

Table of Contents

- 1 Вывод данных.
- 2 Предобработка данных
 - 2.1 Изменение типа данных
- 3 Исследовательский анализ данных
 - 3.1 Количество выпускаемых игр в разные годы
 - 3.2 Анализ компьютерных платформ
 - 3.3 Определение актуального периода
 - 3.4 Рынок платформ для актуального периода.
 - 3.5 "Ящик с усами"
 - 3.5.1 Выводы по графику
 - 3.6 Анализ платформ PS4, XOne, 3DS
 - 3.7 Распределение игр по жанрам
- 4 Портрет пользователя каждого региона
 - 4.1 Самые популярные платформы (топ-5) по регионам
 - 4.2 Самые популярные жанры (топ-5) по регионам
 - 4.3 Влияние рейтинга ESRB на продажи в отдельном регионе
- 5 Проверка гипотез
- 6 Общий вывод

Комментарий ревьюера

Привет, Вячеслав :) Спасибо, что прислал задание :) Меня зовут Ринат Хисамов и я буду проверять твой проект. Предлагаю обращаться друг к другу на ты. Так нам будет гораздо проще и удобней общаться Мои комментарии обозначены пометкой **Комментарий ревьюера**. Далее в файле сможешь найти их в похожих ячейках (если фон комментария зелёный — всё сделано правильно (✓), рекомендации таким же цветом. Отдельным цветом — блок ссылок (примеры ниже, 🍕). Оранжевым или светло желтым рекомендации, которые, хоть и не обязательны, но точно сделают ревью лучше. (⚠); красный комментарий: код, график или вывод стоит переделать (✗)). Не удаляй все эти комментарии и постарайся учесть их в ходе выполнения данного проекта. Будет замечательно, если добавишь свои комментарии и пояснения 🔥. Поехали 🚶

Сборный проект №1

Исследование работы интернет-магазина компьютерных игр

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».

ВЫВОД данных.

```
In [1]: import pandas as pd  
import matplotlib.pyplot as plt  
import numpy as np  
import seaborn as sns  
from scipy import stats as st
```

✓ Комментарий ревьюера

Молодец, что делишь блоки загрузки библиотек и датасета, в случае необходимости добавления новых библиотек не придется загружать весь датасет заново и перезапускать проект целиком

```
In [2]: data = pd.read_csv('https://code.s3.yandex.net/datasets/games.csv')
```

```
In [3]: pd.set_option('display.max_columns', None)
```

```
In [4]: data.head(50)
```

Out[4]:

	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
10	Nintendogs	DS	2005.0	Simulation	9.05	10.95	1.93	2.74	NaN	NaN	NaN
11	Mario Kart DS	DS	2005.0	Racing	9.71	7.47	4.13	1.90	91.0	8.6	E
12	Pokemon Gold/Pokemon Silver	GB	1999.0	Role-Playing	9.00	6.18	7.20	0.71	NaN	NaN	NaN
13	Wii Fit	Wii	2007.0	Sports	8.92	8.03	3.60	2.15	80.0	7.7	E
14	Kinect Adventures!	X360	2010.0	Misc	15.00	4.89	0.24	1.69	61.0	6.3	E
15	Wii Fit Plus	Wii	2009.0	Sports	9.01	8.49	2.53	1.77	80.0	7.4	E
16	Grand Theft Auto V	PS3	2013.0	Action	7.02	9.09	0.98	3.96	97.0	8.2	M
17	Grand Theft Auto: San Andreas	PS2	2004.0	Action	9.43	0.40	0.41	10.57	95.0	9	M
18	Super Mario World	SNES	1990.0	Platform	12.78	3.75	3.54	0.55	NaN	NaN	NaN
19	Brain Age: Train Your Brain in Minutes a Day	DS	2005.0	Misc	4.74	9.20	4.16	2.04	77.0	7.9	E
20	Pokemon Diamond/Pokemon Pearl	DS	2006.0	Role-Playing	6.38	4.46	6.04	1.36	NaN	NaN	NaN

		Name	Platform	Year_of_Release	Genre	NA_sales	EU_sales	JP_sales	Other_sales	Critic_Score	User_Score	Rating
21		Super Mario Land	GB	1989.0	Platform	10.83	2.71	4.18	0.42	NaN	NaN	NaN
22		Super Mario Bros. 3	NES	1988.0	Platform	9.54	3.44	3.84	0.46	NaN	NaN	NaN
23		Grand Theft Auto V	X360	2013.0	Action	9.66	5.14	0.06	1.41	97.0	8.1	M
24		Grand Theft Auto: Vice City	PS2	2002.0	Action	8.41	5.49	0.47	1.78	95.0	8.7	M
25		Pokemon Ruby/Pokemon Sapphire	GBA	2002.0	Role-Playing	6.06	3.90	5.38	0.50	NaN	NaN	NaN
26		Brain Age 2: More Training in Minutes a Day	DS	2005.0	Puzzle	3.43	5.35	5.32	1.18	77.0	7.1	E
27		Pokemon Black/Pokemon White	DS	2010.0	Role-Playing	5.51	3.17	5.65	0.80	NaN	NaN	NaN
28		Gran Turismo 3: A-Spec	PS2	2001.0	Racing	6.85	5.09	1.87	1.16	95.0	8.4	E
29		Call of Duty: Modern Warfare 3	X360	2011.0	Shooter	9.04	4.24	0.13	1.32	88.0	3.4	M
30		Pokémon Yellow: Special Pikachu Edition	GB	1998.0	Role-Playing	5.89	5.04	3.12	0.59	NaN	NaN	NaN
31		Call of Duty: Black Ops 3	PS4	2015.0	Shooter	6.03	5.86	0.36	2.38	NaN	NaN	NaN
32		Call of Duty: Black Ops	X360	2010.0	Shooter	9.70	3.68	0.11	1.13	87.0	6.3	M
33		Pokemon X/Pokemon Y	3DS	2013.0	Role-Playing	5.28	4.19	4.35	0.78	NaN	NaN	NaN
34		Call of Duty: Black Ops II	PS3	2012.0	Shooter	4.99	5.73	0.65	2.42	83.0	5.3	M
35		Call of Duty: Black Ops II	X360	2012.0	Shooter	8.25	4.24	0.07	1.12	83.0	4.8	M
36		Call of Duty: Modern Warfare 2	X360	2009.0	Shooter	8.52	3.59	0.08	1.28	94.0	6.3	M
37		Call of Duty: Modern Warfare 3	PS3	2011.0	Shooter	5.54	5.73	0.49	1.57	88.0	3.2	M
38		Grand Theft Auto III	PS2	2001.0	Action	6.99	4.51	0.30	1.30	97.0	8.5	M
39		Super Smash Bros. Brawl	Wii	2008.0	Fighting	6.62	2.55	2.66	1.01	93.0	8.9	T

		Name	Platform	Year_of_Release	Genre	NA_sales	EU_sales	JP_sales	Other_sales	Critic_Score	User_Score	Rating
40		Mario Kart 7	3DS	2011.0	Racing	5.03	4.02	2.69	0.91	85.0	8.2	E
41		Call of Duty: Black Ops	PS3	2010.0	Shooter	5.99	4.37	0.48	1.79	88.0	6.4	M
42		Grand Theft Auto V	PS4	2014.0	Action	3.96	6.31	0.38	1.97	97.0	8.3	M
43		Animal Crossing: Wild World	DS	2005.0	Simulation	2.50	3.45	5.33	0.86	86.0	8.7	E
44		Halo 3	X360	2007.0	Shooter	7.97	2.81	0.13	1.21	94.0	7.8	M
45		Super Mario 64	N64	1996.0	Platform	6.91	2.85	1.91	0.23	NaN	NaN	NaN
46		Pokemon HeartGold/Pokemon SoulSilver	DS	2009.0	Action	4.34	2.71	3.96	0.76	NaN	NaN	NaN
47		Pokemon Omega Ruby/Pokemon Alpha Sapphire	3DS	2014.0	Role-Playing	4.35	3.49	3.10	0.74	NaN	NaN	NaN
48		Gran Turismo 4	PS2	2004.0	Racing	3.01	0.01	1.10	7.53	89.0	8.5	E
49		Super Mario Galaxy	Wii	2007.0	Platform	6.06	3.35	1.20	0.74	97.0	8.9	E

In [5]: `data.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
```

На первом этапе можно сказать о количестве строк датасета - 16715, столбец с наибольшим количеством пропусков - `Critic_Score`. Что касается типов данных, столбец `User_score` следует привести к вещественному типу. Также столбец `Year_of_Release` лучше бы привести к типу `int`, но наличие пропусков не даст этого сделать сразу. Также название столбцов следует изменить на нижний регистр.

✓ Комментарий ревьюера

загрузка и обзор данных проведены корректно

Предобработка данных

```
In [6]: data.columns = data.columns.str.lower() # приводим название столбцов к нижнему регистру
```

✓ Комментарий ревьюера

стоит привести к нижнему регистру и содержимое категориальных столбцов, в т.ч. с целью поиска дубликатов

Изменение типа данных

В столбце `user_score` присутствуют значения `tbd`, что расшифровывается как to be determined. Данные игры еще только ожидают своей оценки и присваивать какие-либо значения не корректно. В то же время данные значения не позволяют изменить тип данных на вещественный. Поменяем значения `tbd` на `Nan` и изменим тип данных на `float`

```
In [7]: data.loc[data['user_score'] == 'tbd', 'user_score'] = "NAN"
```

✓ Комментарий ревьюера

Верно, по своей сути `tbd` и является `Nan`. Отлично, что определяешь неявные пропущенные значения.

```
In [8]: data['user_score'] = data['user_score'].astype(float)
```

Как уже говорилось ранее, столбец `year_of_release` привести к типу `int` не получится, из-за наличия пропусков.

```
In [9]: data['year_of_release'].isna().sum() # количество пропусков в столбце "year_of_release"
```

```
Out[9]: 269
```

```
In [10]: data.loc[data['year_of_release'].isna()].head(50)
```

Out[10]:

		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
	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
	678	LEGO Indiana Jones: The Original Adventures	Wii	NaN	Action	1.51	0.61	0.00	0.21	78.0	6.6	E10+
	719	Call of Duty 3	Wii	NaN	Shooter	1.17	0.84	0.00	0.23	69.0	6.7	T
	805	Rock Band	Wii	NaN	Misc	1.33	0.56	0.00	0.20	80.0	6.3	T
	1131	Call of Duty: Black Ops	PC	NaN	Shooter	0.58	0.81	0.00	0.23	81.0	5.2	M
	1142	Rock Band	PS3	NaN	Misc	0.99	0.41	0.00	0.22	92.0	8.4	T
	1301	Triple Play 99	PS	NaN	Sports	0.81	0.55	0.00	0.10	NaN	NaN	NaN
	1506	Adventure	2600	NaN	Adventure	1.21	0.08	0.00	0.01	NaN	NaN	NaN
	1538	LEGO Batman: The Videogame	PSP	NaN	Action	0.57	0.44	0.00	0.27	73.0	7.4	E10+
	1585	Combat	2600	NaN	Action	1.17	0.07	0.00	0.01	NaN	NaN	NaN
	1609	LEGO Harry Potter: Years 5-7	Wii	NaN	Action	0.69	0.42	0.00	0.12	76.0	7.8	E10+
	1650	NASCAR Thunder 2003	PS2	NaN	Racing	0.60	0.46	0.00	0.16	84.0	8.7	E
	1699	Hitman 2: Silent Assassin	XB	NaN	Action	0.76	0.38	0.00	0.05	84.0	8.0	M

		name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating
1840		Rock Band	PS2	NaN	Misc	0.71	0.06	0.00	0.35	82.0	6.8	T
1984		Legacy of Kain: Soul Reaver	PS	NaN	Action	0.58	0.40	0.00	0.07	91.0	9.0	T
2010		Donkey Kong Land III	GB	NaN	Platform	0.68	0.31	0.00	0.04	NaN	NaN	NaN
2106		Air-Sea Battle	2600	NaN	Shooter	0.91	0.06	0.00	0.01	NaN	NaN	NaN
2108		Suikoden III	PS2	NaN	Role-Playing	0.29	0.23	0.38	0.08	86.0	7.7	T
2132		LEGO Harry Potter: Years 5-7	X360	NaN	Action	0.51	0.37	0.00	0.09	77.0	7.9	E10+
2157		Wheel of Fortune	PS2	NaN	Misc	0.47	0.36	0.00	0.12	NaN	NaN	E
2169		Yakuza 4	PS3	NaN	Action	0.15	0.13	0.63	0.05	78.0	8.0	M
2273		LEGO Harry Potter: Years 5-7	PS3	NaN	Action	0.36	0.41	0.00	0.15	76.0	8.3	E10+
2281		Namco Museum	XB	NaN	Misc	0.77	0.11	0.00	0.04	59.0	NaN	E
2361		Rhythm Heaven	Wii	NaN	Misc	0.11	0.00	0.77	0.01	NaN	NaN	NaN
2453		The Lord of the Rings: War in the North	X360	NaN	Action	0.52	0.24	0.00	0.08	61.0	7.4	M
2479		Madden NFL 07	PSP	NaN	Sports	0.77	0.03	0.00	0.04	78.0	6.6	E
2492		MLB SlugFest 20-03	PS2	NaN	Sports	0.41	0.32	0.00	0.11	77.0	8.2	E
2522		The Lord of the Rings: War in the North	PS3	NaN	Action	0.25	0.42	0.01	0.13	63.0	7.0	M
2536		Shaun White Snowboarding	X360	NaN	Sports	0.48	0.25	0.00	0.08	60.0	7.6	T
2572		PES 2009: Pro Evolution Soccer	PSP	NaN	Sports	0.04	0.33	0.26	0.17	NaN	NaN	NaN
2773		WarioWare: Twisted!	GBA	NaN	Puzzle	0.16	0.06	0.50	0.02	NaN	NaN	NaN
2849		Madden NFL 11	Wii	NaN	Sports	0.68	0.00	0.00	0.04	75.0	5.4	E

		name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating
2969	Test Drive Unlimited 2		X360		Racing	0.30	0.31	0.00	0.07	68.0	6.4	T
3024	The Chronicles of Narnia: The Lion, The Witch ...		GBA		Action	0.48	0.18	0.00	0.01	66.0	6.8	E
3081	LEGO Harry Potter: Years 5-7		DS		Action	0.34	0.25	0.00	0.07	69.0	NaN	E10+
3187	Monster Hunter 2		PS2		Role-Playing	0.00	0.00	0.63	0.00	NaN	NaN	NaN
3223	Metal Gear Solid 2: Substance		XB		Action	0.38	0.22	0.00	0.03	87.0	8.5	M
3233	Test Drive Unlimited 2		PS3		Racing	0.16	0.34	0.01	0.12	70.0	6.1	T
3289	Advance Wars: Days of Ruin		DS		Strategy	0.43	0.12	0.00	0.05	86.0	8.7	E10+
3352	The Golden Compass		Wii		Action	0.26	0.28	0.00	0.07	35.0	6.8	E10+
3413	Madden NFL 06		X360		Sports	0.54	0.00	0.01	0.03	74.0	4.9	E
3459	NASCAR: Dirt to Daytona		PS2		Racing	0.28	0.22	0.00	0.07	84.0	8.8	E
3486	Madden NFL 2002		XB		Sports	0.53	0.02	0.00	0.03	90.0	8.1	E
3704	Def Jam: Fight for NY		XB		Fighting	0.43	0.10	0.00	0.02	84.0	8.5	M

In [11]: `data.loc[(data['genre'] == 'Sports') & (data['year_of_release'].isna())]`

Out[11]:

		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
1301		Triple Play 99	PS		NaN	Sports	0.81	0.55	0.00	0.10	NaN	NaN	NaN
2479		Madden NFL 07	PSP		NaN	Sports	0.77	0.03	0.00	0.04	78.0	6.6	E
2492		MLB SlugFest 20-03	PS2		NaN	Sports	0.41	0.32	0.00	0.11	77.0	8.2	E
2536		Shaun White Snowboarding	X360		NaN	Sports	0.48	0.25	0.00	0.08	60.0	7.6	T
2572		PES 2009: Pro Evolution Soccer	PSP		NaN	Sports	0.04	0.33	0.26	0.17	NaN	NaN	NaN
2849		Madden NFL 11	Wii		NaN	Sports	0.68	0.00	0.00	0.04	75.0	5.4	E
3413		Madden NFL 06	X360		NaN	Sports	0.54	0.00	0.01	0.03	74.0	4.9	E
3486		Madden NFL 2002	XB		NaN	Sports	0.53	0.02	0.00	0.03	90.0	8.1	E
3739		NBA Street Vol. 2	GC		NaN	Sports	0.41	0.11	0.00	0.01	88.0	8.1	E
3883		Fishing Derby	2600		NaN	Sports	0.48	0.03	0.00	0.01	NaN	NaN	NaN
4205		Tiger Woods PGA Tour 07	Wii		NaN	Sports	0.43	0.00	0.00	0.04	71.0	6.9	E
4635		NHL Slapshot	Wii		NaN	Sports	0.39	0.00	0.00	0.02	76.0	8.1	E
4775		NFL GameDay 2003	PS2		NaN	Sports	0.20	0.15	0.00	0.05	60.0	NaN	E
5156		NBA Live 2003	XB		NaN	Sports	0.31	0.04	0.00	0.01	82.0	8.8	E
5655		All-Star Baseball 2005	PS2		NaN	Sports	0.16	0.12	0.00	0.04	72.0	8.6	E
5889		NBA Live 2003	GC		NaN	Sports	0.23	0.06	0.00	0.01	82.0	8.2	E
6624		College Hoops 2K6	PS2		NaN	Sports	0.12	0.10	0.00	0.03	77.0	7.3	E
6636		Jonah Lomu Rugby Challenge	PS3		NaN	Sports	0.00	0.19	0.00	0.06	64.0	NaN	E
6999		Tony Hawk's Downhill Jam	Wii		NaN	Sports	0.21	0.00	0.00	0.02	69.0	6.2	E10+
7108		Big Beach Sports 2	Wii		NaN	Sports	0.09	0.11	0.00	0.02	NaN	NaN	E

		name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating
7387		Move Fitness	PS3		Nan	Sports	0.00	0.16	0.00	0.05	Nan	Nan
7605		Famista 64	N64		Nan	Sports	0.00	0.00	0.17	0.03	Nan	Nan
8067		Backbreaker	X360		Nan	Sports	0.17	0.00	0.00	0.01	54.0	7.6
8197		NBA Starting Five	PS2		Nan	Sports	0.09	0.07	0.00	0.02	53.0	7.3
8260		Backbreaker	PS3		Nan	Sports	0.16	0.00	0.00	0.01	58.0	7.0
8740		Home Run	2600		Nan	Sports	0.14	0.01	0.00	0.00	Nan	Nan
8918		All-Star Baseball 2005	XB		Nan	Sports	0.11	0.03	0.00	0.01	75.0	8.8
9380		Transworld Surf	XB		Nan	Sports	0.10	0.03	0.00	0.00	76.0	Nan
9817		Street Hoops	GC		Nan	Sports	0.09	0.02	0.00	0.00	56.0	7.3
9876		Major League Baseball 2K6	PSP		Nan	Sports	0.11	0.00	0.00	0.01	69.0	Nan
10486		Atsumare! Power Pro Kun no DS Koushien	DS		Nan	Sports	0.00	0.00	0.10	0.00	Nan	Nan
11550		Get Fit with Mel B	X360		Nan	Sports	0.00	0.06	0.00	0.01	57.0	Nan
12959		Mountain Bike Adrenaline	PS2		Nan	Sports	0.03	0.02	0.00	0.01	Nan	Nan
13792		NHL Hitz Pro	GC		Nan	Sports	0.03	0.01	0.00	0.00	81.0	7.8
14141		Major League Baseball 2K8	PSP		Nan	Sports	0.03	0.00	0.00	0.00	63.0	Nan
15338		Mario Tennis	3DS		Nan	Sports	0.00	0.00	0.02	0.00	Nan	Nan
15675		Cabela's Alaskan Adventure	PS2		Nan	Sports	0.01	0.01	0.00	0.00	Nan	Nan
15953		PDC World Championship Darts 2008	DS		Nan	Sports	0.01	0.00	0.00	0.00	Nan	Nan
16079		Football Manager 2007	X360		Nan	Sports	0.00	0.01	0.00	0.00	Nan	Nan
16373		PDC World Championship Darts 2008	PSP		Nan	Sports	0.01	0.00	0.00	0.00	43.0	Nan
											E10+	

Всего пропусков в столбце 269, т.е. около 2% от общего числа данных. Удалять в самом начале исследования не правильно, пока неизвестно насколько эти данные важны. Заполнять, опираясь на таблицу в целом, например медианным или средним значением также не правильно, поскольку год выпуска - это как индивидуальная характеристика игры. Можно обратить внимание, что в названиях некоторых спортивных игр присутствует год. С помощью таблицы выше можно сказать, что таких игр всего 16, что ничтожно мало относительно всего количества данных. Поэтому оставим вещественный тип данных в столбце `year_of_release`, без заполнения пропусков и удаления строк.

