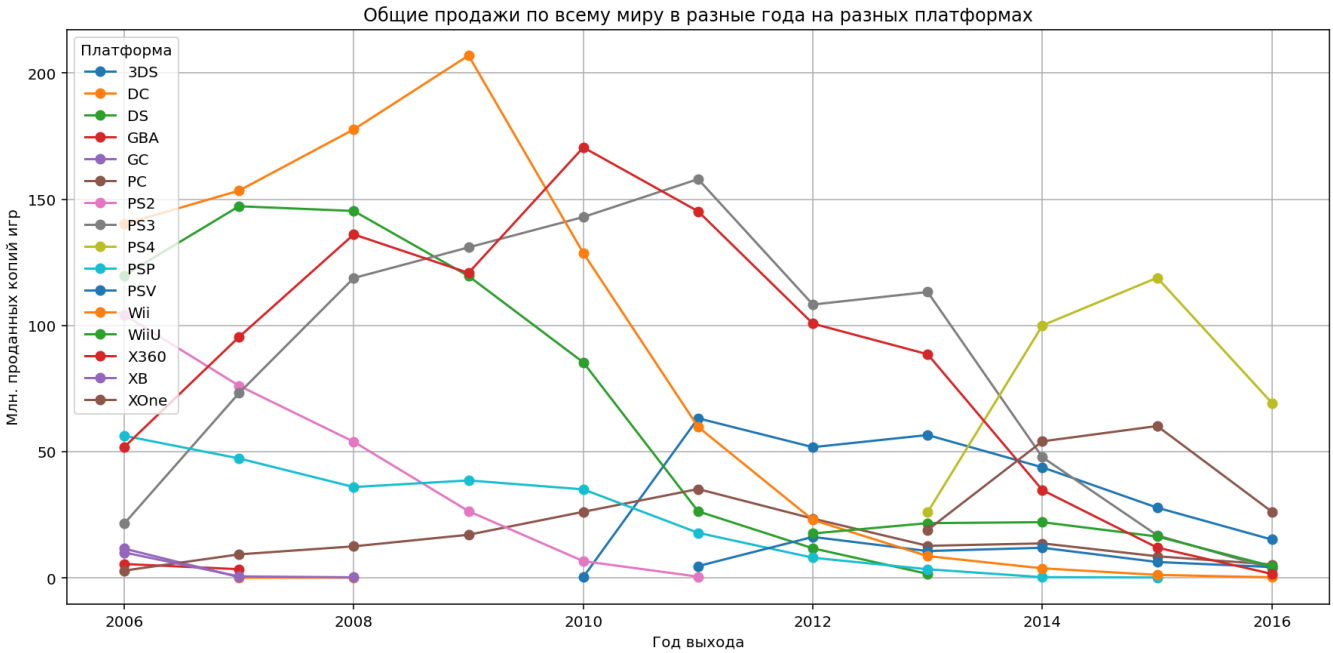
Out[51]: 0.6372201150128471

In [52]:
```
games = games_matters
```

Итого в выборке осталось 63% от изначальной (10 тыс. игр), что должно быть достаточно для анализа.

Посмотрим по уточненной выборке какие платформы лидируют по продажам, растут или падают.

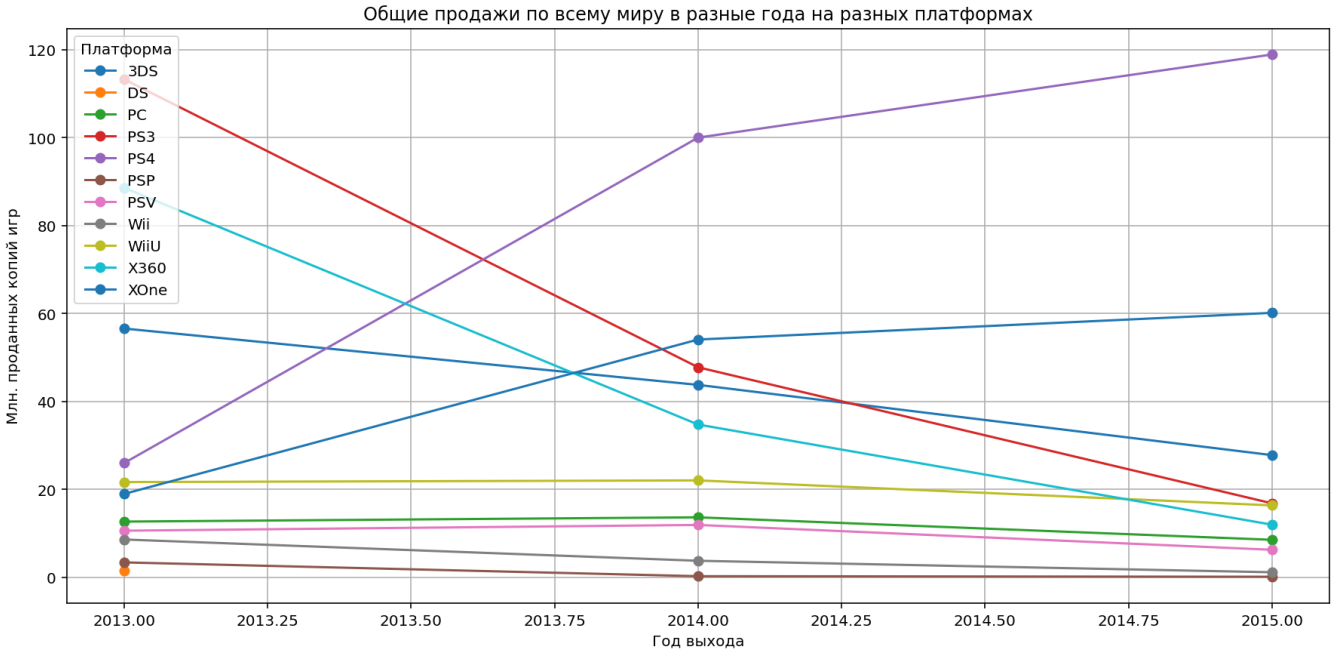
In [53]:
```
games.pivot_table(
    index='year_of_release',
    columns='platform',
    values='world_sales',
    aggfunc='sum').plot(style='o-', figsize=(15,7));
plt.title('Общие продажи по всему миру в разные года на разных платформах');
plt.xlabel('Год выхода');
plt.ylabel('Млн. проданных копий игр');
plt.legend(loc='upper left', title='Платформа');
plt.grid(True, which='both');
```



Если учесть, что по условиям данные за 2016 год могут быть неполными, посмотрим подробнее на годы с 2013 по 2015 включительно.