```
In [12]: data.isna().sum() # общее количество пропусков в каждом столбце
```

```
Out[12]: 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
```

Методом `isna()` получаем информацию о количестве пропусков в каждом из столбцов. Каждая строка датасета является уникальной характеристикой компьютерной игры и найти методы унификации, по которым можно заполнить пропуски, довольно проблематично. К примеру, наибольшее количество пропусков в столбцах `critic_score`, `user_score`. Значения в каждом из этих столбцов - это оценка именно этой конкретной игры, и заполнять пропуски опираясь на данные датасета, анализируя данные других игр - не правильно и может лишь привести к искажению результатов, получаемых при анализе. Аналогичная логика и с данными, `rating`. То есть, текущий анализ чувствителен буквально ко всем данным, каждой ячейке датасета и применять общие методы, вроде среднего и медианного значения, или делать выборки и сводные таблицы и по ним заполнять пропуски, все-таки неправильно

```
In [13]: len(data.query('user_score.isna() & critic_score.isna() & rating.isna()'))
```

```
Out[13]: 6667
```

В 6667 строках (около 40% от всех данных), отсутствуют сразу все значения о рейтингах, оценках пользователей и критиков - как подтверждение, что это слишком большой массив данных для изменения. Это может привести лишь искажению результатов анализа. Возможно эти игры еще ожидают своих оценок.

Пропуски присутствуют также в столбцах `name`, `genre`. Учитывая, что их общее количество составляет сотые доли от общего числа строк, удалим эти значения. Перед удалением проверим, не содержат ли эти строки значений, удаление которых может повлиять на результаты анализа. (например, крупные доли продаж)

```
In [14]: data.loc[(data['name'].isna()) | (data['genre'].isna())]
```

```
Out[14]:      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
```

```
In [15]: data = data.dropna(subset=['name', 'genre'])
```

```
In [16]: # Создаем столбец total_sales, в котором указана сумма продаж по всем регионам
data['total_sales'] = data[['na_sales', 'eu_sales', 'jp_sales', 'other_sales']].sum(axis=1)
```

```
In [17]: data.head()
```

```
Out[17]:      name  platform  year_of_release  genre  na_sales  eu_sales  jp_sales  other_sales  critic_score  user_score  rating  total_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
```

```
In [18]: # check  
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
Int64Index: 16713 entries, 0 to 16714  
Data columns (total 12 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    
 11  total_sales      16713 non-null   float64  
dtypes: float64(8), object(4)  
memory usage: 1.7+ MB
```

✖ Комментарий ревьюера

Рейтинг ESRB — категориальные данные, стоит внимательно взглянуть на содержимое и предложить чем заполнить пропуски, возможно это поможет найти необычные различия в поведении клиентов, допом можно подумать над сокращением количества категорий

Т.к. записи с пропусками не учитываются при группировке данных, мы не сможем выявить реальный портрет клиента

```
In [19]: data['rating'].unique()
```

```
Out[19]: array(['E', nan, 'M', 'T', 'E10+', 'K-A', 'AO', 'EC', 'RP'], dtype=object)
```

```
In [20]: data['rating'].value_counts()
```

```
Out[20]: E      3990  
T      2961  
M      1563  
E10+    1420  
EC       8  
K-A       3  
RP       3  
AO       1  
Name: rating, dtype: int64
```

Выше уже находили количество пропусков по столбцам, в частности в столбце `rating` пропусков 6766 (но две строки уже удалили). ESRB не единственная организация, выставляющая рейтинги видеоиграм, и работает на рынке США, Канады и Мексики. В Европейском Союзе, Японии, Австралии также есть организации, занимающиеся выставлением рейтинга видеоиграм. По сути некорректно сравнивать разные регионы по рейтингу ESRB. Но других данных у нас нет и возрастной ценз во всех организация примерно одинаковый – для маленьких детей, детей постарше, подростков, взрослых. Пропуски могут говорить о том, что, либо игру еще не оценивали, либо данная игра в другом регионе имеет рейтинг, отличный от ESRB. В любом случае наиболее корректно дать пропускам значение ND (not determined). Таким образом эти данные не будут исключены при группировке и при этом такое заполнение не исказит результаты анализа.

```
In [21]: data['rating'] = data['rating'].fillna('ND')
```

```
In [22]: data['rating'].value_counts()
```

```
Out[22]: ND      6764  
E      3990  
T      2961  
M      1563  
E10+    1420  
EC       8  
K-A       3  
RP       3  
AO       1  
Name: rating, dtype: int64
```

По поводу сокращения количества категорий - можно категории с небольшим количеством игр объединить с более крупными категориями.

- EC Early childhood изменить на E Everyone;
- RP Rating pending изменить на ND;
- AO Adults Only изменить на M Mature;
- K-A Kids to Adults изменить на E;

```
In [23]: data = data.replace({'rating':{"EC":"E", "K-A":"E", "RP":"ND", "AO":"M"}})
```

```
In [24]: data['rating'].value_counts()
```

```
Out[24]: ND    6767  
E     4001  
T     2961  
M     1564  
E10+   1420  
Name: rating, dtype: int64
```

✓ Комментарий ревьюера в2

хорошее решение

```
data = data.replace({'rating':{"EC":"E", "K-A":"E", "RP":"ND", "AO":"M"}})
```

```
In [25]: data.isna().sum()
```

```
Out[25]: name          0  
platform        0  
year_of_release 269  
genre           0  
na_sales        0  
eu_sales        0  
jp_sales        0  
other_sales     0  
critic_score    8576  
user_score      9123  
rating          0  
total_sales     0  
dtype: int64
```

```
In [26]: data.duplicated().sum()
```

```
Out[26]: 0
```

Полных дубликатов в наборе данных нет

Также проверим на наличие дубликатов по трем столбцам: название игры, год выпуска и платформа. Причем название и год выпуска могут дублироваться, поскольку одни и те же игры выпускаются под разные платформы. Поэтому проверка пройдется по этим трем столбцам.

```
In [27]: data[data.duplicated(subset=['name', 'platform', 'year_of_release'], keep=False)]
```

```
Out[27]:   name  platform  year_of_release  genre  na_sales  eu_sales  jp_sales  other_sales  critic_score  user_score  rating  total_sales  
604    Madden      NFL 13       2012.0  Sports     2.11     0.22      0.0      0.23      83.0        5.5       E      2.56  
16230   Madden      NFL 13       2012.0  Sports     0.00     0.01      0.0      0.00      83.0        5.5       E      0.01
```

Дубли в наименовании обоснованы тем, что представляют разные реализации одной игры для разных платформ, что несёт за собой для каждой платформы свои уникальные показатели.

Исследовательский анализ данных

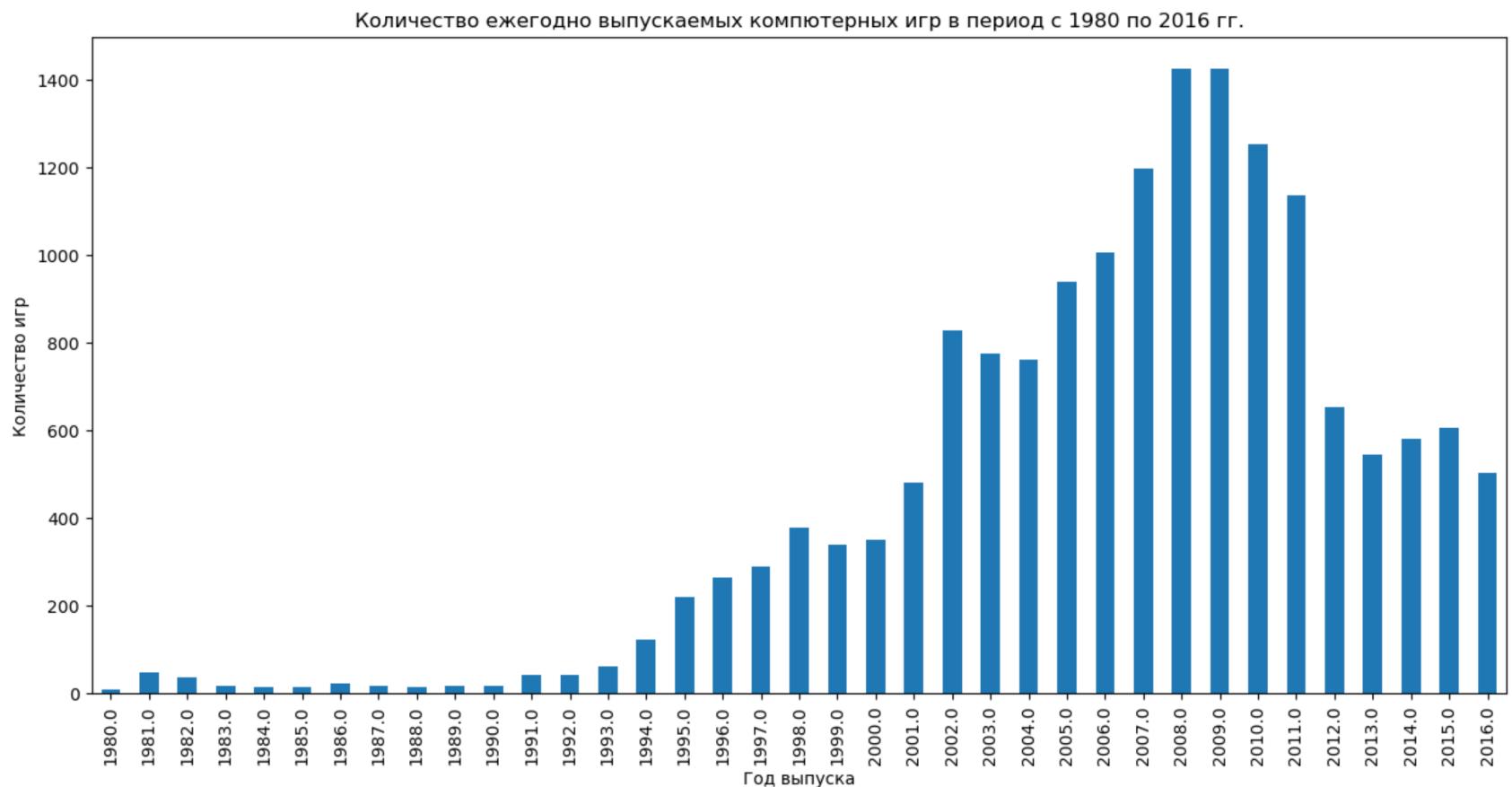
Количество выпускаемых игр в разные годы

In [28]: `data.groupby('year_of_release')['name'].count()`

```
Out[28]: year_of_release  
1980.0      9  
1981.0     46  
1982.0     36  
1983.0     17  
1984.0     14  
1985.0     14  
1986.0     21  
1987.0     16  
1988.0     15  
1989.0     17  
1990.0     16  
1991.0     41  
1992.0     43  
1993.0     60  
1994.0    121  
1995.0    219  
1996.0    263  
1997.0    289  
1998.0    379  
1999.0    338  
2000.0    350  
2001.0    482  
2002.0    829  
2003.0    775  
2004.0    762  
2005.0    939  
2006.0   1006  
2007.0   1197  
2008.0   1427  
2009.0   1426  
2010.0   1255  
2011.0   1136  
2012.0    653  
2013.0    544  
2014.0    581  
2015.0    606  
2016.0    502  
Name: name, dtype: int64
```

Для более наглядного представления построим барплот

```
In [29]: data.groupby('year_of_release')['name'].count().plot(kind='bar', figsize=(15,7))
plt.title('Количество ежегодно выпускаемых компьютерных игр в период с 1980 по 2016 гг.')
plt.xlabel('Год выпуска')
plt.ylabel('Количество игр');
```



Из полученных выше данных можно сказать, что в период с 1980 по 1992 было крайне мало выпущенных игр, что не удивительно, это только самое начало развития компьютерных технологий и по сути, индустрия компьютерных игр еще находится в стадии зародыша. С 1993 года начинается рост выпуска игр, с резким скачком в 2002 году и пиком в 2008-2009 годах. Что касается падения начиная с 2012 года - сложно дать конкретные причины, это может быть связано с тем, что производители перешли от количества к качеству и производство игр стало занимать гораздо большее время с одновременным улучшением качеством этих игр.

Анализ компьютерных платформ

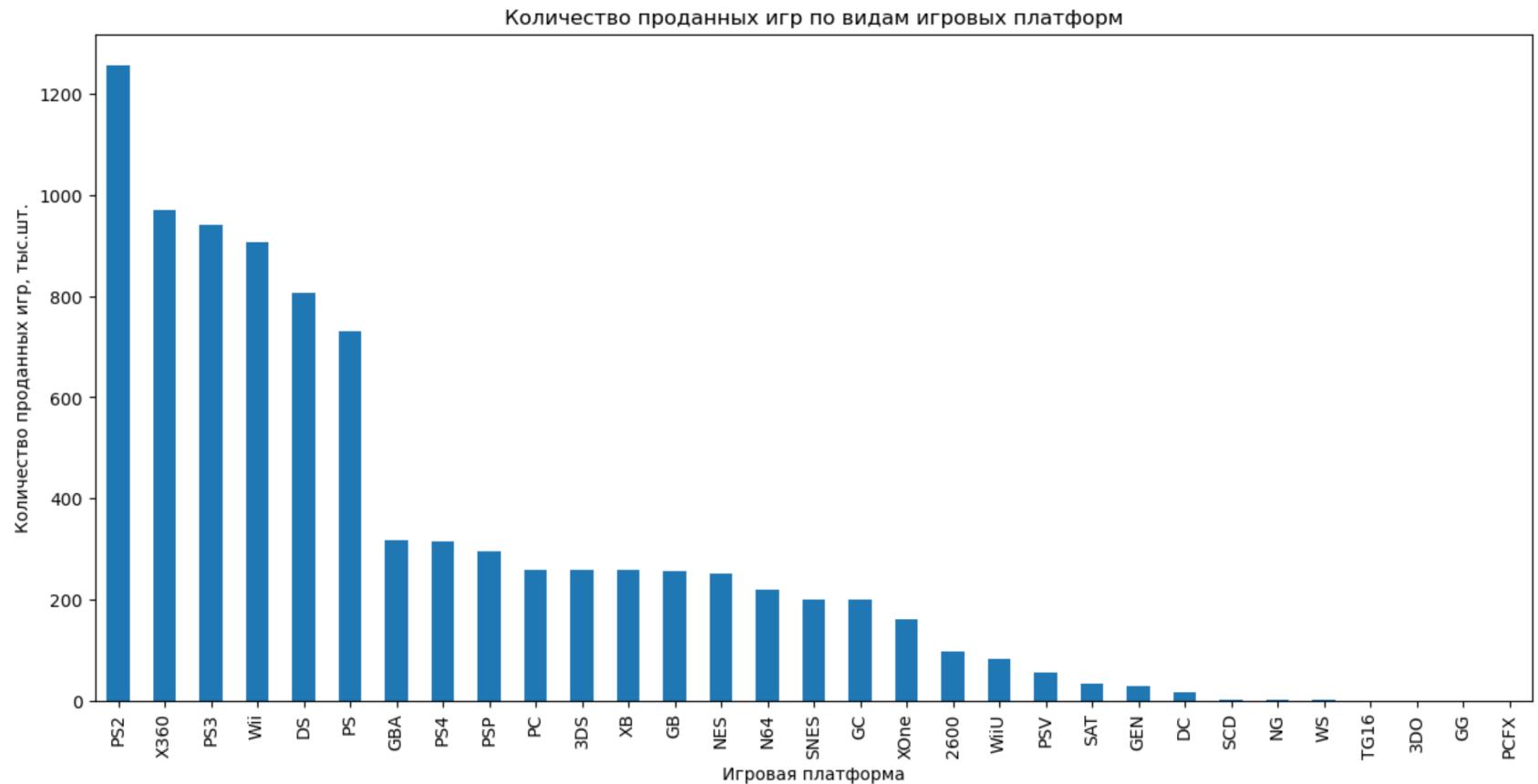
Найдем значения общих продаж для всех платформ, представленных в наборе данных

```
In [30]: plat_sales = data.groupby('platform')['total_sales'].sum().sort_values(ascending=False)
```

```
In [31]: plat_sales
```

```
Out[31]: platform
PS2      1255.77
X360     971.42
PS3      939.65
Wii       907.51
DS        806.12
PS        730.86
GBA       317.85
PS4       314.14
PSP       294.05
PC        259.52
3DS      259.00
XB        257.74
GB        255.46
NES       251.05
N64       218.68
SNES      200.04
GC        198.93
XOne     159.32
2600      96.98
WiiU      82.19
PSV       54.07
SAT       33.59
GEN       28.35
DC        15.95
SCD       1.86
NG        1.44
WS        1.42
TG16      0.16
3DO       0.10
GG        0.04
PCFX      0.03
Name: total_sales, dtype: float64
```

```
In [32]: plat_sales.plot(kind='bar', figsize=(15,7))
plt.title('Количество проданных игр по видам игровых платформ')
plt.xlabel('Игровая платформа')
plt.ylabel('Количество проданных игр, тыс.шт.');
```



Из полученных данных можно увидеть абсолютного лидера - PS2 . Для дальнейшего анализа выберем первые шесть платформ по общему количеству продаж.

```
In [33]: max_sales = data.query('platform == ["PS2", "X360", "PS3", "Wii", "DS", "PS"]')
```

```
In [34]: max_sales
```

Out[34]:

		name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating	total_sale
0		Wii Sports	Wii	2006.0	Sports	41.36	28.96	3.77	8.45	76.0	8.0	E	82.5
2		Mario Kart Wii	Wii	2008.0	Racing	15.68	12.76	3.79	3.29	82.0	8.3	E	35.5
3		Wii Sports Resort	Wii	2009.0	Sports	15.61	10.93	3.28	2.95	80.0	8.0	E	32.7
6		New Super Mario Bros.	DS	2006.0	Platform	11.28	9.14	6.50	2.88	89.0	8.5	E	29.8
7		Wii Play	Wii	2006.0	Misc	13.96	9.18	2.93	2.84	58.0	6.6	E	28.9
...	
16700		Mezase!! Tsuri Master DS	DS	2009.0	Sports	0.00	0.00	0.01	0.00	NaN	NaN	ND	0.0
16704		Plushees	DS	2008.0	Simulation	0.01	0.00	0.00	0.00	NaN	NaN	E	0.0
16709		SCORE International Baja 1000: The Official Game	PS2	2008.0	Racing	0.00	0.00	0.00	0.00	NaN	NaN	ND	0.0
16710		Samurai Warriors: Sanada Maru	PS3	2016.0	Action	0.00	0.00	0.01	0.00	NaN	NaN	ND	0.0
16711		LMA Manager 2007	X360	2006.0	Sports	0.00	0.01	0.00	0.00	NaN	NaN	ND	0.0

9422 rows × 12 columns

Далее построим сводную таблицу и график к ней, чтобы узнать на какие периоды приходятся продажи для данных платформ и их значения по годам.

```
In [35]: max_sales.pivot_table(index='year_of_release', columns='platform', values='total_sales', aggfunc='sum')
```

Out[35]:

platform	DS	PS	PS2	PS3	Wii	X360
year_of_release						
1985.0	0.02	NaN	NaN	NaN	NaN	NaN
1994.0	NaN	6.03	NaN	NaN	NaN	NaN
1995.0	NaN	35.96	NaN	NaN	NaN	NaN
1996.0	NaN	94.70	NaN	NaN	NaN	NaN
1997.0	NaN	136.17	NaN	NaN	NaN	NaN
1998.0	NaN	169.49	NaN	NaN	NaN	NaN
1999.0	NaN	144.53	NaN	NaN	NaN	NaN
2000.0	NaN	96.37	39.17	NaN	NaN	NaN
2001.0	NaN	35.59	166.43	NaN	NaN	NaN
2002.0	NaN	6.67	205.38	NaN	NaN	NaN
2003.0	NaN	2.07	184.31	NaN	NaN	NaN
2004.0	17.27	NaN	211.81	NaN	NaN	NaN
2005.0	130.14	NaN	160.66	NaN	NaN	8.25
2006.0	119.81	NaN	103.42	20.96	137.15	51.62
2007.0	146.94	NaN	75.99	73.19	152.77	95.41
2008.0	145.31	NaN	53.90	118.52	171.32	135.26
2009.0	119.54	NaN	26.40	130.93	206.97	120.29
2010.0	85.02	NaN	5.64	142.17	127.95	170.03
2011.0	26.18	NaN	0.45	156.78	59.65	143.84
2012.0	11.01	NaN	NaN	107.36	21.71	99.74
2013.0	1.54	NaN	NaN	113.25	8.59	88.58
2014.0	NaN	NaN	NaN	47.76	3.75	34.74

platform	DS	PS	PS2	PS3	Wii	X360
year_of_release						
2015.0	NaN	NaN	NaN	16.82	1.14	11.96
2016.0	NaN	NaN	NaN	3.60	0.18	1.52

Из таблицы выше можно заметить значение 0.02 для платформы DS в 1985 году. Учитывая, что продажи данной игровой консоли начались только в 2004 году, значение более чем странное.

```
In [36]: data.query('year_of_release == 1985.0 & platform == "DS"')
```

```
Out[36]:
```

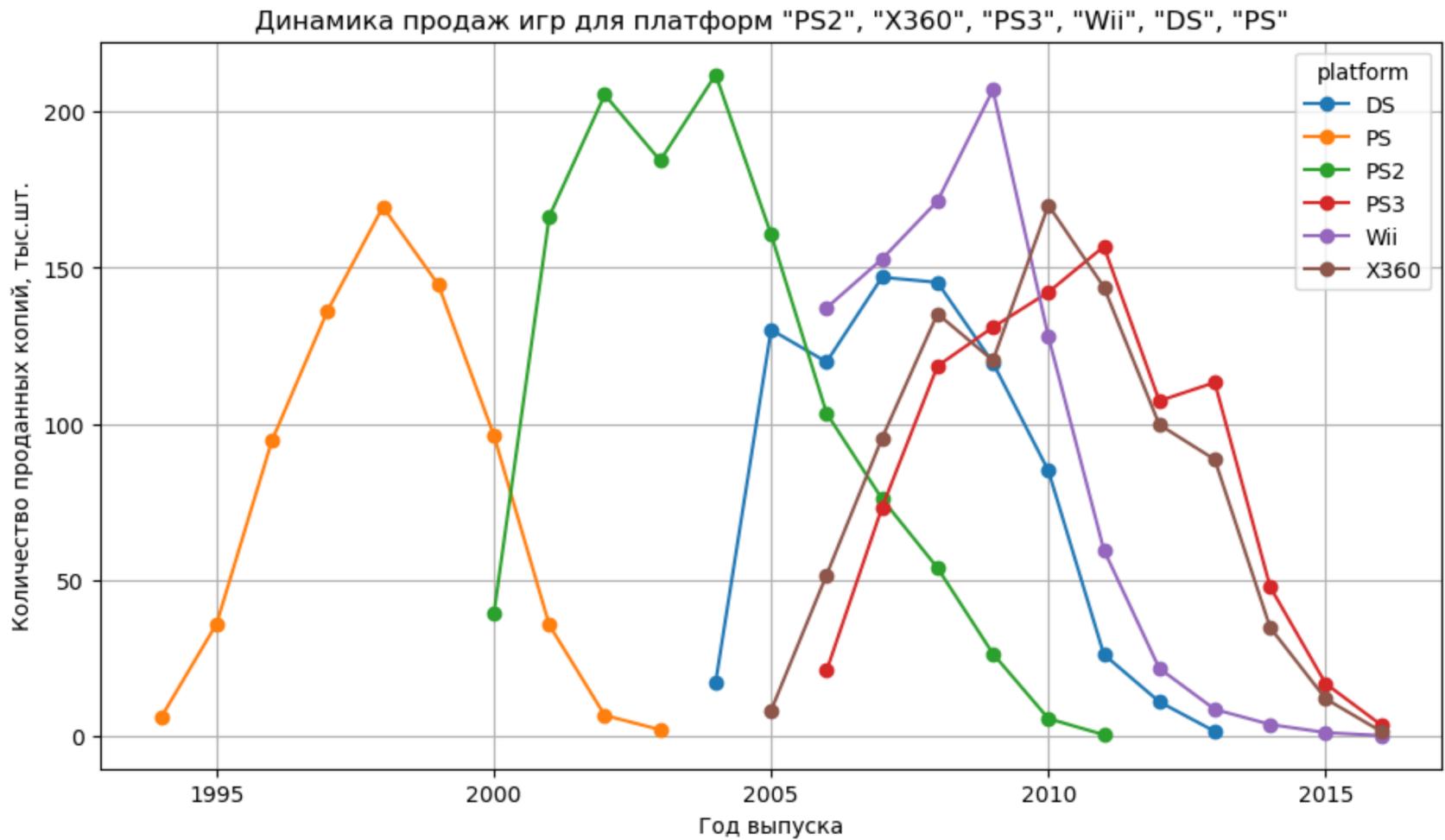
		name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating	total_sales
15957		Strongest Tokyo University Shogi DS	DS	1985.0	Action	0.0	0.0	0.02	0.0	NaN	NaN	ND	0.02

Сложно определить причины данного выброса, но от этой строки лучше избавиться

```
In [37]: data=data.drop(index=15957)
```

```
In [38]: max_sales = data.query('platform == ["PS2", "X360", "PS3", "Wii", "DS", "PS"]')
```

```
In [39]: max_sales.pivot_table(index='year_of_release', columns='platform', values='total_sales', aggfunc='sum')\
.plot(style='o-', grid=True, figsize=(11,6))
plt.title('Динамика продаж игр для платформ "PS2", "X360", "PS3", "Wii", "DS", "PS"')
plt.xlabel('Год выпуска')
plt.ylabel('Количество проданных копий, тыс.шт.');
```



Из графика отчетливо видно, как падают продажи год от года для выбранных платформ. Причина такого поведения в том, что на смену данным платформам пришли более новые. Так, например, лидером продаж за все годы (1985-2016) является PS2, но данную платформу сняли с производства в январе 2013 (данные взяты из Википедии). И строить планы дальнейших продаж на основе игр для платформ, снятых с производства, не совсем корректно.

Построим аналогичные сводную таблицу и график для платформ и количеству выпущенных для них игр.

```
In [40]: plat_amount = data.groupby('platform')[['name']].count()
```

```
In [41]: plat_amount.sort_values(ascending=False)
```

```
Out[41]: platform
PS2      2161
DS       2150
PS3      1331
Wii      1320
X360     1262
PSP      1209
PS       1197
PC       974
XB       824
GBA      822
GC       556
3DS      520
PSV      430
PS4      392
N64      319
XOne     247
SNES     239
SAT      173
WiiU     147
2600     133
NES      98
GB       98
DC       52
GEN      27
NG       12
SCD      6
WS       6
3DO      3
TG16     2
PCFX     1
GG       1
Name: name, dtype: int64
```

Далее построим сводную таблицу для лидеров по количеству игр.

```
In [42]: max_amount = data.query('platform == ["PS2", "DS", "PS3", "Wii", "X360", "PSP", "PS", "PC"]')
```

```
In [43]: max_amount.pivot_table(index='year_of_release', columns='platform', values='name', aggfunc='count')
```

Out[43]:

platform	DS	PC	PS	PS2	PS3	PSP	Wii	X360
year_of_release								
1985.0	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN
1988.0	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN
1992.0	NaN	5.0	NaN	NaN	NaN	NaN	NaN	NaN
1994.0	NaN	6.0	17.0	NaN	NaN	NaN	NaN	NaN
1995.0	NaN	2.0	99.0	NaN	NaN	NaN	NaN	NaN
1996.0	NaN	4.0	164.0	NaN	NaN	NaN	NaN	NaN
1997.0	NaN	6.0	188.0	NaN	NaN	NaN	NaN	NaN
1998.0	NaN	8.0	248.0	NaN	NaN	NaN	NaN	NaN
1999.0	NaN	7.0	200.0	NaN	NaN	NaN	NaN	NaN
2000.0	NaN	7.0	160.0	82.0	NaN	NaN	NaN	NaN
2001.0	NaN	15.0	91.0	185.0	NaN	NaN	NaN	NaN
2002.0	NaN	19.0	20.0	280.0	NaN	NaN	NaN	NaN
2003.0	NaN	33.0	3.0	256.0	NaN	NaN	NaN	NaN
2004.0	23.0	30.0	NaN	259.0	NaN	15.0	NaN	NaN
2005.0	118.0	37.0	NaN	260.0	NaN	95.0	NaN	18.0
2006.0	201.0	52.0	NaN	259.0	27.0	189.0	44.0	93.0
2007.0	376.0	62.0	NaN	214.0	90.0	133.0	185.0	123.0
2008.0	492.0	76.0	NaN	191.0	138.0	100.0	282.0	146.0
2009.0	403.0	107.0	NaN	96.0	162.0	161.0	325.0	172.0
2010.0	323.0	90.0	NaN	38.0	181.0	188.0	253.0	182.0
2011.0	153.0	139.0	NaN	7.0	215.0	139.0	143.0	206.0
2012.0	23.0	61.0	NaN	NaN	148.0	106.0	31.0	106.0

platform	DS	PC	PS	PS2	PS3	PSP	Wii	X360
year_of_release								
2013.0	8.0	38.0	NaN	NaN	126.0	54.0	12.0	75.0
2014.0	NaN	47.0	NaN	NaN	108.0	10.0	6.0	63.0
2015.0	NaN	50.0	NaN	NaN	73.0	3.0	4.0	35.0
2016.0	NaN	54.0	NaN	NaN	38.0	NaN	1.0	13.0

Из сводной таблицы можно увидеть, что к 2016 году, все лидеры показывают устойчивое падение или же вовсе выпуск игр на эти платформы прекратился. Исключение составляет лишь платформа РС. Эти данные пригодятся при выборе актуального периода. Для поиска периода, в котором появляются и исчезают платформы, построим еще одну сводную таблицу и барплот к ней. В сводной таблице будет указано, в течении скольки лет существовала платформа.

```
In [44]: total_amount = (
    data.loc[:,['platform', 'year_of_release']]
    .drop_duplicates()
    .pivot_table(index='platform', values='year_of_release', aggfunc='count')
)
```

```
In [45]: total_amount
```

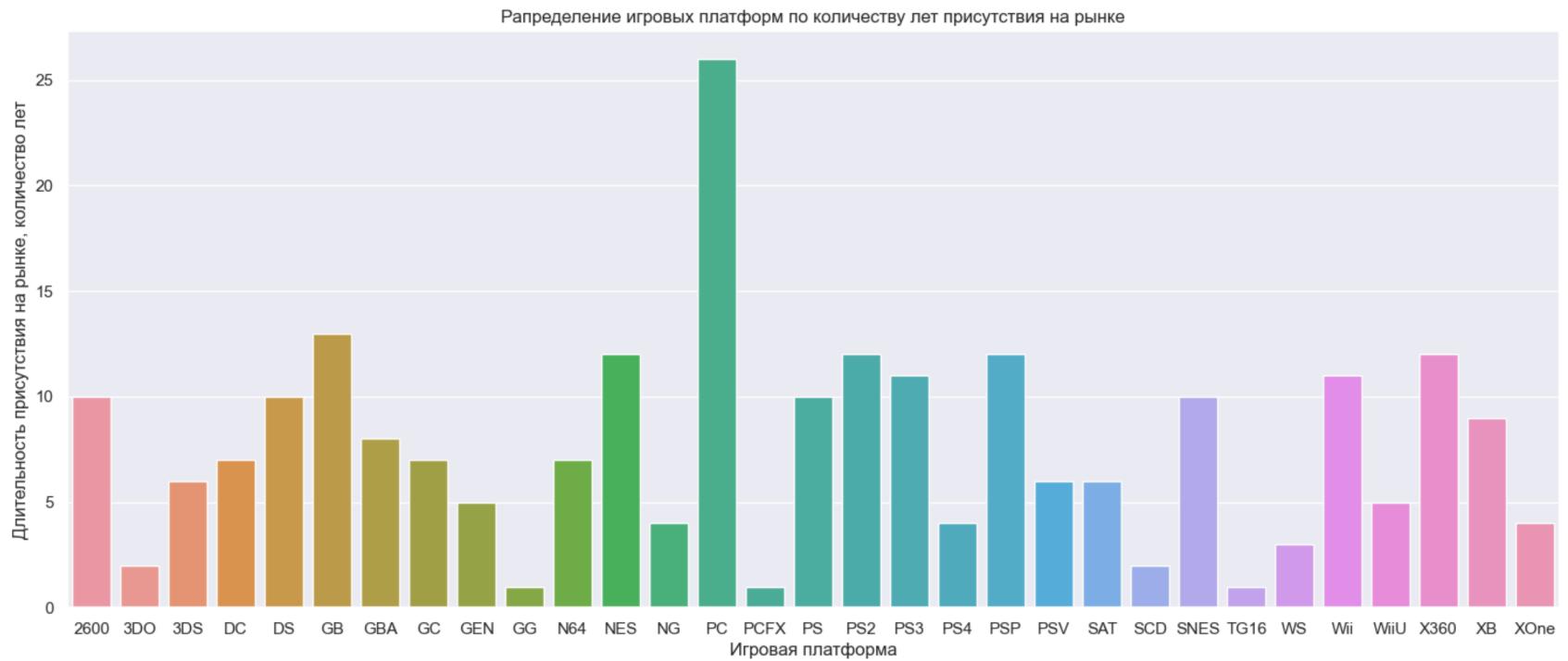
Out[45]:

year_of_release

platform	year_of_release
2600	10
3DO	2
3DS	6
DC	7
DS	10
GB	13
GBA	8
GC	7
GEN	5
GG	1
N64	7
NES	12
NG	4
PC	26
PCFX	1
PS	10
PS2	12
PS3	11
PS4	4
PSP	12
PSV	6
SAT	6

platform	year_of_release
SCD	2
SNES	10
TG16	1
WS	3
Wii	11
WiiU	5
X360	12
XB	9
XOne	4

```
In [118]:  
sns.barplot(x=total_amount.index, y='year_of_release', data = total_amount)  
sns.set(rc={'figure.figsize':(18,7)})  
plt.title('Распределение игровых платформ по количеству лет присутствия на рынке')  
plt.xlabel('Игровая платформа')  
plt.ylabel('Длительность присутствия на рынке, количество лет');
```



Из построенного барплота можно выделить в первую очередь PC, как старожил игровой индустрии. В целом довольно сложно определить именно средний период существования платформ. Обратимся к методу `describe()`

```
In [47]: total_amount.year_of_release.describe()
```

```
Out[47]: count    31.000000
mean     7.645161
std      5.063256
min     1.000000
25%     4.000000
50%     7.000000
75%    10.500000
max    26.000000
Name: year_of_release, dtype: float64
```

Можно сказать, что в среднем жизненный цикл платформы длится 7 лет.

```
In [48]: new_platforms = data.query('platform != ["PS4", "3DS", "WiiU", "XOne", "PC"]')
```

```
In [49]: total_amount_2 = (
    new_platforms.loc[:,['platform', 'year_of_release']]
    .drop_duplicates()
    .pivot_table(index='platform', values='year_of_release', aggfunc='count')
)
```

```
In [50]: total_amount_2.year_of_release.describe()
```

```
Out[50]: count    26.000000
mean     7.384615
std      3.970662
min     1.000000
25%    4.250000
50%    7.500000
75%   10.750000
max   13.000000
Name: year_of_release, dtype: float64
```

Удалив из датасета данные о платформах, срок продаж которых не завершен, а также платформу РС, которая слишком долго на рынке и вряд ли когда покинет рынок видеоигр, вновь получили значение среднего жизненного цикла платформы в районе 7 лет.

Определение актуального периода

Выше уже было отмечено, что для платформ, лидирующих по количеству продаж в целом за период с 1980 по 2016 года, к 2016 году либо не выпускаются игры, либо имеют тенденцию к сокращению их числа. Поэтому далее построим сводную таблицу за весь период и посмотрим на последние 7 лет. Так мы увидим все актуальные к 2016 году платформы.