In [54]:
```
games.query('2013 <= year_of_release <= 2015').pivot_table(
    index='year_of_release',
    columns='platform',
    values='world_sales',
    aggfunc='sum').plot(style='o-', figsize=(15,7));
plt.title('Общие продажи по всему миру в разные года на разных платформах');
plt.xlabel('Год выхода');
```

```
plt.ylabel('Млн. проданных копий игр');
plt.legend(loc='upper left', title='Платформа');
plt.grid(True);
```



Как видно по предыдущим годам:

- для платформ PS4 и XOne наблюдается рост продаж
- для платформ PSV, PS3 и X360 наблюдается падение продаж
- платформа PS4 является лидером продаж за 2015 год, в 2 раза меньшие успехи у XOne, остальные по продажам уступают еще в 2 раза.

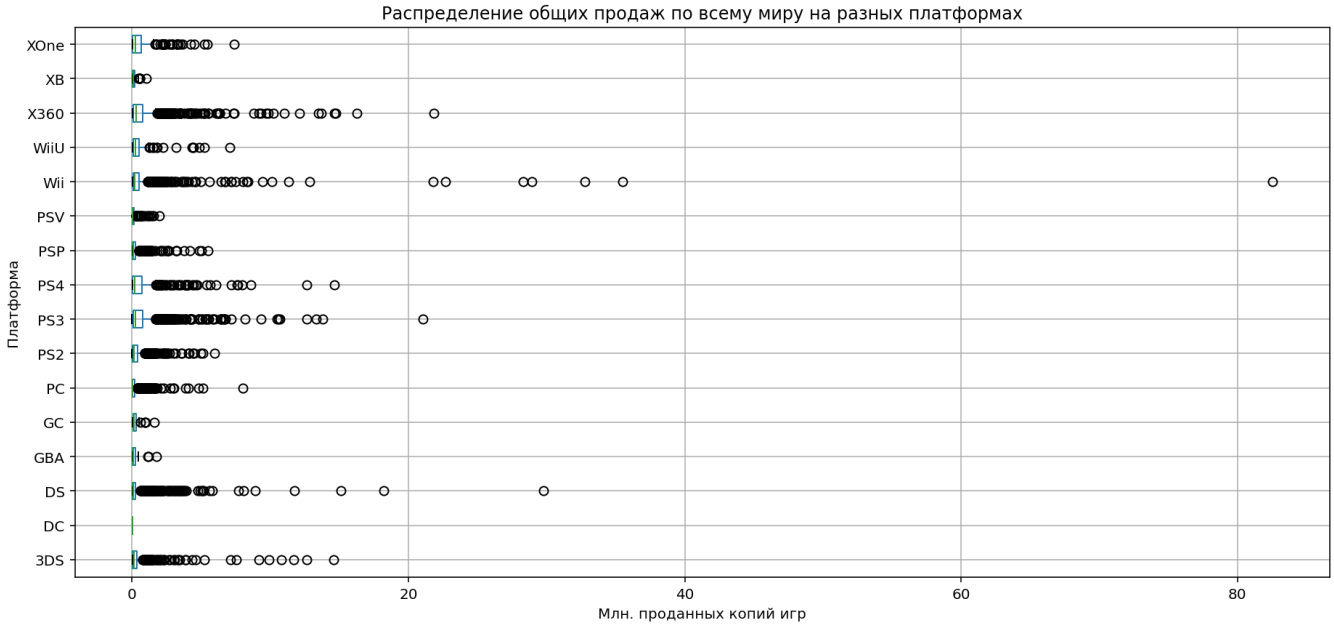
По видимому, происходит смена "поколений" платформ.

Таким образом, можно выделить платформы **PS4 и XOne** как потенциально прибыльные в ближайшие годы. Если предположить, что платформы достигнут своего пика продаж примерно через то же время, что и прошлое поколение, то эти плафтформы достигнут пика в 2017-2018 годах, что как раз необходимо заказчикам исследования.

Изучение распределений продаж

Посмотрим на **диаграмму размаха** по глобальным продажам игр в разбивке по платформам.

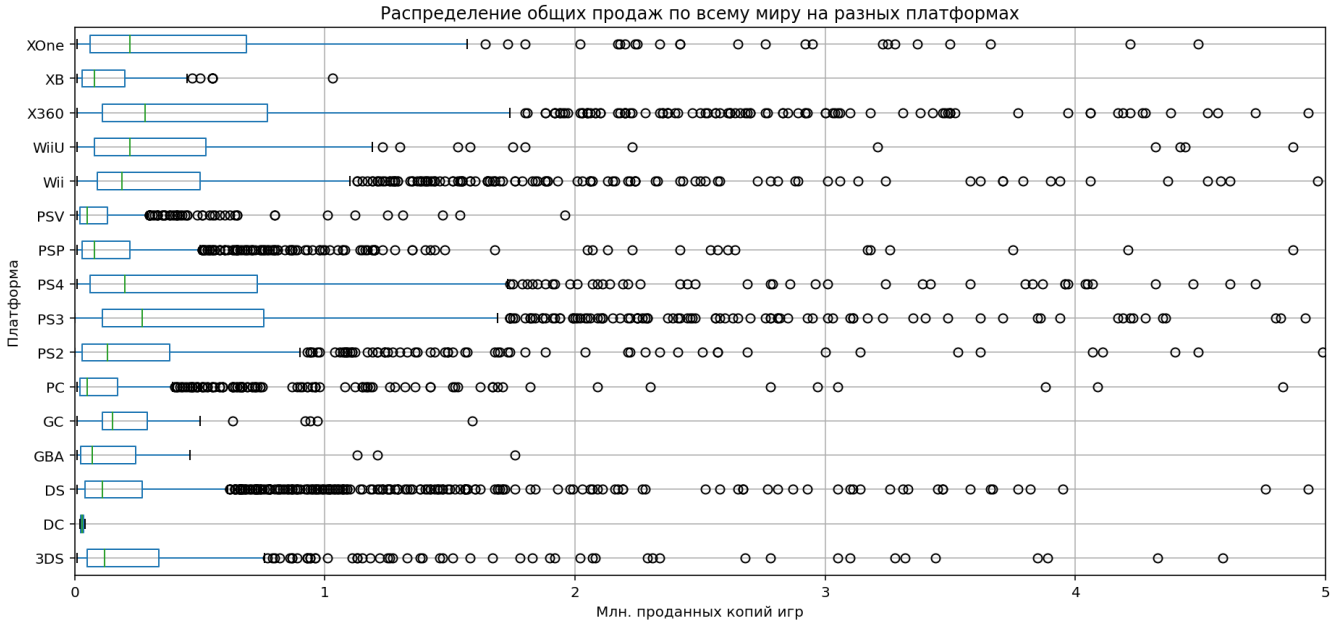
In [55]:
```
games[['platform', 'world_sales']].boxplot(by='platform', figsize=(15,7), vert=False);
plt.title('Распределение общих продаж по всему миру на разных платформах');
plt.suptitle('');
plt.ylabel('Платформа');
plt.xlabel('Млн. проданных копий игр');
plt.grid(True);
```



Как видно по графику, на нескольких платформах есть выдающиеся игры, которые продаются сильно лучше, чем в среднем на платформе. Среди таких платформ DS, PS3, PS4, Wii, X360.