```
In [51]: data.pivot_table(index='year_of_release', columns='platform', values='name', aggfunc='count').tail(7)
```

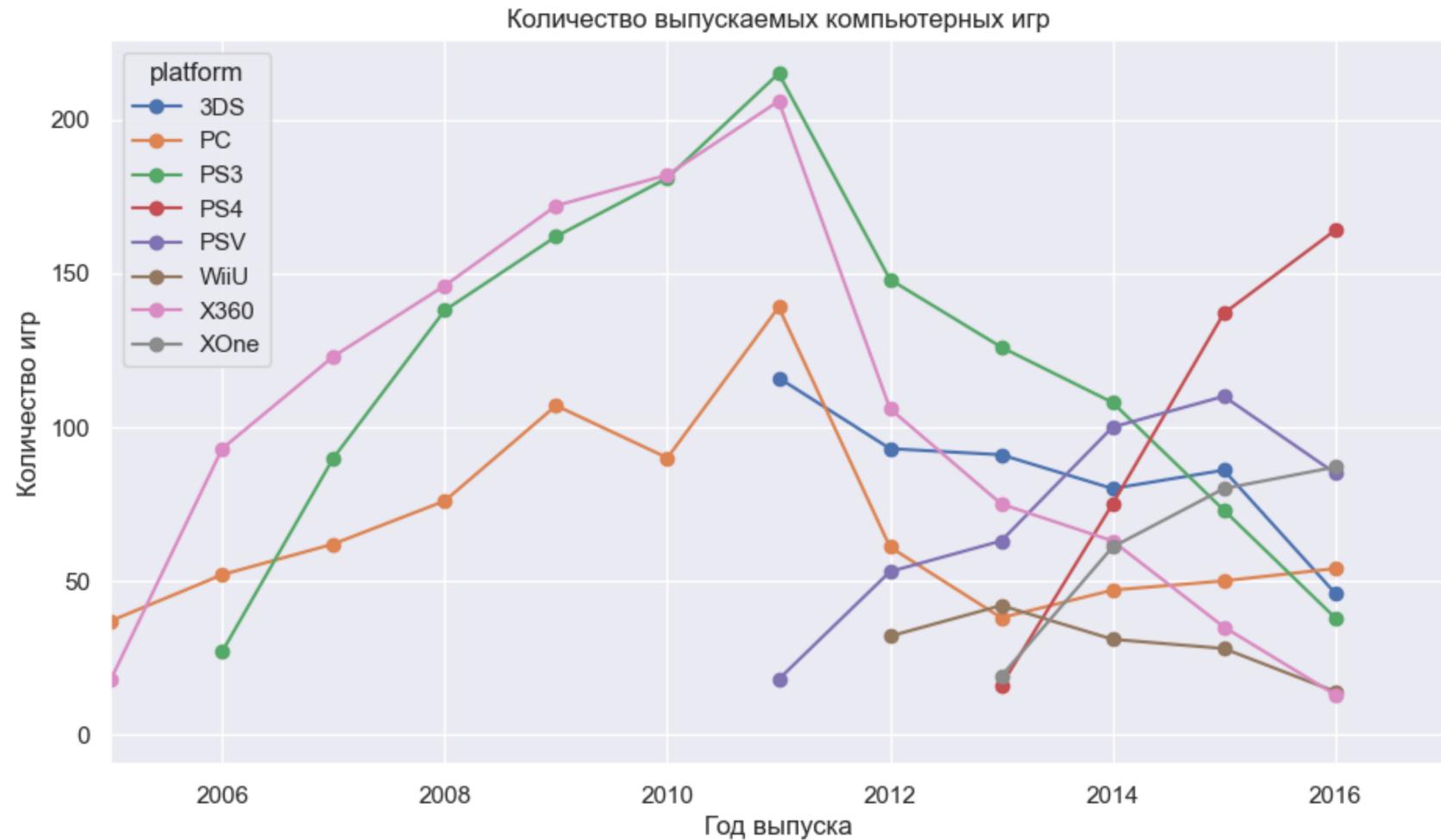
Out[51]:

	platform	2600	3DO	3DS	DC	DS	GB	GBA	GC	GEN	GG	N64	NES	NG	PC	PCFX	PS	PS2	PS3	PS4	PSV
year_of_release																					
2010.0	NaN	NaN	NaN	NaN	NaN	323.0	NaN	90.0	NaN	NaN	38.0	181.0	NaN	188.0							
2011.0	NaN	NaN	116.0	NaN	153.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	139.0	NaN	NaN	7.0	215.0	NaN	139.0
2012.0	NaN	NaN	93.0	NaN	23.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	61.0	NaN	NaN	NaN	148.0	NaN	106.0
2013.0	NaN	NaN	91.0	NaN	8.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	38.0	NaN	NaN	NaN	126.0	16.0	54.0
2014.0	NaN	NaN	80.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	47.0	NaN	NaN	NaN	108.0	75.0	10.0
2015.0	NaN	NaN	86.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	50.0	NaN	NaN	NaN	73.0	137.0	3.0
2016.0	NaN	NaN	46.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	54.0	NaN	NaN	NaN	38.0	164.0	NaN

Далее построим график для сводной таблицы, в которой фигурируют все платформы, для которых продавались видеоигры в 2016 году

In [52]: `current_platforms = data.query('platform == ["3DS", "PC", "PS3", "PS4", "PSV", "WiiU", "X360", "XOne"]')`

In [53]: `current_platforms.pivot_table(index='year_of_release', columns='platform', values='name', aggfunc='count')\n.plot(style='o-', xlim=(2005, 2017), grid=True, figsize=(11,6));\nplt.title('Количество выпускаемых компьютерных игр')\nplt.xlabel('Год выпуска')\nplt.ylabel('Количество игр');`



Далее уберем из таблицы платформы, показывающие устойчивое падение, или платформы, на смену которым пришло новое поколение

```
In [54]: current_platforms_2 = data.query('platform == ["3DS", "PC", "PS4", "WiiU", "XOne"]')
```

```
In [55]: current_platforms_2.pivot_table(index='year_of_release', columns='platform', values='name', aggfunc='count')\n.plot(style='o-', xlim=(2010, 2017), grid=True, figsize=(11,6));\nplt.title('Количество выпускаемых компьютерных игр')\nplt.xlabel('Год выпуска')\nplt.ylabel('Количество игр');
```



В качестве актуального периода примем 2014-2016гг. В 2013 году только поступили в продажу новые консоли PS4 и XboxOne и продажи игр для них лучше посмотреть через какое-то время после выхода на рынок (продажи могли быть как слишком высоким из-за больших ожиданий, так и неоправданно низкими, так как покупатели могли ждать первых отзывов на них и стоит ли их покупать). В то же время таким образом исключим отжившие платформы, такие как DS

<div class="alert alert-warning", style="border:solid coral 3px; padding: 20px"> **Комментарий ревьюера**

После представления общей картины стоит выделить на отдельном графике 5-7 самых важных категорий (платформ) для детального анализа

Рынок платформ для актуального периода.

```
In [56]: # relevant_data = data.query('year_of_release >= 2013')
```

```
In [57]: relevant_data = data.query('year_of_release >= 2014')
```

<div class="alert alert-warning", style="border:solid coral 3px; padding: 20px"> **Комментарий ревьюера**

Для целей прогнозирования продаж на следующий год даже в традиционных бизнесах редко берут данные более чем за 2-3 года. А в такой динамично меняющейся индустрии, как компьютерные игры и вовсе не стоит брать слишком большой временной интервал - иначе обязательно захватишь уже отжившие тренды. Но и слишком короткий период тоже брать не стоит.

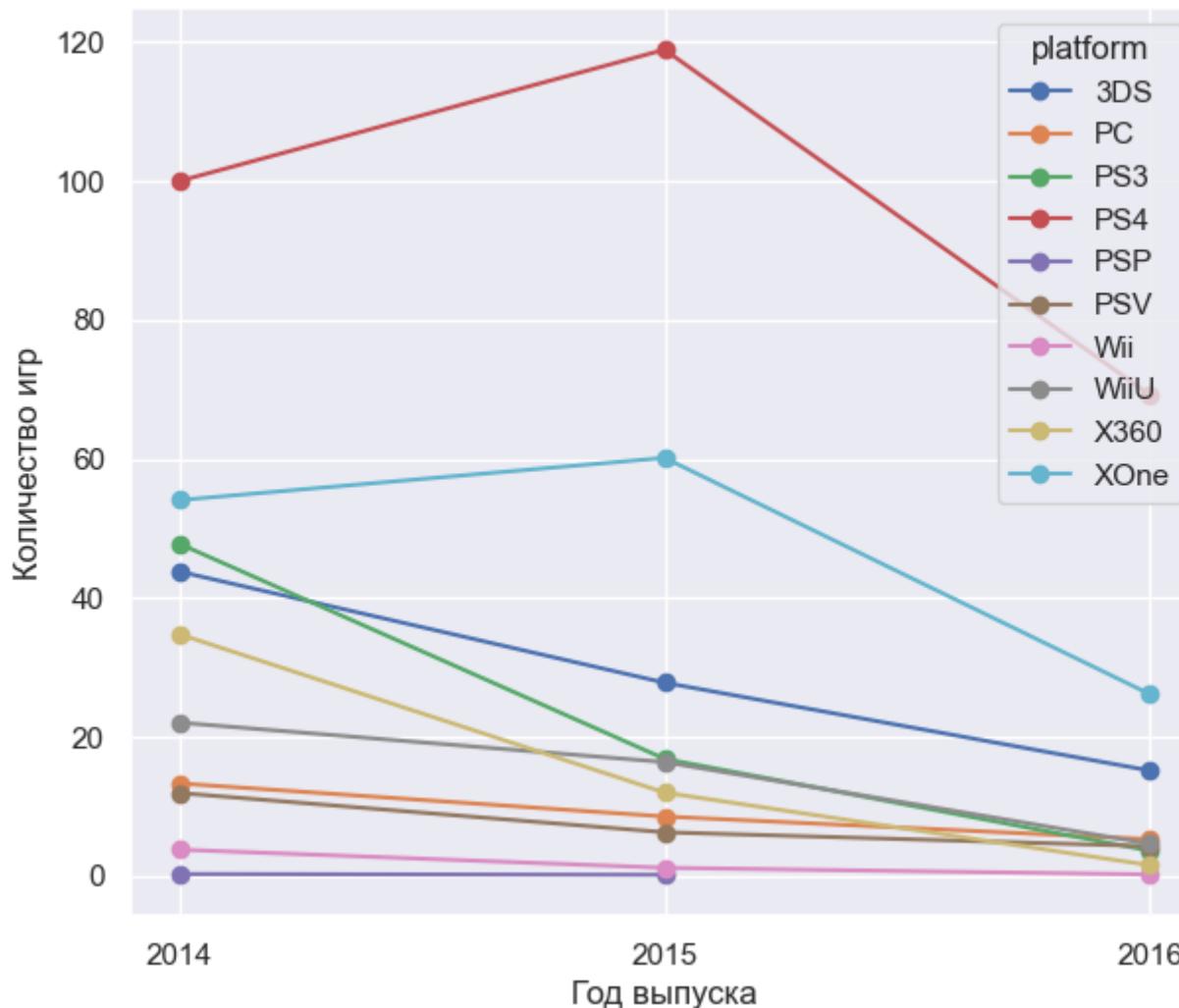
Далее работаем только с актуальными данными. Определим какие платформы лидируют по продажам, растут или падают.

```
In [58]: relevant_data.groupby('platform')['total_sales'].sum().sort_values(ascending=False)
```

```
Out[58]: platform
PS4      288.15
XOne    140.36
3DS     86.68
PS3      68.18
X360    48.22
WiiU    42.98
PC      27.05
PSV     22.40
Wii      5.07
PSP      0.36
Name: total_sales, dtype: float64
```

```
In [59]: relevant_data.pivot_table(index='year_of_release', columns='platform', values='total_sales', aggfunc='sum')\
.plot(style='o-', grid=True, figsize=(7,6))
plt.xticks([2014,2015,2016])
plt.title('Количество выпускаемых компьютерных игр в актуальный период с 1980 по 2016 гг.')
plt.xlabel('Год выпуска')
plt.ylabel('Количество игр');
```

Количество выпускаемых компьютерных игр в актуальный период с 1980 по 2016 гг.



Для сравнения платформ возьмем 2015 год к 2014 году. Отчетливо видно как растут платформы, недавно вышедшие на рынок (PS4 и Xbox One) и при этом падение платформ предыдущего поколения (PS3 и X360)

<div class="alert alert-warning", style="border:solid coral 3px; padding: 20px"> ⚠

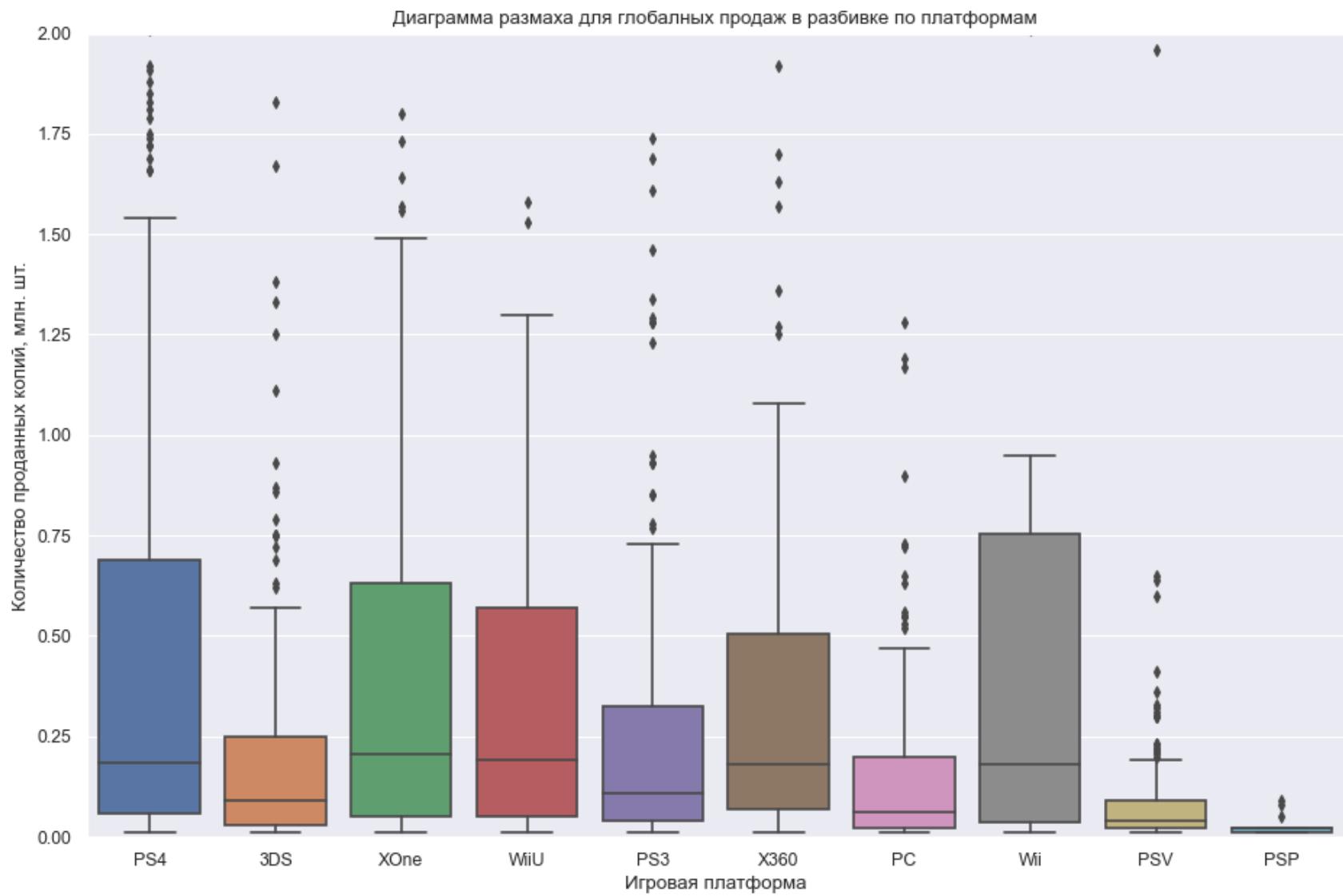
Комментарий ревьюера

Данные за 2016 год неполные, и делать вывод о падении продаж в 2016 году не стоит, можно сравнивать рост/падение на других промежутках, например 2015 год к 2014 ...

"Ящик с усами"

Далее, для актуальных данных, построим график "ящик с усами" для глобальных продаж в разбивке по платформам.

```
In [60]: plt.figure(figsize=(14,9), dpi= 80)
sns.boxplot(x='platform', y='total_sales', data=relevant_data, notch=False)
plt.ylim(0,2)
plt.title('Диаграмма размаха для глобальных продаж в разбивке по платформам')
plt.xlabel('Игровая платформа')
plt.ylabel('Количество проданных копий, млн. шт.');
```



<div class="alert alert-warning", style="border:solid coral 3px; padding: 20px"> ⚠

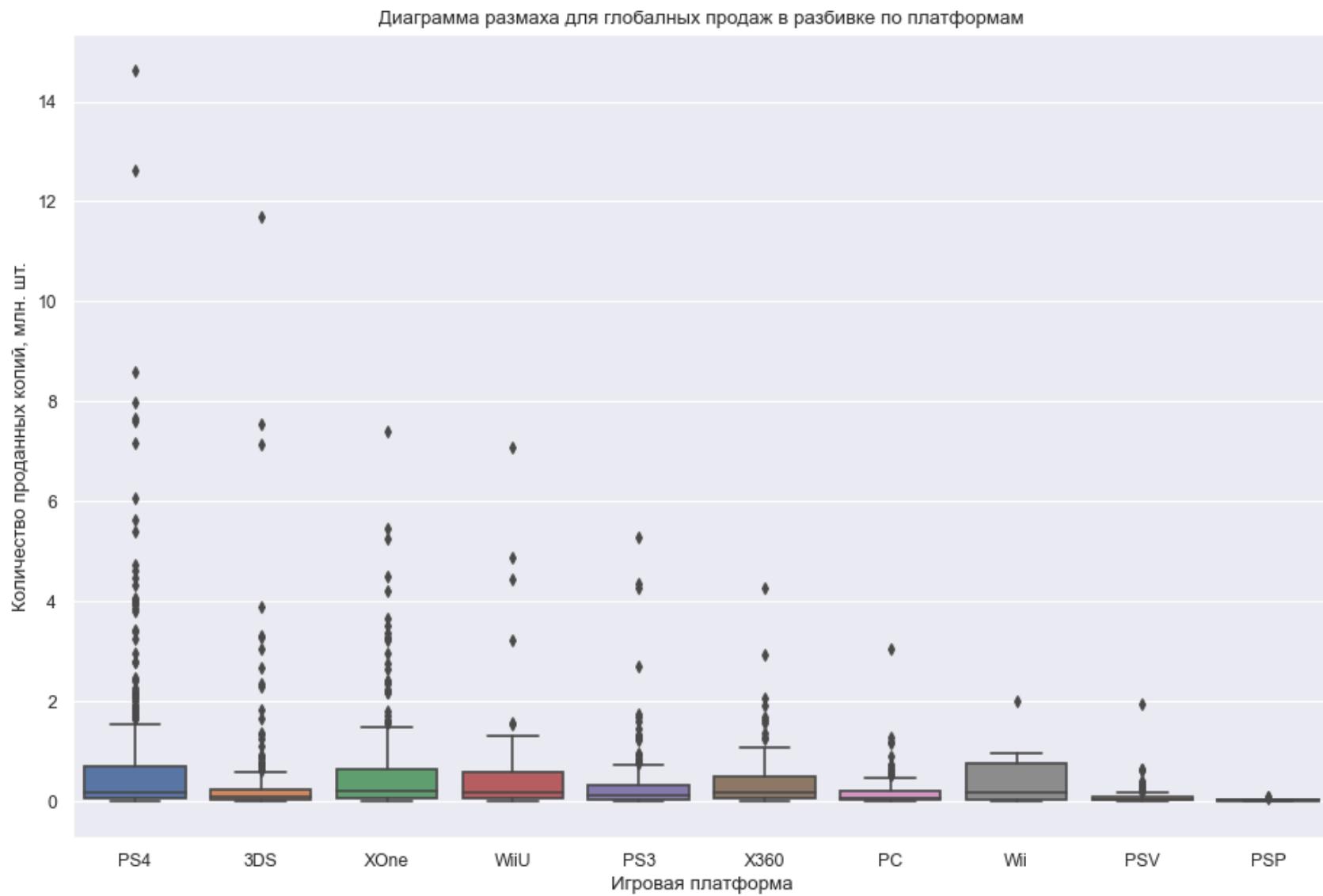
Комментарий ревьюера

Отлично, ты используешь диаграмму размаха для определения успешности платформы, молодец

Для более полной оценки продаж на платформах стоит добавить график со 100% масштабом, посмотреть на максимальные продажи

```
In [61]: plt.figure(figsize=(14,9), dpi= 80)
sns.boxplot(x='platform', y='total_sales', data=relevant_data, notch=False)

plt.title('Диаграмма размаха для глобальных продаж в разбивке по платформам')
plt.xlabel('Игровая платформа')
plt.ylabel('Количество проданных копий, млн. шт.');
```



 ✓ **Комментарий ревьюера в2**

Что показывают нам две диаграммы размаха — кол-во выбросов, это игры, которые принесли максимум выручки. Т.е. можно платформы/жанры сравнить по кол-ву игр-рекордсменов, а значит определить, какая из них способна выпустить наиболее привлекательные для игроманов игры. Это про выбросы

Второй вид мы используем для того, чтобы сравнить медианные продажи по платформе/жанру, чтобы уточнить в каком кол-ве продаются игры на платформе/жанре, какая из них более стабильна в продажах...

Две хорошие статьи про диаграмму размаха

[Визуализация категориальных данных](#)

[Исследуем отношение между переменными](#)

Выводы по графику

Из представленных выше данных можно отметить следующее:

- в выделенный период Wii как наиболее стабильная в продажах. Несмотря на то, что поддержка данной платформы была прекращена в 2013 году, 75% игр для этих платформ продавались тиражом до 750 тыс. копий, с выбросами до 2 млн.
- для платформ нового поколения (PS4, XOne, WiiU) характерно разброс в районе 600 - 700 тыс. копий, с медианным значением 200 тыс. При этом XOne имеет наибольшее медианное значение среди всех платформ этого периода
- Для всех платформ характерны успешные релизы, с выбросами, многократно превышающими их медианные значения. Так например для PS4 это значение стремится к 15 млн. копий, 3DS к 12 млн, а XOne к 8 млн.
- в целом также можно сказать, что к наиболее перспективным однозначно отметить PS4, XOne, WiiU. Также не стоит забывать о Wii, стабильность которой еще может конкурировать с более современными платформами. Также не стоит забывать, что спрос на игры для РС стабилен на протяжении десятка лет, хоть и не такой высокий как у конкурентов (в то же время это и не игровая консоль). И стоит отметить окончательный закат платформ PSP и PSV с незначительными продажами.

✖ Комментарий ревьюера

Стоит очистить перечень перспективных платформ от старожилов, которым уже по 10-11 лет ...

и дополнить перечень перспективных платформ

даже на падающих продажах можно заработать, когда объем составляет около 15-20 млн. копий, как например с 3DS

30-тилетняя история персональных компьютеров говорит, что игры для РС можно включить в рекомендацию

Анализ платформ PS4, XOne, 3DS

Для анализа как влияют отзывы критиков и пользователей на продажи, рассмотрим PS4, конкурента - XOne и наиболее популярную платформу в Японии - 3DS

```
In [62]: ps4 = relevant_data[relevant_data['platform'] == 'PS4']
```

```
In [63]: xone = relevant_data[relevant_data['platform'] == 'XOne']
```

```
In [64]: ds = relevant_data[relevant_data['platform'] == '3DS']
```

```
In [65]: ps4
```

Out[65]:

		name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating	total
31		Call of Duty: Black Ops 3	PS4	2015.0	Shooter	6.03	5.86	0.36	2.38	NaN	NaN	ND	
42		Grand Theft Auto V	PS4	2014.0	Action	3.96	6.31	0.38	1.97	97.0	8.3	M	
77		FIFA 16	PS4	2015.0	Sports	1.12	6.12	0.06	1.28	82.0	4.3	E	
87		Star Wars Battlefront (2015)	PS4	2015.0	Shooter	2.99	3.49	0.22	1.28	NaN	NaN	ND	
92		Call of Duty: Advanced Warfare	PS4	2014.0	Shooter	2.81	3.48	0.14	1.23	83.0	5.7	M	
...
16500		Root Letter	PS4	2016.0	Adventure	0.00	0.00	0.01	0.00	69.0	7.5	ND	
16503		Shin Hayarigami 2	PS4	2016.0	Adventure	0.00	0.00	0.01	0.00	NaN	NaN	ND	
16526		Dungeons 2	PS4	2016.0	Role-Playing	0.01	0.00	0.00	0.00	61.0	7.9	T	
16530		Carmageddon: Max Damage	PS4	2016.0	Action	0.01	0.00	0.00	0.00	51.0	5.5	M	
16585		Farming 2017 - The Simulation	PS4	2016.0	Simulation	0.00	0.01	0.00	0.00	NaN	NaN	ND	

376 rows × 12 columns

In [66]: xone

Out[66]:

		name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating	total_sales
99		Call of Duty: Black Ops 3	XOne	2015.0	Shooter	4.59	2.11	0.01	0.68	NaN	NaN	ND	7.3€
165		Grand Theft Auto V	XOne	2014.0	Action	2.81	2.19	0.00	0.47	97.0	7.9	M	5.4€
179		Call of Duty: Advanced Warfare	XOne	2014.0	Shooter	3.22	1.55	0.01	0.48	81.0	5.4	M	5.2€
242		Halo 5: Guardians	XOne	2015.0	Shooter	2.78	1.27	0.03	0.41	84.0	6.4	T	4.4€
270		Fallout 4	XOne	2015.0	Role-Playing	2.51	1.32	0.01	0.38	88.0	6.2	M	4.22
...
16630		Sébastien Loeb Rally Evo	XOne	2016.0	Racing	0.00	0.01	0.00	0.00	63.0	8.2	E	0.01
16643		Rugby Challenge 3	XOne	2016.0	Sports	0.00	0.01	0.00	0.00	NaN	6.6	E	0.01
16645		ZombiU	XOne	2016.0	Action	0.00	0.01	0.00	0.00	NaN	NaN	ND	0.01
16660		Prison Architect	XOne	2016.0	Action	0.01	0.00	0.00	0.00	74.0	6.7	ND	0.01
16672		Metal Gear Solid V: The Definitive Experience	XOne	2016.0	Action	0.01	0.00	0.00	0.00	NaN	NaN	M	0.01

228 rows × 12 columns

In [67]: `ds`

Out[67]:

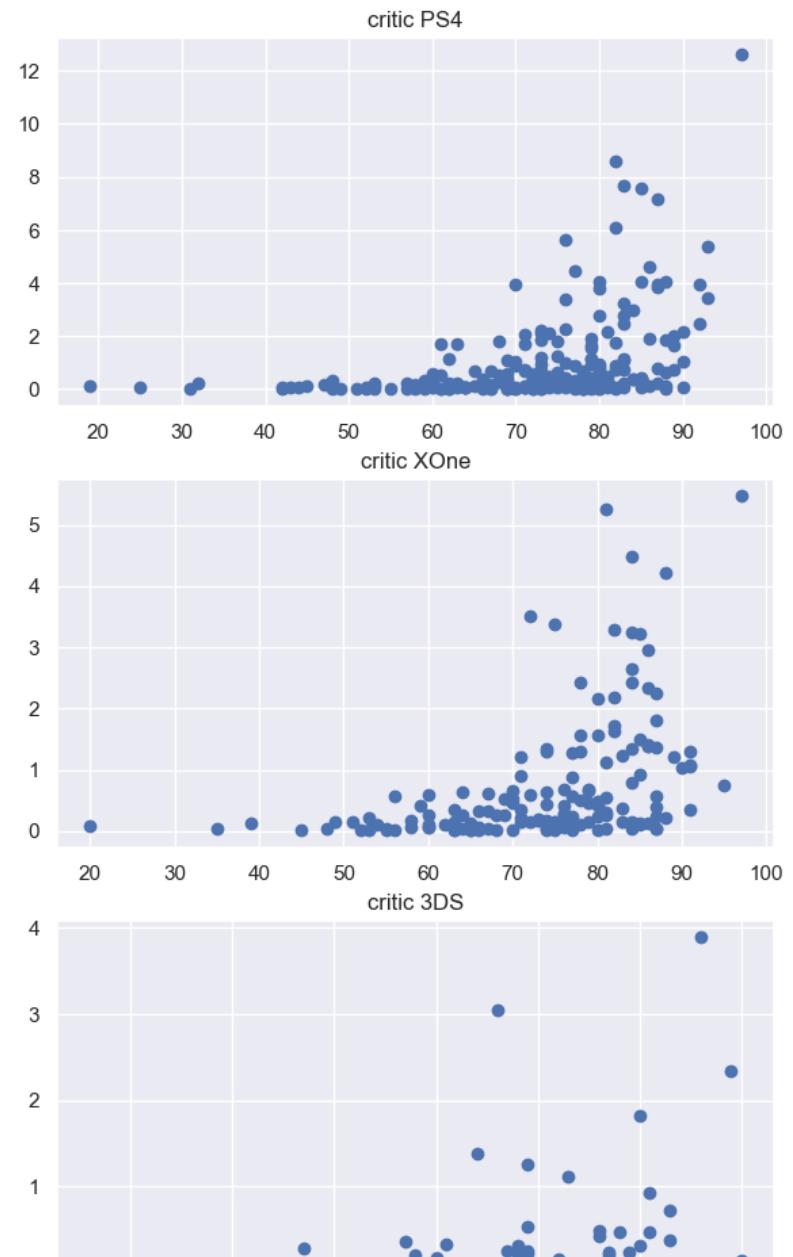
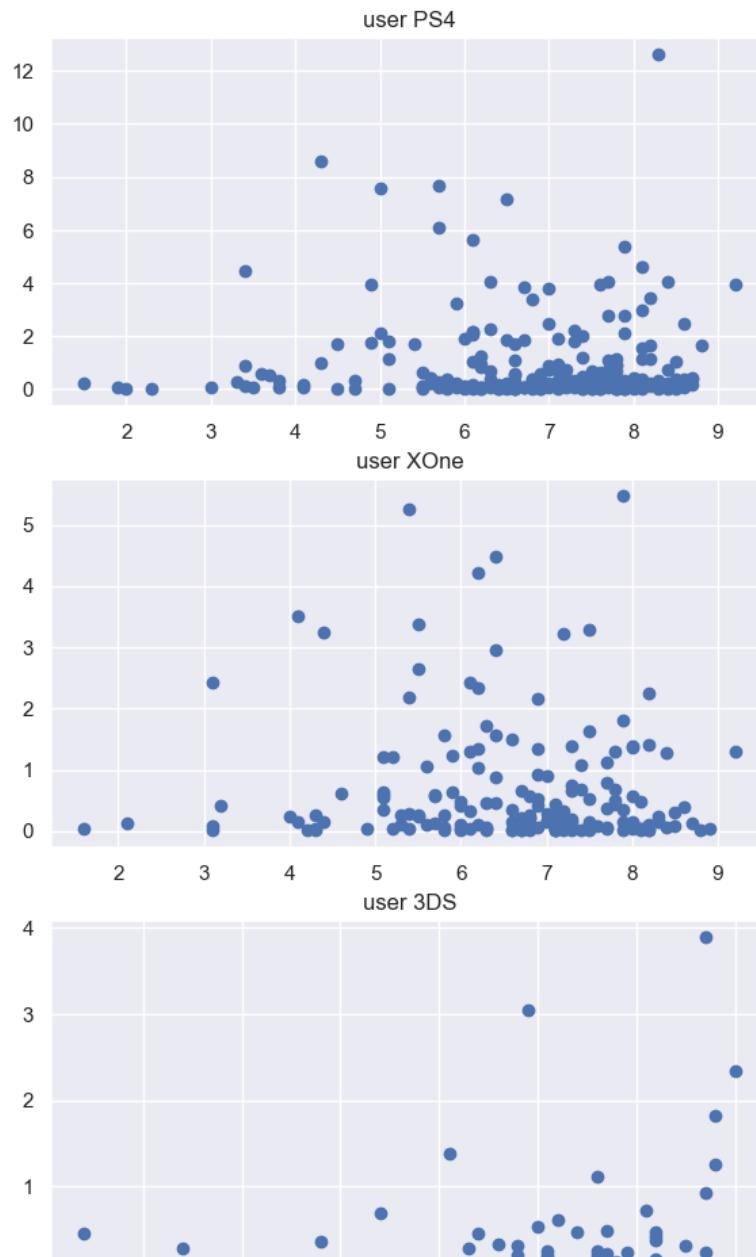
		name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating	tot:
	47	Pokemon Omega Ruby/Pokemon Alpha Sapphire	3DS	2014.0	Role-Playing	4.35	3.49	3.10	0.74	NaN	NaN	ND	
	96	Super Smash Bros. for Wii U and 3DS	3DS	2014.0	Fighting	3.27	1.37	2.43	0.48	NaN	NaN	ND	
	108	Pokemon Sun/Moon	3DS	2016.0	Role-Playing	2.98	1.45	2.26	0.45	NaN	NaN	ND	
	312	Monster Hunter 4 Ultimate	3DS	2014.0	Role-Playing	0.68	0.48	2.62	0.11	86.0	8.7	T	
	406	Monster Hunter X	3DS	2015.0	Action	0.27	0.21	2.79	0.05	NaN	NaN	ND	

	16599	Legends of Oz: Dorothy's Return	3DS	2014.0	Puzzle	0.00	0.01	0.00	0.00	NaN	NaN	E	
	16608	Mario & Luigi: Paper Jam & Mario Kart 7 Double...	3DS	2015.0	Misc	0.00	0.00	0.01	0.00	NaN	NaN	ND	
	16610	Kiniro no Corda 3	3DS	2015.0	Adventure	0.00	0.00	0.01	0.00	NaN	NaN	ND	
	16668	Fujiko F. Fujio Characters: Great Assembly! Sl...	3DS	2014.0	Action	0.00	0.00	0.01	0.00	NaN	NaN	ND	
	16677	Aikatsu Stars! My Special Appeal	3DS	2016.0	Action	0.00	0.00	0.01	0.00	NaN	NaN	ND	

212 rows × 12 columns

```
In [68]: fig, axs = plt.subplots(nrows=3 , ncols=2, figsize=(15,12))
fig.suptitle('Диаграммы рассеяния')
axs[0, 0].scatter(x='user_score', y='total_sales', data=ps4)
axs[0, 0].set(title='user PS4')
axs[0, 1].scatter(x='critic_score', y='total_sales', data=ps4)
axs[0, 1].set(title='critic PS4')
axs[1, 0].scatter(x='user_score', y='total_sales', data=xone)
axs[1, 0].set(title='user XOne')
axs[1, 1].scatter(x='critic_score', y='total_sales', data=xone)
axs[1, 1].set(title='critic XOne')
axs[2, 0].scatter(x='user_score', y='total_sales', data=ds)
axs[2, 0].set(title='user 3DS')
axs[2, 1].scatter(x='critic_score', y='total_sales', data=ds)
axs[2, 1].set(title='critic 3DS');
```

Диаграммы рассеяния





По диаграммам рассеяния:

- для начала посмотрим на распределения отзывов пользователей и количество проданных копий по выбранным платформам. Основные оценки для всех платформ сосредоточены в интервале 6-9 баллов. Для платформ PS4 и XOne можно отметить, что даже игры с низким пользовательским рейтингом (ниже 6) продавались с успехом, а вот у 3DS подобного не наблюдается - все крупные продажи (свыше 1 млн. копий) начинаются для игр выше 6.
- отзывы критиков для всех трех платформ в основном находятся в интервале 60-90 баллов. Для XOne отчетливо видно, что самые крепкие продажи соответствуют баллам в районе 80.

```
In [69]: ps4['total_sales'].corr(ps4['user_score'])
```

```
Out[69]: -0.040131589472697356
```

```
In [70]: xone['total_sales'].corr(xone['user_score'])
```

```
Out[70]: -0.0703839280647581
```

```
In [71]: ds['total_sales'].corr(ds['user_score'])
```

```
Out[71]: 0.2151932718527028
```

```
In [72]: ps4['total_sales'].corr(ps4['critic_score'])
```

```
Out[72]: 0.40266141068104083
```

```
In [73]: xone['total_sales'].corr(xone['critic_score'])
```

```
Out[73]: 0.42867694370333226
```

```
In [74]: ds['total_sales'].corr(ds['critic_score'])
```

```
Out[74]: 0.31411749286905105
```

Полученные числовые коэффициенты подтверждают графические данные. Для платформ PS4 и XOne по сути нет корреляционной связи между отзывами пользователей и продажами. В целом, если смотреть на числовые коэффициенты и диаграммы рассеяния, можно сказать, что причинно-следственных связей между отзывами пользователей и количеством проданных копий - нет. Ведь вполне возможно, что пользователи ожидали выход каких-то игр и, разочаровавшись уже после покупки, могли выставить низкие рейтинги. Для платформы 3DS корреляция слабовыраженная. По поводу отзывов критиков - для платформ она варьируется от 0,31 до 0,42, то есть корреляция "средневыраженная" (если можно так сказать).

<div class="alert alert-warning", style="border:solid coral 3px; padding: 20px"> ⚠

Комментарий ревьюера

Возможно пользователи более критичны к играм, чем критики, но мы не сможем оценить какие действия повлияли на рост продаж в рамках нашего проекта (ограниченность имеющихся данных)

Достаточно много игр с высокой оценкой критиков и слабой выручкой

Приведу пример ложной корреляции, весьма известный в статистической литературе. Была исследована корреляционная связь между числом аистов, свивших гнезда в южных районах Швеции, и рождаемостью в эти же годы в Швеции. Расчёты, выполненные ради шутки, показали существенную положительную корреляцию между этими явлениями, хотя любому понятно, что это ложная корреляция.

Ещё пример ложной корреляции между приемом на работу новых менеджеров и созданием новых производственных мощностей. Возможно, именно менеджеры являются «причиной» капиталовложений в новые производственные мощности? Или же, наоборот, создание новых производственных мощностей послужило «причиной» приема на работу новых менеджеров?

Например, можно обнаружить сильную положительную связь (корреляцию) между разрушениями, вызванными пожаром, и числом пожарных, тушивших пожар. Следует ли заключить, что пожарные вызывают разрушения? Конечно, наиболее вероятное объяснение этой корреляции состоит в том, что размер пожара (внешняя переменная, которую забыли включить в исследование) оказывает влияние, как на масштаб разрушений, так и на числе привлеченных пожарных (т. е. чем больше пожар, тем большее количество пожарных вызывается на его тушение) .

✖ Комментарий ревьюера

Стоит выполнить исследование зависимости продаж от оценок пользователей и критиков по платформам конкурентам — рассмотреть в этом разделе отдельно две-три платформы

ps3, xone

Выводы, сделанные на основе расчетов по нескольким платформам выглядят "весомей" и убедительней

Распределение игр по жанрам

Для определения в каких жанрах самое большое количество игр и какие жанры самые прибыльные, сделаем сводные таблицы и построим круговые диаграммы к ним.