Посмотрим подробнее на продажи до 5 млн. копий.

In [56]:
```
games[['platform', 'world_sales']].boxplot(by='platform', figsize=(15,7), vert=False);
plt.title('Распределение общих продаж по всему миру на разных платформах');
plt.suptitle('');
plt.ylabel('Платформа');
plt.xlabel('Млн. проданных копий игр');
plt.xlim(0, 5);
plt.grid(True);
```



На всех популярных платформах распределение не симметрично и смещено вправо (медиана (вертикальная зеленая линия) смещена в левую часть межквартильного интервала). Для наиболее популярных платформ наблюдаются длинные "усы" вплоть до **1,7 млн. проданных копий** (платформы XOne, X360, PS3, PS4). Это значит, что на этих платформах много "нормальных" игр по продажам, превышающим 1 млн. копий, чего нельзя сказать о других платформах. По всем платформам наблюдаются "выбросы" продаж игр, сильно превышающих характерное распределение, но их количество невелико.

По медианам продаж лидирует платформа X360, за ней PS3 и XOne.

Выделенные по графику **платформы XOne, X360, PS3 и PS4 можно назвать самыми популярными**.

Влияние отзывов на продажи

Посмотрим, как влияют на продажи внутри одной популярной платформы отзывы пользователей и критиков.

In [57]:
```
popular_platforms = ['XOne', 'X360', 'PS3', 'PS4']
```

Посмотрим на выборку по популярным платформам.

In [58]:
```
games.query('platform.isin(@popular_platforms)').groupby('platform')[['name', 'user_score', 'critic_score']].count()
```

Out[58]:

platform	name	user_score	critic_score
PS3	1319	869	813
PS4	392	257	252
X360	1229	938	887
XOne	247	182	169

Посмотрим отзывы и продажи для **PS3**, так как у нее достаточно много отзывов и игр в выборке. Также удалим из выборки выбросы по продажам свыше 1,7 млн. копий (по итогам уже проведенного анализа).

In [59]:
```
ps3_games = games.query('platform == "PS3" and world_sales <= 1.7')
```

In [60]:
```
ps3_games.head()
```

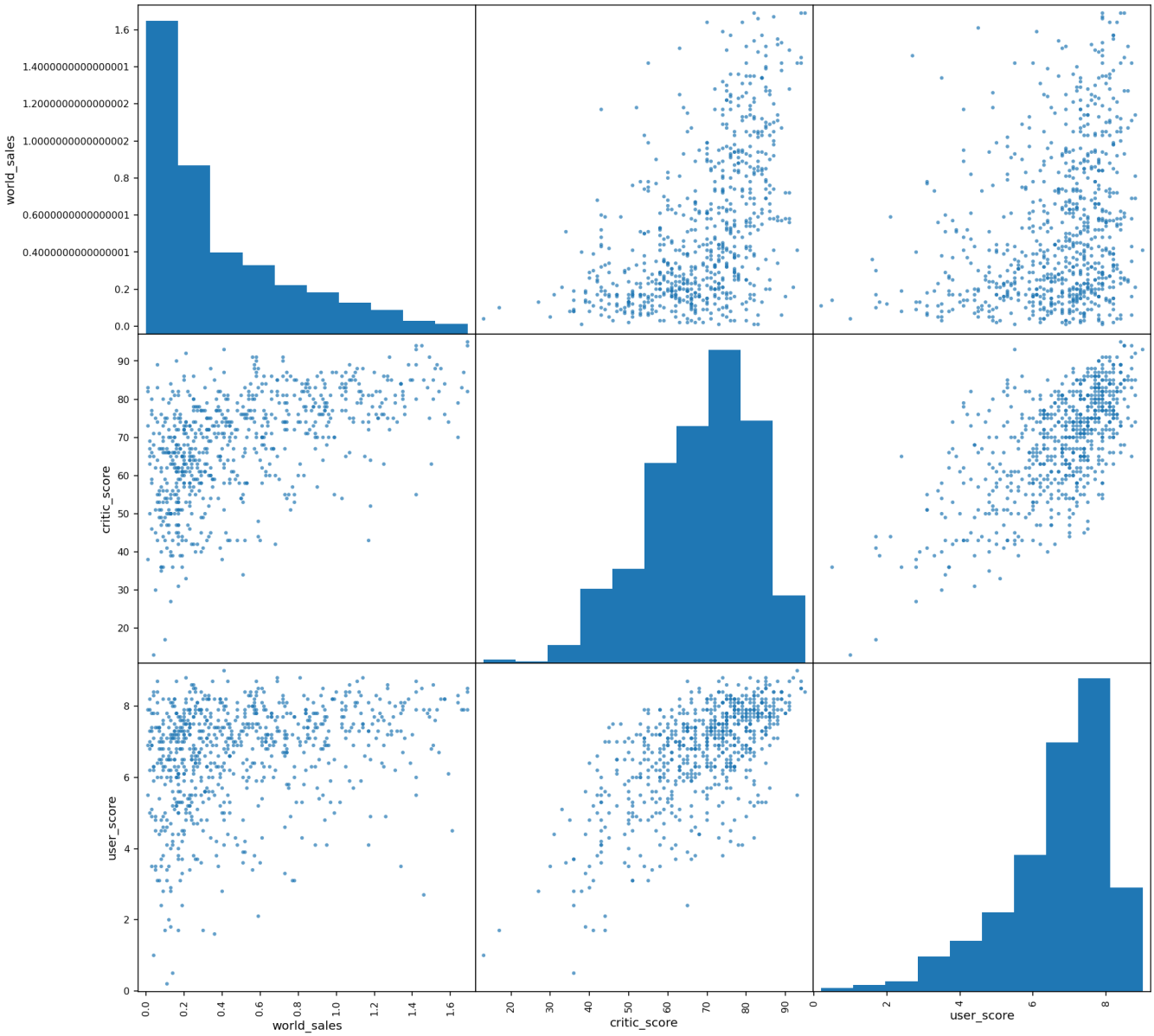
Out[60]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating	world_sales
1052	Portal 2	PS3	2011.0	Shooter	0.83	0.60	0.02	0.24	95.0	8.4	E10+	1.69
1060	BioShock Infinite	PS3	2013.0	Shooter	0.72	0.65	0.04	0.28	94.0	8.5	M	1.69
1062	Call of Duty: Black Ops 3	PS3	2015.0	Shooter	0.49	0.87	0.07	0.26	NaN	NaN	NaN	1.69
1075	Saints Row 2	PS3	2008.0	Action	0.88	0.54	0.02	0.25	82.0	7.9	M	1.69
1091	Dragon Age: Origins	PS3	2009.0	Role-Playing	0.96	0.42	0.08	0.21	87.0	7.9	M	1.67

Построим диаграммы рассеяния по интересующим нас параметрам.

In [61]:
```
pd.plotting.scatter_matrix(ps3_games[['world_sales', 'critic_score', 'user_score']], alpha=0.7, figsize=(15,15));
plt.suptitle('Диаграммы рассеяния мировых продаж, рейтингов критиков и рейтингов пользователей');
```

Диаграммы рассеяния мировых продаж, рейтингов критиков и рейтингов пользователей



Как видно по диаграммам рассеяния, продажи показывают некоторую зависимость от рейтинга критиков, но зависимость от рейтинга пользователей более хаотична. В то же время, рейтинг пользователей показывает линейную зависимость от рейтинга критиков.

Проверим корреляцию.

In [62]:
```
ps3_games['world_sales'].corr(ps3_games['critic_score'])
```

Out[62]: 0.5230682796180766

In [63]:
```
ps3_games['world_sales'].corr(ps3_games['user_score'])
```

Out[63]: 0.28342359186415794

In [64]:
```
ps3_games['critic_score'].corr(ps3_games['user_score'])
```

Out[64]: 0.6469528719032711

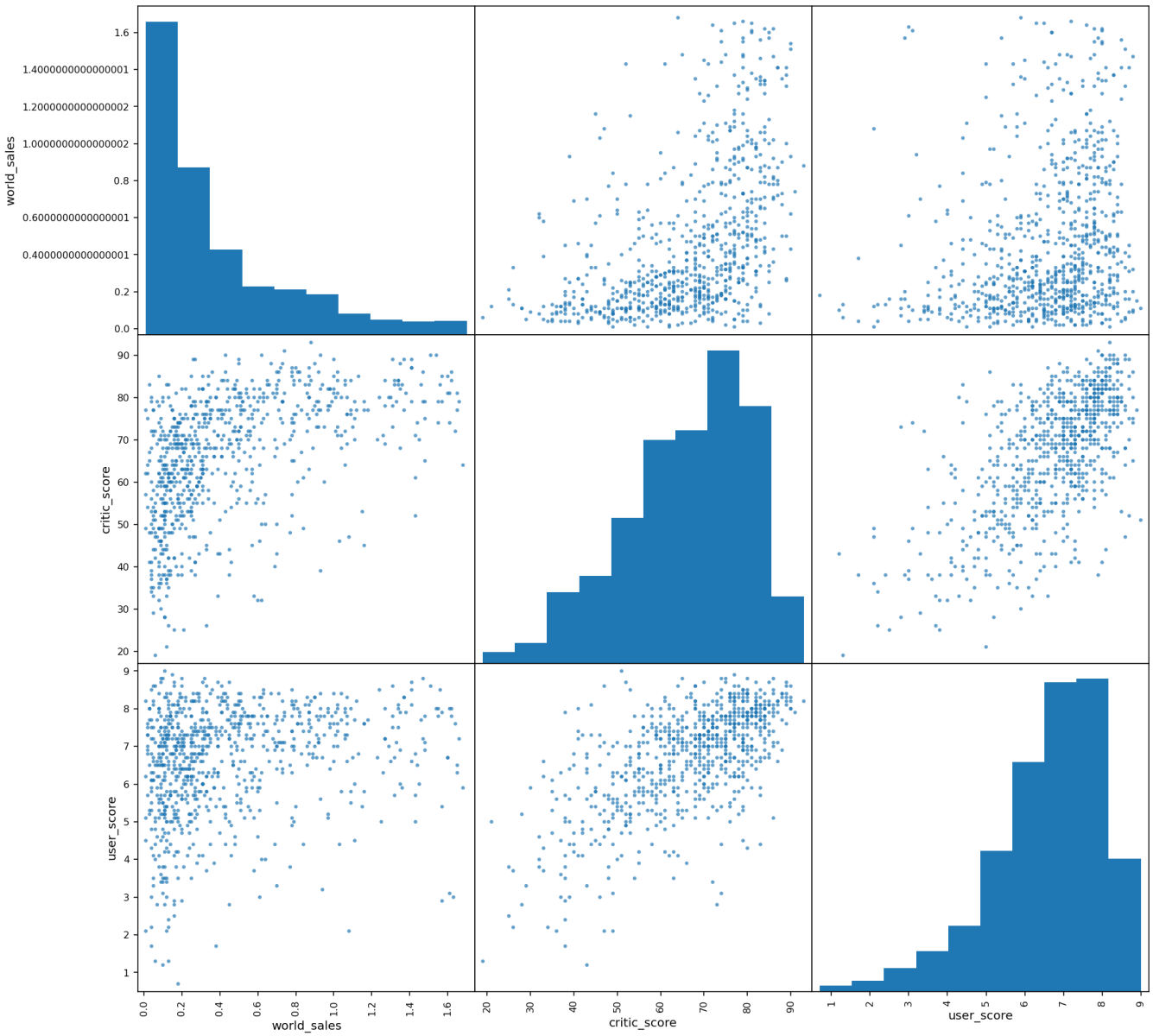
Итого можно заключить, что **мировые продажи в целом показывают среднюю корреляцию 0.52 от рейтинга критиков и слабую (0.28) от рейтинга пользователей**. Между собой **рейтинги пользователей и критиков связаны также со средним коэффициентом корреляции Пирсона 0.64**.

Проверим этот вывод на второй популярной платформе **X360**.