```
In [75]: genre_sum = relevant_data.groupby('genre')['total_sales'].sum().sort_values(ascending=False)
genre_count = relevant_data.groupby('genre')['name'].count().sort_values(ascending=False)
```

```
In [76]: genre_sum
```

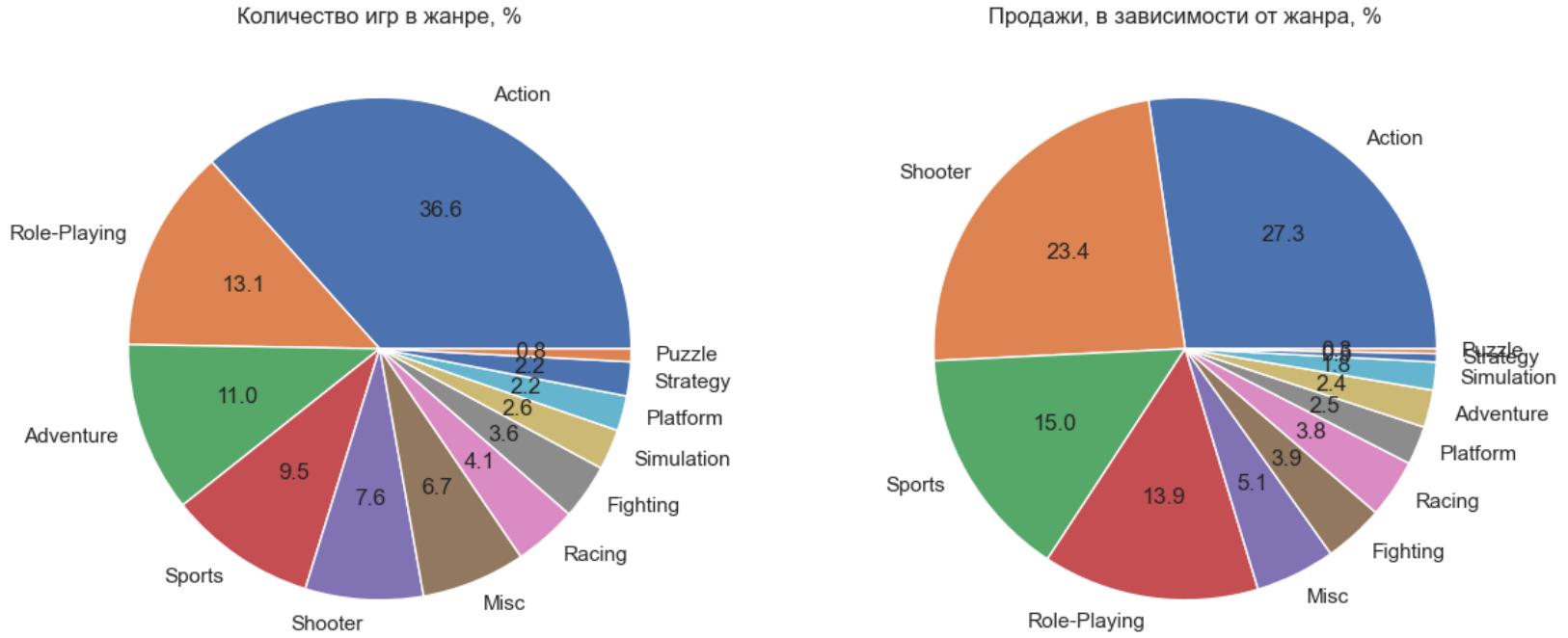
```
Out[76]: genre
Action      199.36
Shooter     170.94
Sports       109.48
Role-Playing 101.44
Misc         37.55
Fighting     28.22
Racing        27.52
Platform      18.09
Adventure     17.55
Simulation    13.13
Strategy      3.96
Puzzle        2.21
Name: total_sales, dtype: float64
```

```
In [77]: genre_count
```

```
Out[77]: genre
Action           619
Role-Playing     221
Adventure        185
Sports            161
Shooter           128
Misc              113
Racing             69
Fighting           60
Simulation         44
Platform            38
Strategy            37
Puzzle              14
Name: name, dtype: int64
```

```
In [78]: fig, axs = plt.subplots(nrows=1 , ncols=2, figsize=(14,6))
fig.suptitle('Разделение по количеству игр в определенном жанре и количество проданных копий, в зависимости от жанра')
genre_count.plot(kind='pie', autopct='%.1f', ylabel="", ax=axs[0])
axs[0].set(title='Количество игр в жанре, %')
genre_sum.plot(kind='pie', autopct='%.1f', ylabel="", ax=axs[1])
axs[1].set(title='Продажи, в зависимости от жанра, %');
```

Разделение по количеству игр в определенном жанре и количество проданных копий, в зависимости от жанра



Из полученных данных можно увидеть, в каких жанрах выпускается самое большое количество игр - Action, Role-Playing, Adventure. А вот что касается продаж, в зависимости от жанра, выяснилось, что общее количество игр в определенном жанре не всегда соответствует количеству проданных копий. В продажах лидирует также жанр Action, а вот на втором и третьем местах - Shooter и Sports. Жанр Adventure по продажам и вовсе 4й с конца. Аутсайдеры и по количеству игр в жанрах и по количеству проданных копий одинаковые в обоих выборках - Puzzle, Strategy, Simulation.

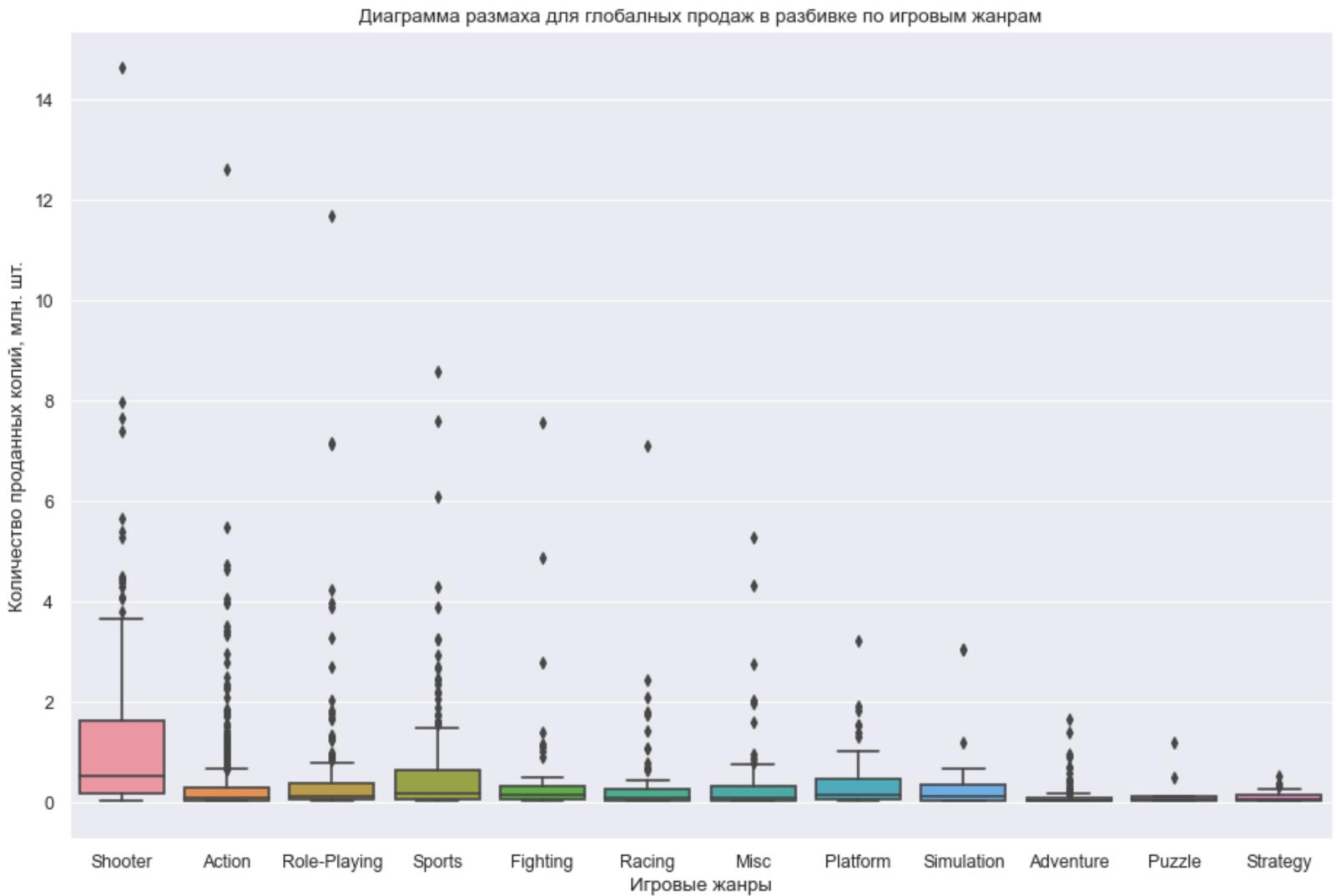
✖ Комментарий ревьюера

Стоит проанализировать прибыльность жанров на диаграмме размаха, сравнить медианные продажи на каждом жанре и проверить какая из них более стабильна и имеет более длинный ряд успешно продающихся игр

График нарисовать **в двух масштабах с выбросами и без** (чтобы было видно 0.75-квантиль)

```
In [79]: plt.figure(figsize=(14,9), dpi= 80)
sns.boxplot(x='genre', y='total_sales', data=relevant_data, notch=False)

plt.title('Диаграмма размаха для глобальных продаж в разбивке по игровым жанрам')
plt.xlabel('Игровые жанры')
plt.ylabel('Количество проданных копий, млн. шт.');
```



```
In [80]: plt.figure(figsize=(14,9), dpi= 80)
sns.boxplot(x='genre', y='total_sales', data=relevant_data, notch=False)
plt.ylim(0,4)
plt.title('Диаграмма размаха для глобальных продаж в разбивке по игровым жанрам')
plt.xlabel('Игровые жанры')
plt.ylabel('Количество проданных копий, млн. шт.');
```

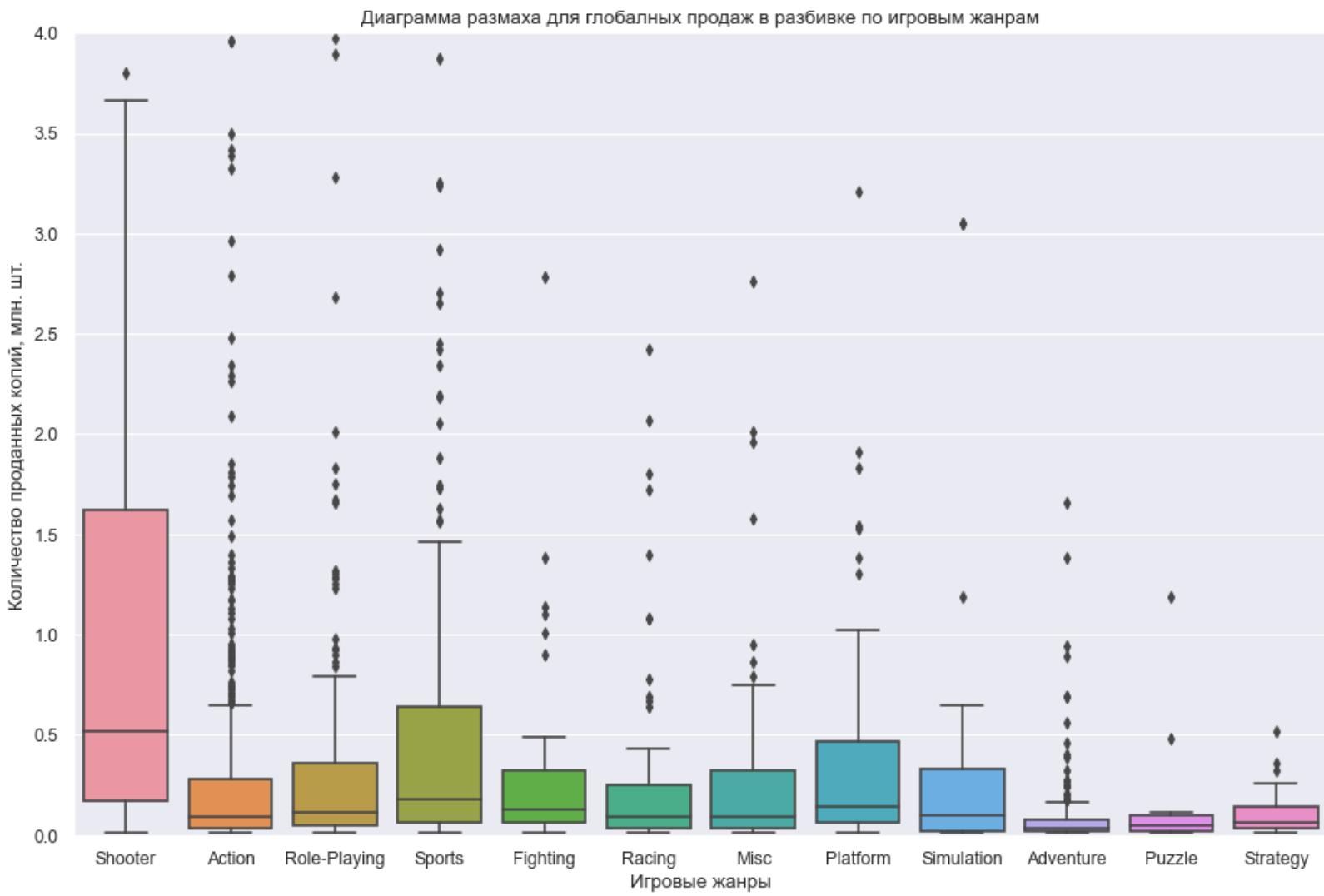


Диаграмма размаха дает немного неожиданные результаты - для лидера, жанра Action в 75% случаев число копий составляет до 300 тысяч. И хотя 3/4 игр этого жанра и продаются относительно небольшим тиражом (по сравнению с другими жанрами), успех отдельных игр (видно на выбросах) и общее количество игр в жанре (видно на круговой диаграмме - 36,6% от общего числа), позволяют лидировать по количеству проданных копий в целом среди всех платформ. Жанр шутеров, где 3й квантиль находится в районе 1.6 млн копий, а медианное значение в районе 500 тысяч - лидер данного рейтинга.

Портрет пользователя каждого региона

```
In [81]: relevant_data
```

Out[81]:

		name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating
	31	Call of Duty: Black Ops 3	PS4	2015.0	Shooter	6.03	5.86	0.36	2.38	NaN	NaN	ND
	42	Grand Theft Auto V	PS4	2014.0	Action	3.96	6.31	0.38	1.97	97.0	8.3	M
	47	Pokemon Omega Ruby/Pokemon Alpha Sapphire	3DS	2014.0	Role-Playing	4.35	3.49	3.10	0.74	NaN	NaN	ND
	77	FIFA 16	PS4	2015.0	Sports	1.12	6.12	0.06	1.28	82.0	4.3	E
	87	Star Wars Battlefront (2015)	PS4	2015.0	Shooter	2.99	3.49	0.22	1.28	NaN	NaN	ND

	16703	Strawberry Nauts	PSV	2016.0	Adventure	0.00	0.00	0.01	0.00	NaN	NaN	ND
	16707	Aiyoku no Eustia	PSV	2014.0	Misc	0.00	0.00	0.01	0.00	NaN	NaN	ND
	16710	Samurai Warriors: Sanada Maru	PS3	2016.0	Action	0.00	0.00	0.01	0.00	NaN	NaN	ND
	16712	Haitaka no Psychedelica	PSV	2016.0	Adventure	0.00	0.00	0.01	0.00	NaN	NaN	ND
	16714	Winning Post 8 2016	PSV	2016.0	Simulation	0.00	0.00	0.01	0.00	NaN	NaN	ND

1689 rows × 12 columns

Самые популярные платформы (топ-5) по регионам

Для начала сгруппируем данные по платформам в зависимости от региона (NA, EU, JP) и построим к ним барплот.

```
In [82]: relevant_data.groupby('platform')['na_sales'].sum().sort_values(ascending=False).head()
```

```
Out[82]: platform
PS4      98.61
XOne    81.27
X360    28.30
3DS     22.64
PS3     22.05
Name: na_sales, dtype: float64
```

```
In [83]: # relevant_data.groupby('platform')['na_sales'].sum().sort_values(ascending=False).head().plot(kind='bar', figsize=(10, 6))
```

```
In [84]: relevant_data.groupby('platform')['eu_sales'].sum().sort_values(ascending=False).head()
```

```
Out[84]: platform
PS4      130.04
XOne    46.25
PS3     25.54
PC      17.97
3DS     16.12
Name: eu_sales, dtype: float64
```

```
In [85]: # relevant_data.groupby('platform')['eu_sales'].sum().sort_values(ascending=False).head().plot(kind='bar', figsize=(10, 6))
```

```
In [86]: relevant_data.groupby('platform')['jp_sales'].sum().sort_values(ascending=False).head()
```

```
Out[86]: platform
3DS      44.24
PS4      15.02
PSV      14.54
PS3      11.22
WiiU     7.31
Name: jp_sales, dtype: float64
```

```
In [87]: # relevant_data.groupby('platform')['jp_sales'].sum().sort_values(ascending=False).head().plot(kind='bar', figsize=(10, 6))
```

In [88]:

```
fig, axs = plt.subplots(nrows=1, ncols=3, figsize=(14,5))
fig.suptitle('ТОП-5 популярных платформ по регионам - Северная Америка, Евросоюз, Япония')
relevant_data.groupby('platform')['na_sales'].sum().sort_values(ascending=False).head().plot(kind='bar', xlabel=
axs[0].set(title='Северная Америка'))
relevant_data.groupby('platform')['eu_sales'].sum().sort_values(ascending=False).head().plot(kind='bar', xlabel=
axs[1].set(title='Евросоюз'))
relevant_data.groupby('platform')['jp_sales'].sum().sort_values(ascending=False).head().plot(kind='bar', xlabel=
axs[2].set(title='Япония'));
```



Из полученных данных можно увидеть, что для Северной Америки и Европейского Союза пятерка платформ-лидеров одинаковая, но отличается их порядок. Так для Северной Америки характерно плавное снижение интересов к разным платформам - значения PS4 превышает значения XOne лишь в районе 10%. Примерно такое же различие между вторым и третьим местами (XOne, X360). Для Европейского Союза же характерно тотальное лидерство PS4 - в два раза превышает значение для занявшей второе место PS3. В Японии же тотальным лидером является платформа 3DS, которая в СА и ЕС лишь замкнула ТОП-5. Возможными причинами может являться то, что в Северной Америке Nintendo и Microsoft являются давними конкурентами, и хотя PS4 лидирует, Xbox One не сильно отстает, и лидеры могут поменяться местами. В Японии Nintendo является безоговорочным лидером. Что касается ЕС - в регионе нет собственных производителей игровых платформ и, возможно, компания Sony провела лучше рекламную кампанию, или изначально ориентировалась на европейский рынок, поэтому захватила там абсолютное лидерство.

```
In [89]: def top_platform(row):
    platform = row['platform']
    top = ['PS4', 'PS3', 'XOne', 'X360', 'PC', '3DS', 'PSV', 'WiiU']
    for i in top:
        if i == platform:
            return i
    return 'другие'
```

```
In [90]: data['top_platform'] = data.apply(top_platform, axis=1)
```

Написанная выше функция `top_platform` относит платформы, не входящие в ТОП-5 в категорию `другие`. Аргумент функции `row` - строка датафрейма целиком, переменной `platform` обращаемся к конкретному столбцу. Далее циклом перебираем список популярных платформ, и платформам, не входящим в этот список, присваиваем значение `другой`. Далее создаем новый столбец `top_platform`, куда будут записываться результаты и применяем метод `apply` к датафрейму. Параметром `axis=1` указываем, что на вход нужно отправить строки датафрейма.