In [65]:
```
x360_games = games.query('platform == "X360" and world_sales <= 1.7')
```

In [66]:
```
pd.plotting.scatter_matrix(x360_games[['world_sales', 'critic_score', 'user_score']], alpha=0.7, figsize=(15,15));
plt.suptitle('Диаграммы рассеяния мировых продаж, рейтингов критиков и рейтингов пользователей');
```

Диаграммы рассеяния мировых продаж, рейтингов критиков и рейтингов пользователей



In [67]:
```
x360_games['world_sales'].corr(x360_games['critic_score'])
```

Out[67]:
```
0.48876969046857766
```

In [68]:
```
x360_games['world_sales'].corr(x360_games['user_score'])
```

Out[68]:
```
0.214662473169124
```

In [69]:
```
x360_games['critic_score'].corr(x360_games['user_score'])
```

Out[69]:
```
0.6342692337801112
```

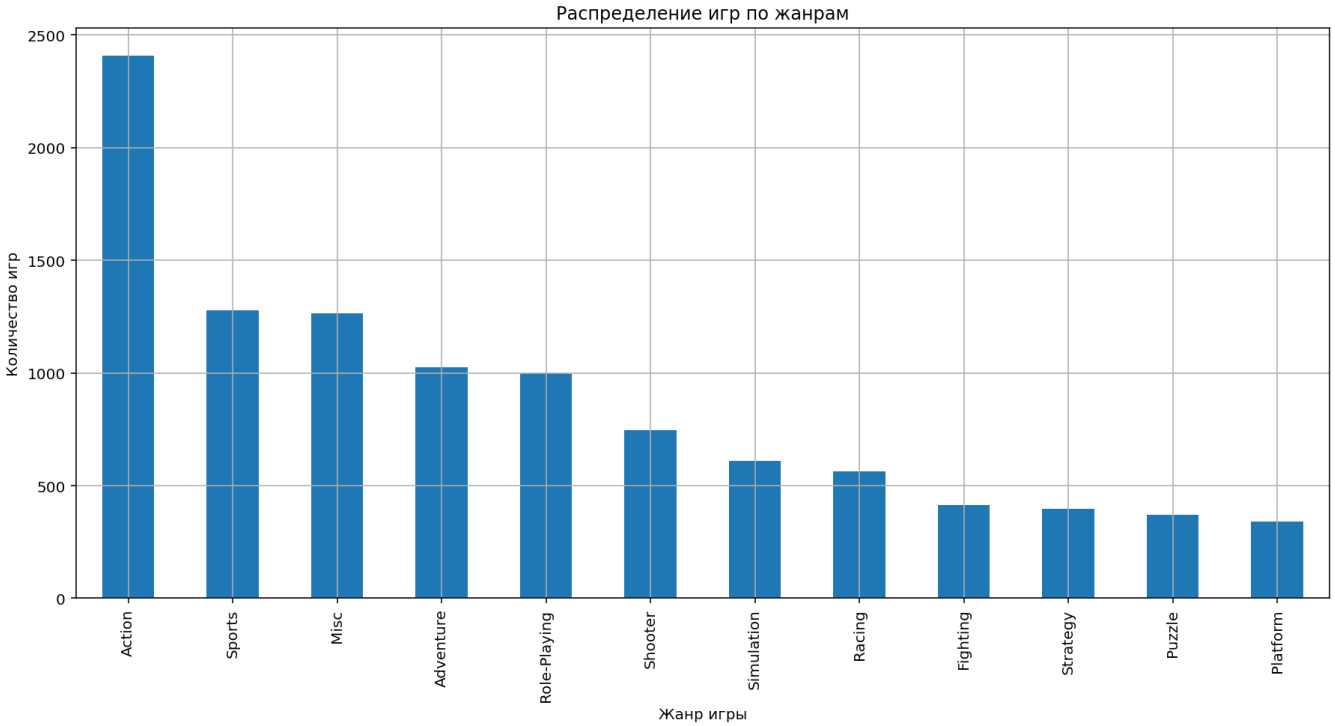
Как видно, эти же выводы верны для платформы X360 примерно с такими же показателя корреляции.

In [70]:
```
xone_games = games.query('platform == "XOne" and world_sales <= 1.7')
```

Зависимость продаж от жанра игры

Посмотрим на общее распределение игр по жанрам.

In [71]:
```
plt.figure(figsize=(15,7));
games.groupby('genre')['name'].count().sort_values(ascending=False).plot(kind='bar');
plt.grid(True);
plt.title('Распределение игр по жанрам');
plt.xlabel('Жанр игры');
plt.ylabel('Количество игр');
```



In [72]:
```
games.groupby('genre')['name'].count().sort_values(ascending=False)
```

Out[72]:
```
genre
Action          2409
Sports          1277
Misc            1265
Adventure       1024
Role-Playing     998
Shooter          747
Simulation       609
Racing           564
Fighting         415
Strategy         397
Puzzle           370
Platform         341
Name: name, dtype: int64
```

По распределению видно, что **наиболее частым жанром является Action**, которого в выборке представлено практически в 2 раза больше, чем другие жанры. Также большое количество игр в данных входят в категории Sports, Misc, Adventure, Role-Playing и Shooter.

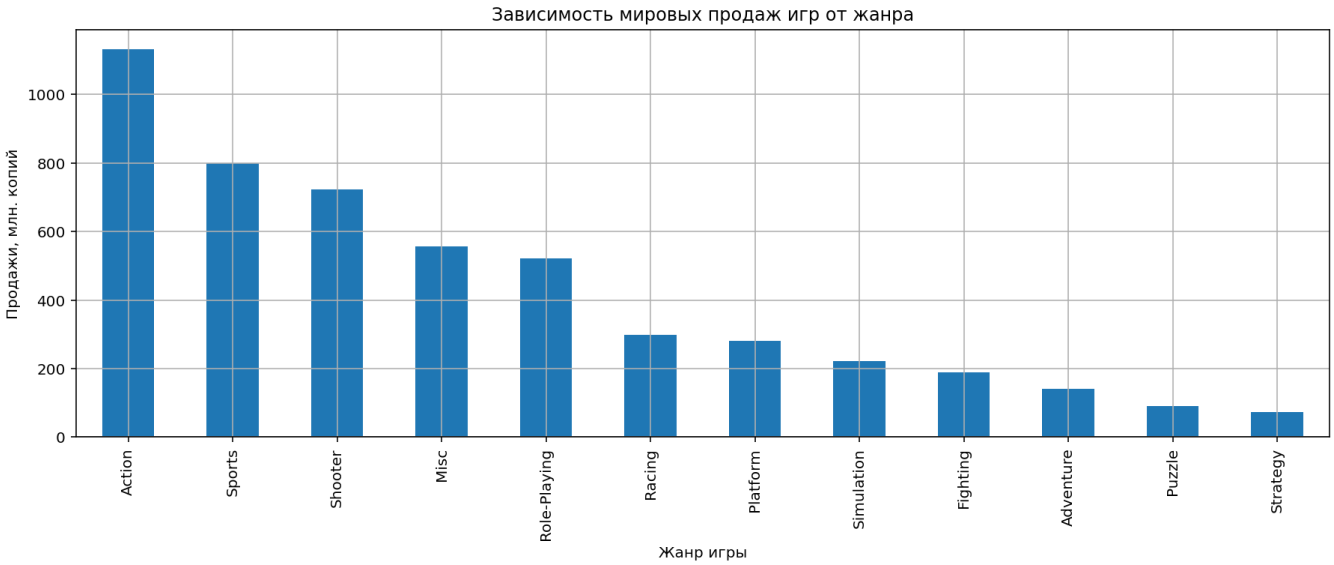
Посмотрим зависимость общих продаж от жанра.

In [73]:
```
games.groupby('genre')['world_sales'].sum().sort_values(ascending=False)
```

Out[73]:
```
genre
Action          1132.65
Sports           798.50
Shooter          722.36
Misc             557.78
Role-Playing     522.76
Racing           299.08
Platform         280.22
Simulation       221.15
Fighting         188.65
Adventure        141.37
Puzzle            90.57
```

```
Strategy          73.77
Name: world_sales, dtype: float64
```

In [74]:
```
plt.figure(figsize=(15,5));
games.groupby('genre')['world_sales'].sum().sort_values(ascending=False).plot(kind='bar', grid=True);
plt.title('Зависимость мировых продаж игр от жанра');
plt.xlabel('Жанр игры');
plt.ylabel('Продажи, млн. копий');
```



Как видно по графику, **наиболее прибыльным жанром является Action, за ним следуют Sports, Shooter, Misc, Role-Playing. Наименее прибыльным жанром являются стратегии.** Разница в продажах между Action и Strategy достигает десятков раз.

Вывод раздела 3

В этом разделе мы:

- изучили сколько игр выпускалось в разные годы и обнаружили, что количество данных в выборке старше 1990 года слишком мало, ими можно пренебречь без потери данных для анализа.
- изучили изменение продаж по платформам, **на характерном промежутке времени 10 лет возникает новое поколение платформ**, которое сменяет предыдущее.
- на основе полученных выводов данные в выборке **ограничили 2006 годом** за 10 лет до исследуемого.
- исследовали рост и падение разных платформ и выделили наиболее перспективные - **XOne и PS4**
- проанализировали глобальные продажи игр в разбивке по платформам и выделили основные PS3, PS4, X360, XOne, а также характерные разбросы продаж для них на уровне **1,7 млн. копий**.
- обнаружили среднюю положительную корреляцию между отзывами критиков и мировыми продажами игр для платформы PS3 и не обнаружили ее для рейтинга пользователей, полученные выводы подтвердились для платформы X360
- изучили зависимость продаж от жанра игры и выделили наиболее популярный (**Action**) и наименее (**Strategy**).

4. Изученение пользователей из разных регионов

Определим для пользователя каждого региона (NA, EU, JP) самые популярные платформы (ТОП-5).

In [75]:
```
games.groupby('platform')['na_sales'].sum().nlargest(5).reset_index()
```

Out[75]:

	platform	na_sales
0	X360	591.47
1	Wii	494.01
2	PS3	391.60
3	DS	325.06
4	PS2	115.75

In [76]:
```
games.groupby('platform')['eu_sales'].sum().nlargest(5).reset_index()
```

Out[76]:

	platform	eu_sales
0	PS3	328.83
1	X360	269.25
2	Wii	261.48
3	DS	143.30
4	PS4	141.09

In [77]:
```
games.groupby('platform')['jp_sales'].sum().nlargest(5).reset_index()
```

Out[77]:

	platform	jp_sales
0	DS	141.49
1	3DS	100.64
2	PS3	79.51
3	PSP	71.13
4	Wii	69.05

Тройка лидеров по продажам не отличается для NA и EU, но порядок их различен. **На рынке NA лидирует X360, на рынке EU - PS3**. Для **региона JP ситуация в корне иная и лидирует платформа DS, а X360 отсутствует в пятерке лидеров**. В то же время, на других рынках платформа DS находится на 4-ом месте по продажам.

Проверим распределение продаж по всему миру.

In [78]:
```
games.groupby('platform')['world_sales'].sum().nlargest(5).reset_index()
```

Out[78]:

	platform	world_sales
0	X360	957.39
1	PS3	935.00
2	Wii	903.31
3	DS	656.85
4	PS4	314.14

Платформа PS3 входит в тройку лидеров для всех регионов и является из-за этого одной из самых прибыльных. В то же время, **лидером по абсолютным продажам является X360 за счет рынка NA, несмотря на очень скромные продажи в регионе JP**.

Скорее всего, на долю продаж влияют как культурные различия (платформа из NA плохо продается в JP), так и качественная разница платформ по их мощности (в лидерах продаж мощные платформы X360 и PS3, а также относительно слабая Wii).

Посмотрим на самые популярные жарны и выделим топ-5 в каждом регионе.

In [79]:
```
games.groupby('genre')['na_sales'].sum().nlargest(5).reset_index()
```

Out[79]:

	genre	na_sales
0	Action	538.92
1	Sports	399.54
2	Shooter	375.65
3	Misc	286.20
4	Role-Playing	199.71

In [80]:
```
games.groupby('genre')['eu_sales'].sum().nlargest(5).reset_index()
```

Out[80]:

	genre	eu_sales
0	Action	356.35
1	Sports	252.52

	genre	eu_sales
2	Shooter	241.06
3	Misc	147.43
4	Racing	113.42

In [81]:
```
games.groupby('genre')['jp_sales'].sum().nlargest(5).reset_index()
```

Out[81]:

	genre	jp_sales
0	Role-Playing	170.58
1	Action	102.08
2	Misc	65.55
3	Sports	49.25
4	Platform	35.20

В регионах NA и EU тройка лидеров по жанрам одинаковая - лидирует с сильным преимуществом жанр Action, затем Sports и Shooter. Для рынка JP абсолютным лидером является жанр **Role-Playing**, не смотря на то, что он не присутствует в лидерах остальных рынков. На втором месте в JP идет также популярный жанр Action, затем Misc, а уже потом Sports.

Проверим зависимость продаж от рейтинга ESRB.