```
In [91]: data.head(50)
```

Out[91]:

		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	8.0	E
1		Super Mario Bros.	NES	1985.0	Platform	29.08	3.58	6.81	0.77	NaN	NaN	ND
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	8.0	E
4		Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	11.27	8.89	10.22	1.00	NaN	NaN	ND
5		Tetris	GB	1989.0	Puzzle	23.20	2.26	4.22	0.58	NaN	NaN	ND
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	ND
10		Nintendogs	DS	2005.0	Simulation	9.05	10.95	1.93	2.74	NaN	NaN	ND
11		Mario Kart DS	DS	2005.0	Racing	9.71	7.47	4.13	1.90	91.0	8.6	E
12		Pokemon Gold/Pokemon Silver	GB	1999.0	Role-Playing	9.00	6.18	7.20	0.71	NaN	NaN	ND
13		Wii Fit	Wii	2007.0	Sports	8.92	8.03	3.60	2.15	80.0	7.7	E
14		Kinect Adventures!	X360	2010.0	Misc	15.00	4.89	0.24	1.69	61.0	6.3	E
15		Wii Fit Plus	Wii	2009.0	Sports	9.01	8.49	2.53	1.77	80.0	7.4	E
16		Grand Theft Auto V	PS3	2013.0	Action	7.02	9.09	0.98	3.96	97.0	8.2	M
17		Grand Theft Auto: San Andreas	PS2	2004.0	Action	9.43	0.40	0.41	10.57	95.0	9.0	M
18		Super Mario World	SNES	1990.0	Platform	12.78	3.75	3.54	0.55	NaN	NaN	ND
19		Brain Age: Train Your Brain in	DS	2005.0	Misc	4.74	9.20	4.16	2.04	77.0	7.9	E

		name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating
Minutes a Day												
		Pokemon Diamond/Pokemon Pearl	DS	2006.0	Role-Playing	6.38	4.46	6.04	1.36	NaN	NaN	ND
20		Super Mario Land	GB	1989.0	Platform	10.83	2.71	4.18	0.42	NaN	NaN	ND
21		Super Mario Bros. 3	NES	1988.0	Platform	9.54	3.44	3.84	0.46	NaN	NaN	ND
22		Grand Theft Auto V	X360	2013.0	Action	9.66	5.14	0.06	1.41	97.0	8.1	M
23		Grand Theft Auto: Vice City	PS2	2002.0	Action	8.41	5.49	0.47	1.78	95.0	8.7	M
		Pokemon Ruby/Pokemon Sapphire	GBA	2002.0	Role-Playing	6.06	3.90	5.38	0.50	NaN	NaN	ND
25		Brain Age 2: More Training in Minutes a Day	DS	2005.0	Puzzle	3.43	5.35	5.32	1.18	77.0	7.1	E
		Pokemon Black/Pokemon White	DS	2010.0	Role-Playing	5.51	3.17	5.65	0.80	NaN	NaN	ND
27		Gran Turismo 3: A-Spec	PS2	2001.0	Racing	6.85	5.09	1.87	1.16	95.0	8.4	E
28		Call of Duty: Modern Warfare 3	X360	2011.0	Shooter	9.04	4.24	0.13	1.32	88.0	3.4	M
		Pokémon Yellow: Special Pikachu Edition	GB	1998.0	Role-Playing	5.89	5.04	3.12	0.59	NaN	NaN	ND
30		Call of Duty: Black Ops 3	PS4	2015.0	Shooter	6.03	5.86	0.36	2.38	NaN	NaN	ND
31		Call of Duty: Black Ops	X360	2010.0	Shooter	9.70	3.68	0.11	1.13	87.0	6.3	M
32		Pokemon X/Pokemon Y	3DS	2013.0	Role-Playing	5.28	4.19	4.35	0.78	NaN	NaN	ND

		name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating
34		Call of Duty: Black Ops II	PS3	2012.0	Shooter	4.99	5.73	0.65	2.42	83.0	5.3	M
35		Call of Duty: Black Ops II	X360	2012.0	Shooter	8.25	4.24	0.07	1.12	83.0	4.8	M
36		Call of Duty: Modern Warfare 2	X360	2009.0	Shooter	8.52	3.59	0.08	1.28	94.0	6.3	M
37		Call of Duty: Modern Warfare 3	PS3	2011.0	Shooter	5.54	5.73	0.49	1.57	88.0	3.2	M
38		Grand Theft Auto III	PS2	2001.0	Action	6.99	4.51	0.30	1.30	97.0	8.5	M
39		Super Smash Bros. Brawl	Wii	2008.0	Fighting	6.62	2.55	2.66	1.01	93.0	8.9	T
40		Mario Kart 7	3DS	2011.0	Racing	5.03	4.02	2.69	0.91	85.0	8.2	E
41		Call of Duty: Black Ops	PS3	2010.0	Shooter	5.99	4.37	0.48	1.79	88.0	6.4	M
42		Grand Theft Auto V	PS4	2014.0	Action	3.96	6.31	0.38	1.97	97.0	8.3	M
43		Animal Crossing: Wild World	DS	2005.0	Simulation	2.50	3.45	5.33	0.86	86.0	8.7	E
44		Halo 3	X360	2007.0	Shooter	7.97	2.81	0.13	1.21	94.0	7.8	M
45		Super Mario 64	N64	1996.0	Platform	6.91	2.85	1.91	0.23	NaN	NaN	ND
46		Pokemon HeartGold/Pokemon SoulSilver	DS	2009.0	Action	4.34	2.71	3.96	0.76	NaN	NaN	ND
47		Pokemon Omega Ruby/Pokemon Alpha Sapphire	3DS	2014.0	Role-Playing	4.35	3.49	3.10	0.74	NaN	NaN	ND
48		Gran Turismo 4	PS2	2004.0	Racing	3.01	0.01	1.10	7.53	89.0	8.5	E
49		Super Mario Galaxy	Wii	2007.0	Platform	6.06	3.35	1.20	0.74	97.0	8.9	E

Самые популярные жанры (топ-5) по регионам

Выполним аналогичные манипуляции для игровых жанров, в зависимости от региона.

```
In [92]: relevant_data.groupby('genre')[ 'na_sales' ].sum().sort_values(ascending=False).head()
```

```
Out[92]: genre
Shooter      79.02
Action       72.53
Sports        46.13
Role-Playing  33.47
Misc          15.05
Name: na_sales, dtype: float64
```

```
In [93]: relevant_data.groupby('genre')[ 'eu_sales' ].sum().sort_values(ascending=False).head()
```

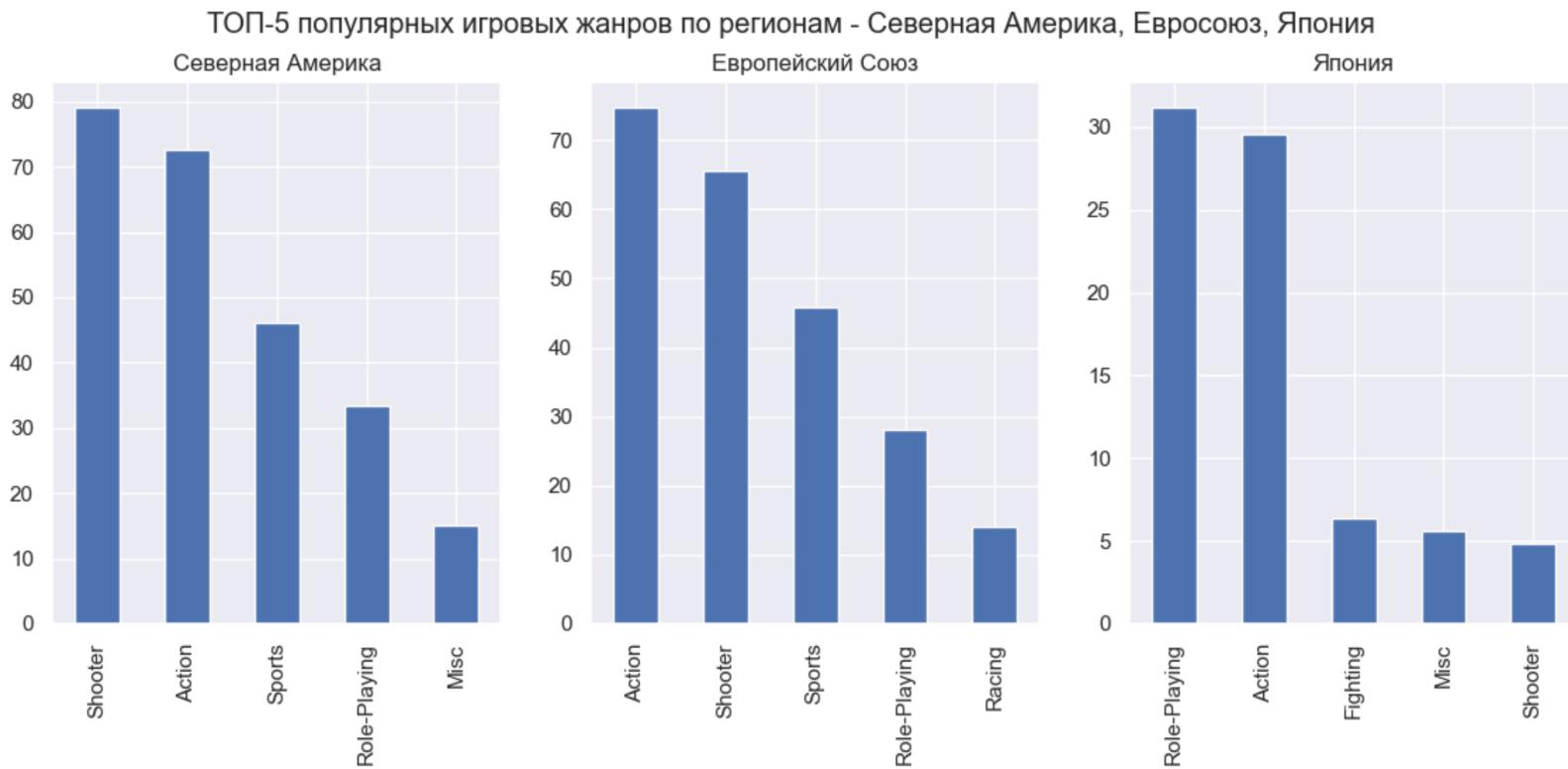
```
Out[93]: genre
Action       74.68
Shooter      65.52
Sports        45.73
Role-Playing  28.17
Racing        14.13
Name: eu_sales, dtype: float64
```

```
In [94]: relevant_data.groupby('genre')[ 'jp_sales' ].sum().sort_values(ascending=False).head()
```

```
Out[94]: genre
Role-Playing  31.16
Action        29.58
Fighting      6.37
Misc          5.61
Shooter       4.87
Name: jp_sales, dtype: float64
```

In [95]:

```
fig, axs = plt.subplots(nrows=1, ncols=3, figsize=(14,5))
fig.suptitle('ТОП-5 популярных игровых жанров по регионам - Северная Америка, Евросоюз, Япония')
relevant_data.groupby('genre')[['na_sales']].sum().sort_values(ascending=False).head().plot(kind='bar', xlabel="", 
axs[0].set(title='Северная Америка'))
relevant_data.groupby('genre')[['eu_sales']].sum().sort_values(ascending=False).head().plot(kind='bar', xlabel="", 
axs[1].set(title='Европейский Союз'))
relevant_data.groupby('genre')[['jp_sales']].sum().sort_values(ascending=False).head().plot(kind='bar', xlabel="", 
axs[2].set(title='Япония'));
```



И вновь результаты в СА и ЕС оказались похожи, даже очень - в лидерах жанр Action, даже значения по количеству проданных копий примерно равны. Отличаются жанры, замыкающие ТОП-5 - для СА это Misc, а для ЕС это Racing. Для Японии же неоспоримым лидером стали игры в жанре Role-Playing, но и Action в Японии также любят. Причины подобных различий можно все таки к культурным, национальным, ментальным отличиям между жителями отдельных стран и континентов. То есть если между жителями СА и ЕС есть много общего, Японии же далека от этих регионов. Не случайно же, крупные компании, прежде чем выйти на новый для них рынок, тщательно изучают страны, их нравы, традиции и особенности.

```
In [96]: def top_genre(row):
    genre = row['genre']
    top = ['Action', 'Shooter', 'Sports', 'Role-Playing', 'Racing', 'Fighting', 'Misc']
    for i in top:
        if i == genre:
            return i
    return 'другие'
```

```
In [97]: data['top_genre'] = data.apply(top_genre, axis=1)
```

```
In [98]: data.head(50)
```

Out[98]:

		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	8.0	E
1		Super Mario Bros.	NES	1985.0	Platform	29.08	3.58	6.81	0.77	NaN	NaN	ND
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	8.0	E
4		Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	11.27	8.89	10.22	1.00	NaN	NaN	ND
5		Tetris	GB	1989.0	Puzzle	23.20	2.26	4.22	0.58	NaN	NaN	ND
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	ND
10		Nintendogs	DS	2005.0	Simulation	9.05	10.95	1.93	2.74	NaN	NaN	ND
11		Mario Kart DS	DS	2005.0	Racing	9.71	7.47	4.13	1.90	91.0	8.6	E
12		Pokemon Gold/Pokemon Silver	GB	1999.0	Role-Playing	9.00	6.18	7.20	0.71	NaN	NaN	ND
13		Wii Fit	Wii	2007.0	Sports	8.92	8.03	3.60	2.15	80.0	7.7	E
14		Kinect Adventures!	X360	2010.0	Misc	15.00	4.89	0.24	1.69	61.0	6.3	E
15		Wii Fit Plus	Wii	2009.0	Sports	9.01	8.49	2.53	1.77	80.0	7.4	E
16		Grand Theft Auto V	PS3	2013.0	Action	7.02	9.09	0.98	3.96	97.0	8.2	M
17		Grand Theft Auto: San Andreas	PS2	2004.0	Action	9.43	0.40	0.41	10.57	95.0	9.0	M
18		Super Mario World	SNES	1990.0	Platform	12.78	3.75	3.54	0.55	NaN	NaN	ND
19		Brain Age: Train Your Brain in	DS	2005.0	Misc	4.74	9.20	4.16	2.04	77.0	7.9	E

		name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating
Minutes a Day												
		Pokemon Diamond/Pokemon Pearl	DS	2006.0	Role-Playing	6.38	4.46	6.04	1.36	NaN	NaN	ND
20		Super Mario Land	GB	1989.0	Platform	10.83	2.71	4.18	0.42	NaN	NaN	ND
21		Super Mario Bros. 3	NES	1988.0	Platform	9.54	3.44	3.84	0.46	NaN	NaN	ND
22		Grand Theft Auto V	X360	2013.0	Action	9.66	5.14	0.06	1.41	97.0	8.1	M
23		Grand Theft Auto: Vice City	PS2	2002.0	Action	8.41	5.49	0.47	1.78	95.0	8.7	M
		Pokemon Ruby/Pokemon Sapphire	GBA	2002.0	Role-Playing	6.06	3.90	5.38	0.50	NaN	NaN	ND
25		Brain Age 2: More Training in Minutes a Day	DS	2005.0	Puzzle	3.43	5.35	5.32	1.18	77.0	7.1	E
		Pokemon Black/Pokemon White	DS	2010.0	Role-Playing	5.51	3.17	5.65	0.80	NaN	NaN	ND
27		Gran Turismo 3: A-Spec	PS2	2001.0	Racing	6.85	5.09	1.87	1.16	95.0	8.4	E
28		Call of Duty: Modern Warfare 3	X360	2011.0	Shooter	9.04	4.24	0.13	1.32	88.0	3.4	M
		Pokémon Yellow: Special Pikachu Edition	GB	1998.0	Role-Playing	5.89	5.04	3.12	0.59	NaN	NaN	ND
30		Call of Duty: Black Ops 3	PS4	2015.0	Shooter	6.03	5.86	0.36	2.38	NaN	NaN	ND
31		Call of Duty: Black Ops	X360	2010.0	Shooter	9.70	3.68	0.11	1.13	87.0	6.3	M
32		Pokemon X/Pokemon Y	3DS	2013.0	Role-Playing	5.28	4.19	4.35	0.78	NaN	NaN	ND
33												

		name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating
34		Call of Duty: Black Ops II	PS3	2012.0	Shooter	4.99	5.73	0.65	2.42	83.0	5.3	M
35		Call of Duty: Black Ops II	X360	2012.0	Shooter	8.25	4.24	0.07	1.12	83.0	4.8	M
36		Call of Duty: Modern Warfare 2	X360	2009.0	Shooter	8.52	3.59	0.08	1.28	94.0	6.3	M
37		Call of Duty: Modern Warfare 3	PS3	2011.0	Shooter	5.54	5.73	0.49	1.57	88.0	3.2	M
38		Grand Theft Auto III	PS2	2001.0	Action	6.99	4.51	0.30	1.30	97.0	8.5	M
39		Super Smash Bros. Brawl	Wii	2008.0	Fighting	6.62	2.55	2.66	1.01	93.0	8.9	T
40		Mario Kart 7	3DS	2011.0	Racing	5.03	4.02	2.69	0.91	85.0	8.2	E
41		Call of Duty: Black Ops	PS3	2010.0	Shooter	5.99	4.37	0.48	1.79	88.0	6.4	M
42		Grand Theft Auto V	PS4	2014.0	Action	3.96	6.31	0.38	1.97	97.0	8.3	M
43		Animal Crossing: Wild World	DS	2005.0	Simulation	2.50	3.45	5.33	0.86	86.0	8.7	E
44		Halo 3	X360	2007.0	Shooter	7.97	2.81	0.13	1.21	94.0	7.8	M
45		Super Mario 64	N64	1996.0	Platform	6.91	2.85	1.91	0.23	NaN	NaN	ND
46		Pokemon HeartGold/Pokemon SoulSilver	DS	2009.0	Action	4.34	2.71	3.96	0.76	NaN	NaN	ND
47		Pokemon Omega Ruby/Pokemon Alpha Sapphire	3DS	2014.0	Role-Playing	4.35	3.49	3.10	0.74	NaN	NaN	ND
48		Gran Turismo 4	PS2	2004.0	Racing	3.01	0.01	1.10	7.53	89.0	8.5	E
49		Super Mario Galaxy	Wii	2007.0	Platform	6.06	3.35	1.20	0.74	97.0	8.9	E

```
In [99]: def graph (df, year, region, name, axes):  
  
    df = df.query('year_of_release >= @year')  
  
    sales = df.pivot_table(index='platform',  
                           values=region,  
                           aggfunc='sum').nlargest(5, region)  
  
    sales = sales.reset_index()  
  
    sales = (  
        sales.append({'platform': 'Other', region: df[region].sum()  
                     - sales[region].sum()}, ignore_index= True)  
    )  
  
    sales.columns = ['platform', 'sales']  
  
    labels_c=sales.platform  
    colours = {'Wii':'C0', 'NES':'C1', 'GB':'C2', 'DS':'C3', 'X360':'C4',  
               'PS3':'C5', 'PS2':'C6', 'SNES':'C7', 'GBA':'C8',  
               'PS4':'steelblue', '3DS':'orange',  
               'N64':'C11', 'PS':'C12', 'XB':'C13', 'PC':'C14', '2600':'C15', 'PSP':'C16',  
               'XOne':'C17',  
               'WiiU':'C18', 'GC':'C19', 'GEN':'C20', 'DC':'C21', 'PSV':'C22',  
               'SAT':'C23', 'SCD':'C24', 'WS':'C25', 'NG':'C26',  
               'TG16':'C27', '3DO':'C28', 'GG':'C29', 'PCFX':'C30', 'Other':'darkred'}  
  
    sales.plot(kind='pie',  
               y="sales",  
  
               autopct='%1.0f%%',  
               wedgeprops={'linewidth': 3.0, 'edgecolor': 'white'},  
               textprops={'size': 'x-large'},  
               labels= labels_c,  
               colors=[colours[key] for key in labels_c],  
               legend=False,  
               title = f"Популярность платформ в {name} ",  
               ax = axes).set(ylabel='')  
  
    plt.tight_layout()
```

In [100...]

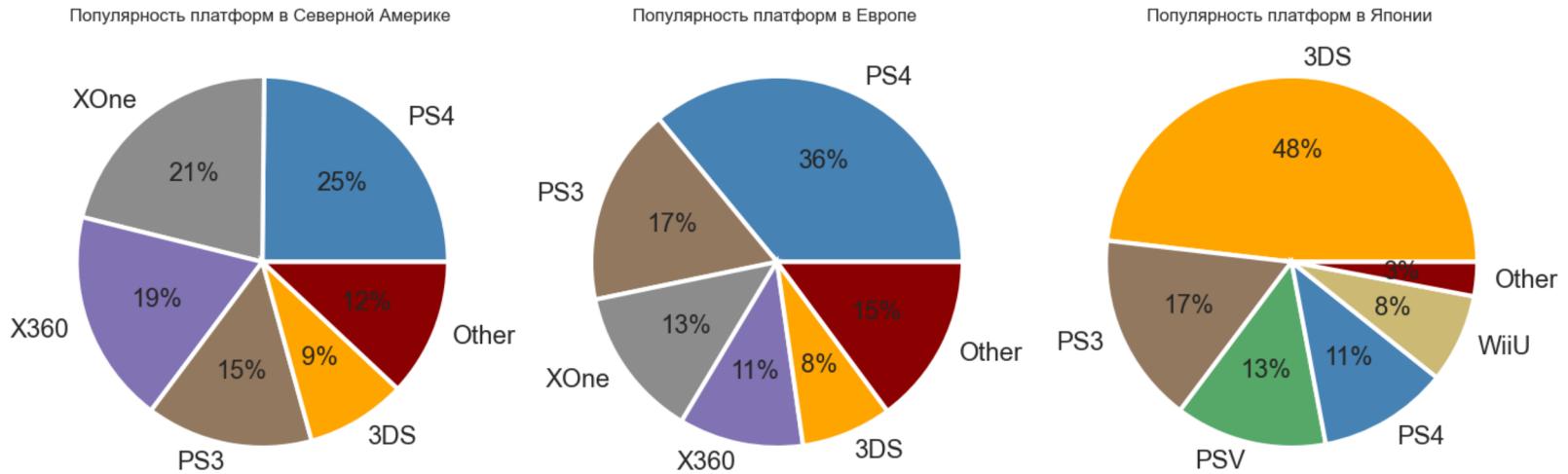
```
fig, axes = plt.subplots(1, 3, figsize = (15,6))
fig.suptitle('Обзор рынка платформ (портрет покупателя)', fontsize = 30, fontweight='bold')