In [82]:
```
games.groupby('rating')['na_sales'].sum().nlargest(5).reset_index()
```

Out[82]:

	rating	na_sales
0	E	800.93
1	M	576.57
2	T	426.74
3	E10+	322.46
4	EC	1.32

In [83]:
```
games.groupby('rating')['eu_sales'].sum().nlargest(5).reset_index()
```

Out[83]:

	rating	eu_sales
0	E	462.11
1	M	401.36
2	T	236.87
3	E10+	178.24
4	RP	0.03

In [84]:
```
games.groupby('rating')['jp_sales'].sum().nlargest(5).reset_index()
```

Out[84]:

	rating	jp_sales
0	E	120.36
1	T	84.08
2	M	43.82
3	E10+	36.57
4	EC	0.00

На этих сводных данных можно сделать вывод, что для рынков EU и NA рейтинги одинаково влияют на продажи игр и первые 3 места занимают игры для возрастных групп E, M, T. В то же время для рынка JP наиболее популярным рейтингом являются игры с E, а далее T и M. Для рынка EU разница в продажах между рейтингами E и M практическ отсутствует, в то время как для NA игры с E опережают M в 1,5 раза. Таким образом, **для всех регионов наиболее популярными являются игры с рейтингом E**, а тройка лидеров по продажам не отличается.

Вывод раздела 4

В этом разделе мы определили для пользователей каждого региона (NA, EU, JP):

- самые популярные платформы (топ-5)
- самые популярные жанры (топ-5)
- зависимость продаж от рейтинга ESRB на разных рынках.

В результате можно составить 3 разных "портрета" пользователя для разных регионов с наибольшими продажами (наиболее популярны):

- для региона **NA** пользователи:
  - используют платформу **X360**
  - играют в жанр **Action**
  - с рейтингом ESRB E
- для региона **EU** пользователи:
  - используют платформу **PS3**
  - играют в жанр **Action**
  - с рейтингом ESRB E
- для региона **JP** пользователи:
  - используют платформу **DS**
  - играют в жанр **Role-Playing**
  - с рейтингом ESRB E

5. Проверка гипотез

Проверим следующие гипотезы:

- Средние пользовательские рейтинги платформ Xbox One и PC одинаковые;
- Средние пользовательские рейтинги жанров Action (англ. «действие», экшен-игры) и Sports (англ. «спортивные соревнования») разные.

Подготовим пользовательские рейтинги платформ Xbox One и PC.

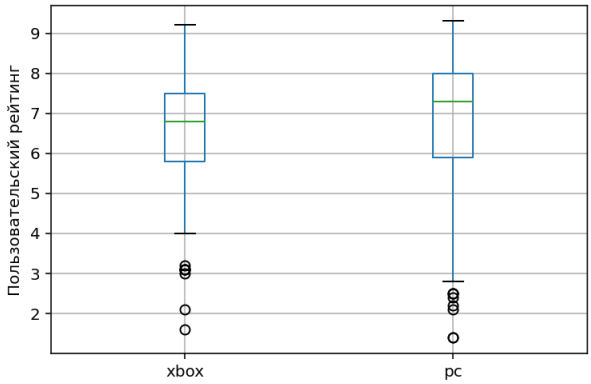
In [85]:
```
xbox_ratings = games.query('platform == "XOne"')['user_score'].reset_index(drop=True)
```

In [86]:
```
pc_ratings = games.query('platform == "PC"')['user_score'].reset_index(drop=True)
```

In [87]:
```
ratings = pd.DataFrame({'xbox': xbox_ratings, 'pc': pc_ratings})
```

In [88]:
```
ratings.boxplot();
plt.ylabel('Пользовательский рейтинг');
plt.suptitle('Распределение рейтинга пользователей для Xbox One и PC');
```

Распределение рейтинга пользователей для Xbox One и PC



In [89]:
```
ratings.describe()
```

Out[89]:

	xbox	pc
count	182.000000	626.000000
mean	6.521429	6.835942

	xbox	pc
std	1.380941	1.516586
min	1.600000	1.400000
25%	5.800000	5.900000
50%	6.800000	7.300000
75%	7.500000	8.000000
max	9.200000	9.300000

Как видно по распределениям оценок, для PC очень широк диапазон характерных значений пользовательских оценок от 3.0 до 9.3, тогда как для Xbox One характерные оценки начинаются с 4.0 и до 9.2. Медианные значения чуть выше для PC и составляет 7.3. Средние значения чуть ниже.

Избавимся от выбросов в оценках для подготовки к работе с гипотезами, чтобы результаты были надежнее.

In [90]:
```
xbox_ratings = xbox_ratings[xbox_ratings >= 4.0]
pc_ratings = pc_ratings[pc_ratings >= 3.0]
ratings = pd.DataFrame({'xbox': xbox_ratings, 'pc': pc_ratings})
```

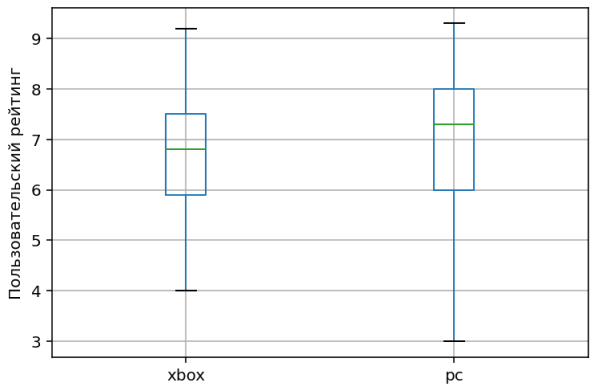
In [91]:
```
ratings.describe()
```

Out[91]:

	xbox	pc
count	174.000000	616.000000
mean	6.693103	6.910065
std	1.142990	1.410332
min	4.000000	3.000000
25%	5.900000	6.000000
50%	6.800000	7.300000
75%	7.500000	8.000000
max	9.200000	9.300000

In [92]:
```
ratings.boxplot();
plt.ylabel('Пользовательский рейтинг');
plt.suptitle('Распределение рейтинга пользователей для Xbox One и PC');
```

Распределение рейтинга пользователей для Xbox One и PC



Теперь данные подготвлены, можно использовать для работы с гипотезами.

Так как мы хотим изучить среднее двух выборок, для этого подойдет использование гипотезы о равенстве средних двух генеральных совокупностей.

Для первой гипотезы сформулируем нулевую гипотезу таким образом: средние пользовательские рейтинги платформ Xbox One и PC одинаковые (не отличаются). Альтернативной гипотезой будет ситуация, обратная этой.

Примем пороговое значение вероятности за 5%.