x_year = 2013

graph(data, x_year, 'na_sales', 'Северной Америке', axes[0])
graph(data, x_year, 'eu_sales', 'Европе', axes[1])
graph(data, x_year, 'jp_sales', 'Японии', axes[2])
```

C:\Users\Ilyushik_VY\AppData\Local\Temp\ipykernel_22288\1310682966.py:12: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
sales.append({'platform': 'Other', region: df[region].sum()})
C:\Users\Ilyushik_VY\AppData\Local\Temp\ipykernel_22288\1310682966.py:12: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
sales.append({'platform': 'Other', region: df[region].sum()})
C:\Users\Ilyushik_VY\AppData\Local\Temp\ipykernel_22288\1310682966.py:12: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
sales.append({'platform': 'Other', region: df[region].sum()})

Обзор рынка платформ (портрет покупателя)



In [101...]

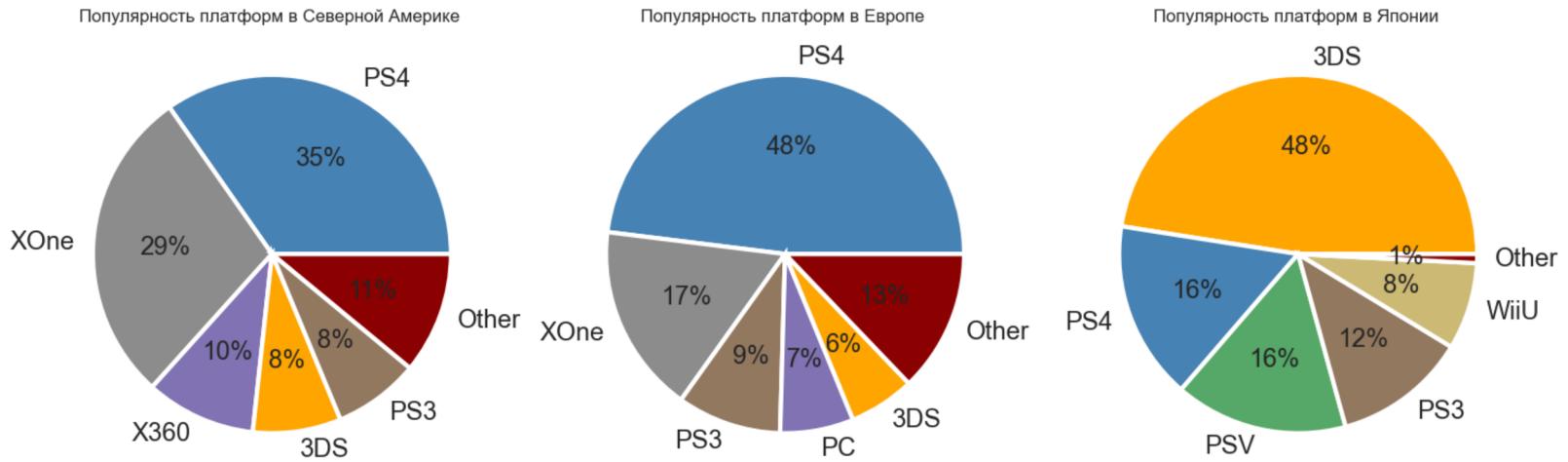
```
fig, axes = plt.subplots(1, 3, figsize = (15,6))
fig.suptitle('Обзор рынка платформ (портрет покупателя)', fontsize = 30, fontweight='bold')

x_year = 2014

graph(data, x_year, 'na_sales', 'Северной Америке', axes[0])
graph(data, x_year, 'eu_sales', 'Европе', axes[1])
graph(data, x_year, 'jp_sales', 'Японии', axes[2])
```

C:\Users\Ilyushik_VY\AppData\Local\Temp\ipykernel_22288\1310682966.py:12: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
sales.append({'platform': 'Other', region: df[region].sum()})
C:\Users\Ilyushik_VY\AppData\Local\Temp\ipykernel_22288\1310682966.py:12: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
sales.append({'platform': 'Other', region: df[region].sum()})
C:\Users\Ilyushik_VY\AppData\Local\Temp\ipykernel_22288\1310682966.py:12: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
sales.append({'platform': 'Other', region: df[region].sum()})

Обзор рынка платформ (портрет покупателя)



In [102...]

```
fig, axes = plt.subplots(1, 3, figsize = (15,6))
fig.suptitle('Обзор рынка платформ (портрет покупателя)', fontsize = 30, fontweight='bold')

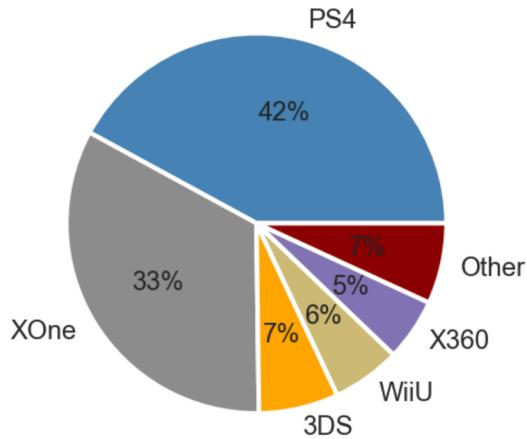
x_year = 2015

graph(data, x_year, 'na_sales', 'Северной Америке', axes[0])
graph(data, x_year, 'eu_sales', 'Европе', axes[1])
graph(data, x_year, 'jp_sales', 'Японии', axes[2])
```

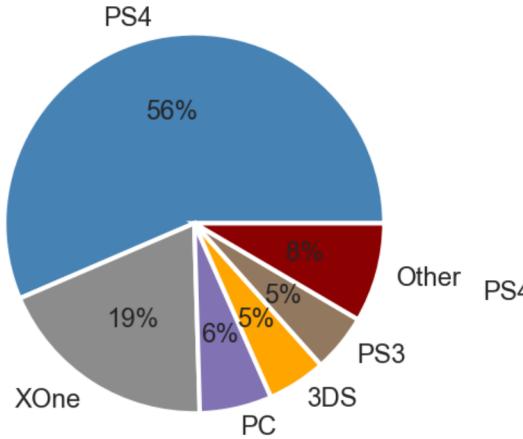
C:\Users\Ilyushik_VY\AppData\Local\Temp\ipykernel_22288\1310682966.py:12: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
sales.append({'platform': 'Other', region: df[region].sum()})
C:\Users\Ilyushik_VY\AppData\Local\Temp\ipykernel_22288\1310682966.py:12: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
sales.append({'platform': 'Other', region: df[region].sum()})
C:\Users\Ilyushik_VY\AppData\Local\Temp\ipykernel_22288\1310682966.py:12: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
sales.append({'platform': 'Other', region: df[region].sum()})

Обзор рынка платформ (портрет покупателя)

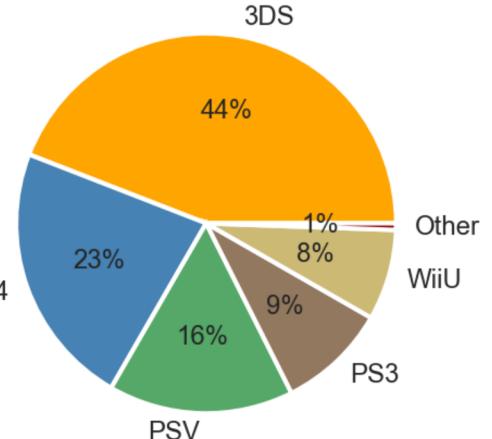
Популярность платформ в Северной Америке



Популярность платформ в Европе



Популярность платформ в Японии



✖ Комментарий ревьюера

Портреты клиентов нарисованы, молодец

Стоит оформить графики раздела ТОП-5:

- для каждого ТОП-5 - построить 3 графика рядом с помощью subplots, оптимальнее сравнивать три региона по каждому виду портрета вместе;
- оформить "двууровневый заголовок" - и у всех трех графиков вместе, и у каждого из трех по отдельности;
- при анализе платформ и жанров стоит все, что не вошло в ТОП-5, объединять в категорию "другие" - так картина анализа будет более полной

Если столкнешься с трудностью выполнения данного пункта — присытай код, который не получился и вопрос, подумаем вместе

<https://proprogs.ru/modules/matplotlib-otobrazhenie-neskolkikh-koordinatnyh-osey-v-odnom-okne>

<https://nagornyy.me/it/vizualizatsiia-dannykh-v-matplotlib/?ysclid=l4q3l4q0p8940570437>

<https://pandas.pydata.org/pandas-docs/stable/reference>

</api/pandas.DataFrame.append.html?highlight=append#pandas.DataFrame.append>

Влияние рейтинга ESRB на продажи в отдельном регионе

In [103...]: `relevant_data.groupby('rating')['eu_sales'].sum().sort_values(ascending=False)`

Out[103]:

rating	eu_sales
M	93.44
ND	58.95
E	58.06
T	34.07
E10+	26.16

Name: eu_sales, dtype: float64

```
In [104...]: relevant_data.groupby('rating')['na_sales'].sum().sort_values(ascending=False)
```

```
Out[104]: rating
M      96.42
ND     64.72
E      50.74
T      38.95
E10+    33.23
Name: na_sales, dtype: float64
```

```
In [105...]: relevant_data.groupby('rating')['jp_sales'].sum().sort_values(ascending=False)
```

```
Out[105]: rating
ND     56.90
T      14.78
E      8.94
M      8.01
E10+    4.46
Name: jp_sales, dtype: float64
```

Зависимость проданных копий и рейтинга видеоигр. В Северной Америке и Европейском Союзе абсолютными лидерами являются игры с рейтингом M (mature), то есть рынок больше ориентирован людей возрастом 17+. В Японии, наибольшее число проданных копий без рейтинга. Это можно объяснить тем, что рейтинг не актуален для рынка Японии, где существуют свои компании, выставляющие рейтинг видеоиграм.

Проверка гипотез

Перед проверкой гипотез необходимо избавится от пропусков в столбце `user_score`, иначе не получим результат проверки гипотез.

```
In [106...]: relevant_data_h = relevant_data.dropna(subset=['user_score'])
```

✓ Комментарий ревьюера

Важно удалить пропуски и «заглушки» перед проведением теста, молодец

Сформулируем нулевую и альтернативную гипотезы:

- H0 - средние пользовательские рейтинги платформ Xbox One и PC одинаковые;
- H1 - средние пользовательские рейтинги платформ Xbox One и PC разные;

✓ **Комментарий ревьюера**

Гипотезы сформулированы верно, для акцентирования можно использовать в нулевой гипотезе слово равны

```
In [107]: xbox = relevant_data_h.loc[relevant_data_h['platform'] == 'XOne', 'user_score']
```

```
In [108]: pc = relevant_data_h.loc[relevant_data_h['platform'] == 'PC', 'user_score']
```

```
In [109]: alpha = .05
```

```
In [110]: results = st.ttest_ind(xbox, pc)
print('p-значение:', results.pvalue)
if results.pvalue < alpha:
    print("Отвергаем нулевую гипотезу")
else:
    print("Не получилось отвергнуть нулевую гипотезу")
```

р-значение: 0.10450507919348415

Не получилось отвергнуть нулевую гипотезу

Применив t-тест для сравнения средних пользовательских рейтингов для платформ Xbox One и PC, в результате получили, что их средние пользовательские рейтинги близки друг к другу.

В нашем случае р-значение составляет 0.10450507919348415, то есть, при условии, что нулевая гипотеза верна, и мы берем случайные выборки из нашей совокупности, тогда в 10% случаев мы получим результат, подтверждающий наше представление о том, что средние пользовательские рейтинги платформ Xbox One и PC одинаковые. Вероятность не маленикая и, в таком случае, нулевую гипотезу отвергать нельзя.



Комментарий ревьюера в2

Приведу пример и теорию для понимания формулировок и интерпретации итогов проведения гипотез

Задача. Приведены два датасета: сумма покупок, совершенных за месяц посетителями, привлеченными по двум разным каналам. В вашем распоряжении случайная выборка из 30 покупок для каждого канала.

H0 - средние чеки равны

H1 - средние чеки НЕ равны

Да сама формулировка нулевой и альтернативной гипотезы звучит именно так, но результат теста интерпретируется другими словами

Из теории на тренажере

Формулирование двусторонних гипотез.

Никакие экспериментально полученные данные никогда не подтверждают какую-либо гипотезу. Это наше фундаментальное ограничение. Данные могут лишь не противоречить ей или, наоборот, показывать крайне маловероятные результаты (при условии, что гипотеза верна). Но и в том, и в другом случае нет оснований утверждать, что выдвинутая гипотеза доказана. Допустим, данные гипотезе не противоречат, тогда мы её не отвергаем. Если же мы приходим к выводу, что получить такие данные в рамках этой гипотезы вряд ли возможно, у нас появляется основание отбросить эту гипотезу.

P-значение (англ. P-value) — величина, используемая при тестировании статистических гипотез.

Фактически это вероятность ошибки при отклонении нулевой гипотезы (ошибки первого рода)
пример ниже

In [111...]

```
# Приведены два датасета: сумма покупок, совершённых за месяц посетителями ...

sample_1 = [3071, 3636, 3454, 3151, 2185, 3259, 1727, 2263, 2015,
2582, 4815, 633, 3186, 887, 2028, 3589, 2564, 1422, 1785,
3180, 1770, 2716, 2546, 1848, 4644, 3134, 475, 2686,
1838, 3352]
sample_2 = [1211, 1228, 2157, 3699, 600, 1898, 1688, 1420, 5048, 3007,
509, 3777, 5583, 3949, 121, 1674, 4300, 1338, 3066,
3562, 1010, 2311, 462, 863, 2021, 528, 1849, 255,
1740, 2596]
alpha = .05 # критический уровень статистической значимости
# если p-value окажется меньше него - отвергнем гипотезу
results = st.ttest_ind(
sample_1,
sample_2)
print('p-значение: ', results.pvalue)
if (results.pvalue < alpha):
    print("Отвергаем нулевую гипотезу")
else:

    print("Не получилось отвергнуть нулевую гипотезу")
```

p-значение: 0.1912450522572209

Не получилось отвергнуть нулевую гипотезу

🍕 Комментарий ревьюера в2

Интерпретация результата:

Полученное значение p-value говорит о том, что хотя средний чек пришедших из разных каналов и неодинаков, **с вероятностью в почти 19% такое или большее различие можно получить случайно.**

Это явно слишком большая вероятность, чтобы делать вывод о значимом различии между средними чеками.

А если p-value будет равно 0,9999, то это значит, что с вероятностью почти 100% такое различие можно получить случайно — то есть почти никогда :) (но учитываем, что тест проводится на выборке из генеральной совокупности, все может поменяться)

- H0 - средние пользовательские рейтинги жанров Action и Sports равны;
- H1 - средние пользовательские рейтинги жанров Action и Sports не равны;

```
In [112]: action = relevant_data_h.loc[relevant_data_h['genre'] == 'Action', 'user_score']
```

```
In [113]: sports = relevant_data_h.loc[relevant_data_h['genre'] == 'Sports', 'user_score']
```

```
In [114]: action.mean()
```

```
Out[114]: 6.760606060606054
```

```
In [115]: sports.mean()
```

```
Out[115]: 5.225196850393697
```

```
In [116]: alpha = .05
```

In [117...]

```
results = st.ttest_ind(action, sports)
print('р-значение:', results.pvalue)
if results.pvalue < alpha:
    print("Отвергаем нулевую гипотезу")
else:
    print("Не получилось отвергнуть нулевую гипотезу")
```

р-значение: 2.8711147985105864e-19
Отвергаем нулевую гипотезу

Получив р-значение = 2.8711147985105864e-19, можно сказать, что вероятность получить устраивающий нас результат слишком мал, поэтому нулевую гипотезу стоит отклонить

С помощью *t*-теста опровергли предположение о том, что средние пользовательские рейтинги жанров *Action* и *Sports* разные.

Применив t-тест для подтверждения гипотезы о равенстве средних пользовательских рейтингов жанров Action и Sports, получили р-значение 2.8711147985105864e-19, таким образом опровергнув нашу гипотезу.

Общий вывод

Цель проекта - анализ рынка компьютерных игр, определить лидеров рынка в разных регионах (Северная Америка, Европейский Союз, Япония и оставшаяся часть планеты) в зависимости от игровых платформ, жанров, рейтинга игр.

На этапе предобработки был изменен тип данных на вещественный в столбце user_score. Найденные пропуски в таблице были оставлены без изменений, так как каждое значение как индивидуальная характеристика игры и заполнение пропусков может привести к некорректному анализу.

Исследовательский анализ данных.

1. Был проведен анализ выпуска игр по годам в период 1980-2016гг.
 - 1.1 Большой рост индустрии компьютерных игр, начиная с 2002 года и пиком в 2008-2009 годах
 - 1.2 Падение количества игр в 2012 году, что можно объяснить изменением политики производителей компьютерных игр, и взамен большого количества сконцентрировались на качестве.
2. Проведен анализ продаж по платформам.
 - 2.1 Абсолютным лидером по количеству проданных копий за весь период времени с 1980 по 2016 годы, является платформа PS2;
 - 2.2 Жизненный цикл игровых платформ составляет 7 лет;
 - 2.4 К наиболее перспективным платформам на ближайшее будущее можно отнести PS4 и Xbox One.
3. Проведен анализ влияния отзывов пользователей и критиков на продажи. Взаимосвязи между отзывами пользователей продажами не удалось обнаружить. Отзывы критиков и продажи коррелируются между собой, коэффициент корреляции между ними равен 0.3-0.4, в зависимости от типа платформы.
4. Проведен анализ игр по жанрам - по общему количеству игр в жанре и по продажам в жанре. Лидером по количеству игр, и по общему количеству проданных копий является жанр Action. Жанр Shooter лидирует по стабильности продаж - 3/4 игр в жанре продавались в интервале до 1.6 млн.шт.
5. Был проведен анализ пользователей по отдельным регионам (Северная Америка, Европейский Союз, Япония). Среди платформ в Северной Америке и Европе лидерами являются PS4, хотя в Северной Америке Xbox One незначительно отстали от лидера. Для Японии абсолютным лидером является платформа Nintendo 3DS. Что касается игровых жанров, в Северной Америке и Европе лидером является жанр Action. Для жителей Японии больший интерес представляет жанр Role Playing.
6. Было определено влияние рейтинга ESRB. Для Северной Америки и Европы абсолютными лидерами являются игры с рейтингом M (mature), то есть рынок больше ориентирован людей возрастом 17+. Определить

однозначного лидера на Японском рынке не удалось, поскольку рейтинг ESRB не актуален для Японии и в результате наибольшее число продаж пришлось на игры без рейтинга. На втором месте по продажам находятся игры с рейтингом T.

7. Была проведена проверка двух гипотез

- была подтверждена гипотеза, что средние пользовательские рейтинги платформ Xbox One и PC одинаковые;
- гипотеза о том, что средние пользовательские рейтинги жанров Action и Sports равны была опровергнута.



Комментарий ревьюера

Может пригодиться

[Подборка статей о работе с библиотеками для анализа данных на языке Python](#)

[Визуализация](#)

[Искусство статистики](#)

[Постер «Графики, которые убеждают всех»](#)

В помощь — как реализовать интерактивный план проекта вручную (для собственных проектов),
смотри по [ссылке](#)

пара ссылок и по разделам проекта можно будет переходить без пролистывания всего кода, особенно
актуально на проектах длина которых > 10 страниц (и там где не установлен плагин TOC)