In [93]:
```
alpha = 0.05
sample1 = xbox_ratings.to_list()
```

```
sample2 = pc_ratings.to_list()
```

In [94]:
```
result = st.ttest_ind(sample1, sample2, equal_var=True)
result.pvalue
```

Out[94]:
```
0.06276780710517475
```

In [95]:
```
result.pvalue < alpha
```

Out[95]:
```
False
```

Нулевую гипотезу нельзя отвергнуть. Получается, что есть вероятность выше 5% совпадения выборок и их средних по платформам Xbox One и PC. В то же время, разница очень не велика и при выборе критического значения p-value другим распределения уже будут отличаться и нулевую гипотезу нужно будет отвергнуть.

Проверим гипотезу о том, что средние пользовательские рейтинги жанров Action и Sports разные.

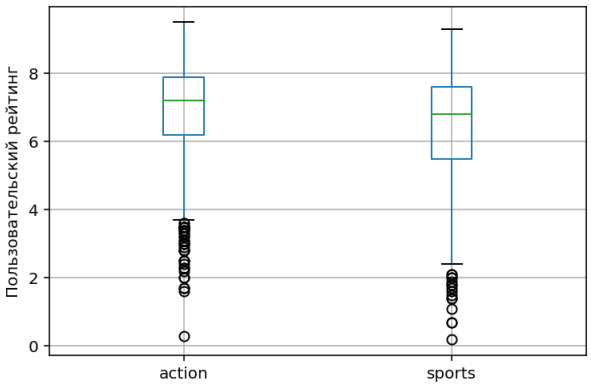
In [96]:
```
action_ratings = games.query('genre == "Action"')['user_score'].reset_index(drop=True)
```

In [97]:
```
sports_ratings = games.query('genre == "Sports"')['user_score'].reset_index(drop=True)
```

In [98]:
```
genre_ratings = pd.DataFrame({'action': action_ratings, 'sports': sports_ratings})
```

In [99]:
```
genre_ratings.boxplot();
plt.ylabel('Пользовательский рейтинг');
plt.suptitle('Распределение рейтинга пользователей для жанров Action и Sports');
```

Распределение рейтинга пользователей для жанров Action и Sports



In [100…
```
genre_ratings.describe()
```

Out[100…

	action	sports
count	1343.000000	682.000000
mean	6.881013	6.418622
std	1.359979	1.647244
min	0.300000	0.200000
25%	6.200000	5.500000
50%	7.200000	6.800000
75%	7.900000	7.600000
max	9.500000	9.300000

Аналогично предыдущей гипотезе избавимся от выбросов.

In [101…
```
action_ratings = action_ratings[action_ratings >= 3.8]
sports_ratings = sports_ratings[sports_ratings >= 2.2]
genre_ratings = pd.DataFrame({'action': action_ratings, 'sports': sports_ratings})
```

In [102…
```
genre_ratings.describe()
```

Out[102…

	action	sports

	action	sports
count	1298.000000	666.000000
mean	7.017720	6.536787
std	1.156535	1.474948
min	3.800000	2.400000
25%	6.300000	5.625000
50%	7.200000	6.800000
75%	7.900000	7.700000
max	9.500000	9.300000

Для второй гипотезы сформулируем нулевую гипотезу таким образом: средние пользовательские рейтинги жанров Action и Sports одинаковые (не отличаются). Альтернативной гипотезой будет ситуация, обратная этой.

Примем пороговое значение вероятности за 5%.

In [103…
```
alpha = 0.05
sample1 = action_ratings.to_list()
sample2 = sports_ratings.to_list()
```

In [104…
```
result = st.ttest_ind(sample1, sample2, equal_var=True)
result.pvalue
```

Out[104… `3.833782784088702e-15`

In [105…
```
result.pvalue < alpha
```

Out[105… `True`

Нулевую гипотезу нужно отвергнуть. Получается, что вероятность совпадения выборок и их средних по жанрам Action и Sports очень мала. Соответственно, альтернативная гипотеза о различии пользовательских оценок может быть подтверждена.

Вывод раздела 5

В этом разделе мы проверили 2 гипотезы:

- средние пользовательские рейтинги платформ Xbox One и PC одинаковые - **гипотеза подтверждена**, но p-value схож с выбранным пороговым значением и отличается только на 1 процентный пункт, поэтому требуется отдельно подробное изучение в зависимости от задачи.
- средние пользовательские рейтинги жанров Action и Sports разные - **гипотеза подтверждена**, обнаружена значительная статистическая разница

6. Общий вывод и рекомендации

В ходе проекта были проведены следующие работы:

- загружены и описаны исходные данные
- сделаны преобразования типов и обработаны пропуски
- добавлен параметр продаж по всему миру
- проанализирован выпуск игр в разные годы и продажи по платформам, выделен характерный период игровых платформ (10 лет)
- выделены набирающие популярность платформы XOne и PS4, которые интересны для планирования продаж на 2017 год
- выделены в целом популярные платформы с продажами свыше 1 млн. копий - X360, XOne, PS3, PS4
- изучено влияние на продажи отзывов критиков и пользователей, обнаружена средняя корреляция между отзывами критиков и продажами, но не обнаружена с отзывами пользователей
- выделен самый прибыльный жанр - Action и наименее прибыльный - Strategy
- составили портрет пользователя каждого региона (NA, EU, JP), разница в продажах от рейтинга ESRB не зависит от региона и основные жанры и платформы оцень схожи в регионах NA и EU и отличаются в JP
- проверили гипотезы:
  - средние пользовательские рейтинги платформ Xbox One и PC одинаковые - подтверждена
  - средние пользовательские рейтинги жанров Action и Sports разные - подтверждена

По результатам работы можно сделать следующие наблюдения и выводы:

- на периоде 10 лет возникают и сменяются другими основные игровые платформы, кроме PC, для подготовки к 2017 году стоит обратить внимание на игры для новых платформ XOne и PS4 как набирающие популярность к 2015-2016 годам.
- в промежутке с 2006 по 2016 года сохраняют популярность основные платформы, на которых возможны высокие продажи игр (свыше 1 млн. копий) - X360, PS3, игры для этих платформ также еще будут пользоваться некоторой популярностью
- с точки зрения наполнения разными играми интернет-магазина некоторую среднюю корреляцию с продажами показывают оценки критиков, а оценки пользователей имеют слишком большой разброс и могут не отражать реальность
- по жанрам игр лидирует Action как самый прибыльный и самый популярный в регионах NA и EU, за ним следуют Sports и Shooter, в JP продажи в целом игр ниже и лидирует жанр Role-Playing, это стоит учесть при закупке игр на продажу в интернет-магазин
- по пользовательским оценкам в среднем платформы XOne и PC не отличаются, но значительно отличаются оценки для Action и Sports (лидирует в среднем Action).

In [ ]: