## ✔ Комментарий ревьюера в2

Привет, Ирина :) Спасибо за работу, все исправления и комментарии

На проекте мы изучали и практиковали навыки из разделов о/об:

- выдвижении гипотез о пропусках в данных, заполнении или удалении,
- работе с категориальными данными
- подготовке данных для решения главной задачи проекта
- исследовании неочищенных данных,
- подборе и создании графиков,
- программировании,
- автоматизации однотипных действий,
- выстраивании структуры проекта и обеспечении аккуратности кода,
- анализе данных,
- определении прибыльности и устойчивости продаж по выбранному параметру (жанр, платформа) на диаграмме размаха,
- составлении портрета пользователя,
- формировании рекомендаций для бизнеса

**Навыки первого модуля освоены хорошо**

**Поздравляю с успешно сданным сборным проектом и завершением первого модуля**
**на факультете дата-аналитики Я.Практикум**

## Успехов в дальнейшей учебе 🤝

добавил бонус

### ✍ Комментарий Ирины 1

Привет, Ринат. Спасибо за ревью. Чувствую еще денек на исправления у меня уйдет. Буду внимательно и последовательно решать. Погнали.

### Комментарий ревьюера

Привет, Ирина :) Спасибо, что прислала 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 🚀

### Примеры комментариев

### 🍕 Пример комментария - совета, здесь м.б. просто ссылка

Тут всего такого разного и вкусного :), есть способы прокачать проект визуализациями (ценит большинство "боссов")

"Библиотека Matplotlib" доступна для скачивания БЕСПЛАТНО! На сайте много полезных материалов, мне самому очень помогло в свое время, до сих пор подсматриваю :)

### ⚠ Пример оформления некритичного комментария

Рекомендации, которые, хоть и не обязательны, но точно сделают ревью лучше

### ❌ Пример оформления комментария к блоку(строке) программного кода (или выводу), который стоит переделать

Отправлен не тот проект, напиши в своих комментариях, что случилось? жду — **это пример**

### ✔ Пример оформления комментария, который нравится большинству студентов

Круто, молодец, отлично, логично, или — 👌, 👍, или — выводы отвечают на все вопросы к данным и проекту

**✍ Комментарий Ирины 1**

Спасибо за книгу. В процессе много гуглила, чтобы сделать графики такими как хочется или хотя бы похожими на такие, как хочется.

👍

## Для твоих вопросов или комментариев оставлю такую ячейку, чтобы было удобнее взаимодействовать на проекте

**✍ Комментарий студента**

... , вот мой вопрос ...

## Исследование успешности компьютерных игр

## Описание проекта

**Цели проекта** - выявить определяющие успешность игры закономерности. Это позволит сделать ставку на потенциально популярный продукт и спланировать рекламные кампании на 2017 год. (Сейчас декабрь 2016 года)

**Заказчик проекта** - интернет-магазин "Стримчик" продающий по всему миру компьютерные игры

**Входные данные:** данные из открытых источников, доступны исторические данные о продажах игр, оценки пользователей и экспертов, жанры и платформы (например, Xbox или PlayStation). Данные до 2016 года. Путь к файлу: /datasets/games.csv.

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

**План исследования:** 1) Ознакомиться с данными

2) Подготовить данные:

- привести к общему виду названия столбцов
- преобразовать типы данных
- обработать пропуски, дубликаты

3) Провести анализ данных:

- изучить количество выпущенных игр в разные годы
- изучить продажи игр по платформам
- определение актуального периода данных, для построения прогноза на 2017 год
- определение потенциально прибыльных игровых платформ
- изучить характер продаж игр в разбивке по платформам
- изучить влияние отзывов на продажи внутри одной популярной платформы
- изучить общее распределение игр по жанрам

4) Составить портрет пользователя каждого региона:

- самые популярные платформы (топ-5)
- самые популярные жанры (топ-5)
- влияние возрастного рейтинга на продажи в отдельном регионе

5) Общие выводы

**✔ Комментарий ревьюера**

Ясная и четкая вступительная часть проекта

## Исходные данные

### ✍ Комментарий Ирины 1

Принято. Отделила блок загрузки библиотек от загрузки датасета

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
data = pd.read_csv('/datasets/games.csv')
data.head(10)
```

| | 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 | | | | | | | | | | |

### ⚠ Комментарий ревьюера

стоит делить блок загрузки библиотек и код загрузки датасета, в случае необходимости добавления новых библиотек не придется загружать весь датасет заново и перезапускать проект целиком

```
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
```

### ✔ Комментарий ревьюера

обзор данных проведен корректно

### ⚠ Комментарий ревьюера

не помешает добавить вывод после первого знакомства с данными

### ✍ Комментарий Ирины 1

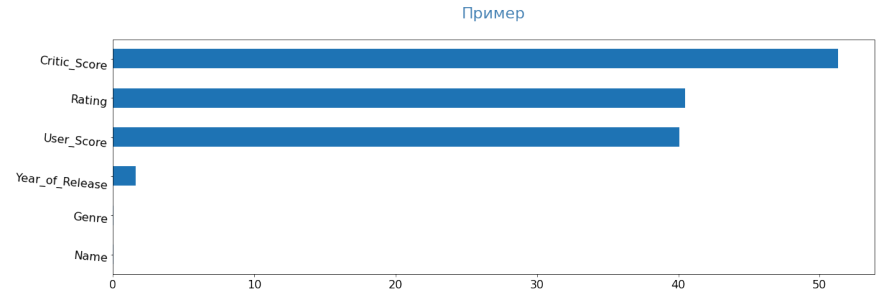
Вывод: первое знакомство с данными показывает:

- объем данных 16715 строк

- в данных наблюдаются пропуски в стобцах: наименование, год релиза, жанр, оценки пользователей и критиков, возрастной рейтинг
- возможно понадобится замена типов данных в стобцах на более подходящие: год выпуска, оценка пользователя и оценка критиков

## Подготовка данных

```
# check
# пропущенные значения бары

def pass_value_barh(df):
    try:
        (
            (df.isna().mean()*100)
            .to_frame()
            .rename(columns = {0:'space'})
            .query('space > 0')
            .sort_values(by = 'space', ascending = True)
            .plot(kind = 'barh', figsize = (19,6), rot = -5, legend = False, fontsize = 16)
            .set_title('Пример' + "\n", fontsize = 22, color = 'SteelBlue')
        );
    except:
        print('пропусков не осталось :) или произошла ошибка в первой части функции ')

pass_value_barh(data)
```




### Комментарий ревьюера

Наглядность представления информации одна из важных составляющих работы дата-аналитика или дата-сайентиста

**мой график оформлен не совсем корректно, сможешь отметить, что стоило бы исправить в графике?**

### Комментарий Ирины 1

Ринат, в данном оформлении мне не нравится следующее:

- нет подписей оси X и Y, не понятно какие данные визуализируются
- нет корректного заголовка, т.к. заголовок "Пример" не отображает сути визуализированных данных.
- не понятно зачем названия столбцов повернуты под углом? Чтобы сократить место ими занимаемой. На мой взгляд, так нужно делать только если места реально не хватает, когда надписи наползают друг на друга. В других случаях горизонтальное и вертикальное отображение подписей более уместное. Как говорится «Quadratisch, praktisch, gut».

### Комментарий ревьюера в2

Принимается

За поворот текста отвечает параметр rot

```
df.isna().mean()*100 — для расчета процента пропусков в колонке
```

На графике мы оцениваем масштаб проблемы с пропусками и возможное совпадение % пропущенных значений в колонках

добавить подписи к осям мешает строка кода

```
.set_title('Пример' + "\n", fontsize = 22, color = 'SteelBlue')
```

её можно заменить на

```
plt.title('Пропущенные значения, %' + "\n", fontsize=22, color='SteelBlue')
plt.xlabel('Процент пропусков', fontsize=22)
plt.ylabel('Столбцы с пропусками')
```

плюс можно установить шкалу от 0 до 100

```
plt.xlim(0, 100)
```

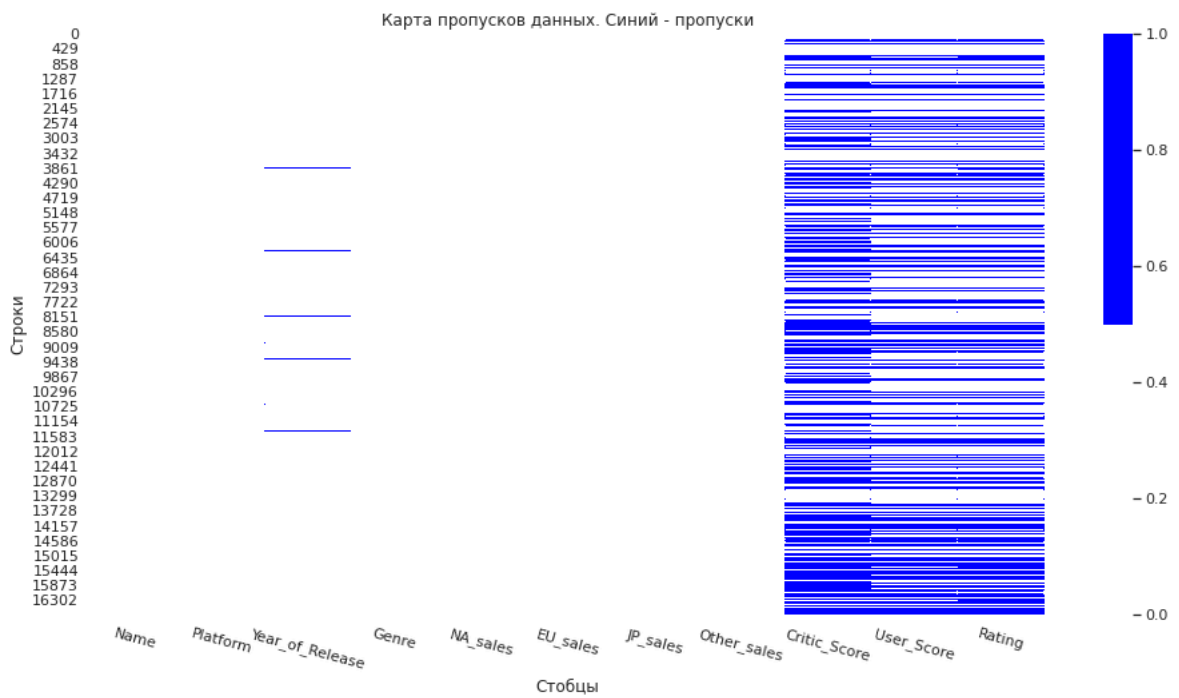
### ✍ Комментарий Ирины 1

Ринат, как тебе такая визуализация пропусков в данных?

### ✔ Комментарий ревьюера в2

Нормальный график для исследования пропусков 🤝

```
cols = data.columns
# определяем цвета
# желтый - пропущенные данные, синий - не пропущенные
colours = ['#ffffff', '#0000ff']
sns.set(rc = {'figure.figsize':(16,8)})
sns.heatmap(data[cols].isnull(), cmap=sns.color_palette(colours))
plt.title('Карта пропусков данных. Синий - пропуски')
plt.xticks(rotation=-15)
plt.xlabel('Стобцы')
plt.ylabel('Строки')
plt.grid()
plt.show()
```




## Замена названий столбцов (приведение к нижнему регистру)

```
# сохраним первоначальные данные в отдельном фрейме row_data, а работать будем с данными в  data
row_data = data.copy()
# выведем названия столбцов
data.columns
```

```
Index(['Name', 'Platform', 'Year_of_Release', 'Genre', 'NA_sales', 'EU_sales',
       'JP_sales', 'Other_sales', 'Critic_Score', 'User_Score', 'Rating'],
      dtype='object')
```

```
# приведем названия столбцов к нижнему регистру
data.columns = data.columns.str.lower()
data.columns
```

```
Index(['name', 'platform', 'year_of_release', 'genre', 'na_sales', 'eu_sales',
       'jp_sales', 'other_sales', 'critic_score', 'user_score', 'rating'],
      dtype='object')
```

## Преобразование типов данных

Преобразование типов данных необходимо в следующих столбцах:

- year_of_release заменить float на int, т.к. год выпуска не может быть дробным
- user_score из объектного в float, т.к. это не строковое значение, а числовое
- critic_score из float в int, если не встречаются значения с числом после запятой

Чтобы преобразовать типы иногда требуестя чтобы в столбцах не было пропусков данных. Посмотрим долю пропусков в данных

### Преобразование типа данных для года выпуска - year_of_release

```
data['year_of_release'].count()
print('Количество пропусков в данных год выпуска:', data['year_of_release'].isna().sum() )
print(f'Доля пропусков: { data["year_of_release"].isna().mean():.0%}')
```

```
Количество пропусков в данных год выпуска: 269
Доля пропусков: 2%
```

Доля пропусков всего 2%. Можно эти данные не учитывать. Удалим 2% данных и не будем плакать по этому поводу.

**⚠ Комментарий ревьюера**

Решение верное, перед удалением стоит посмотреть на записи — может они по 100 млн. продаж принесли ...

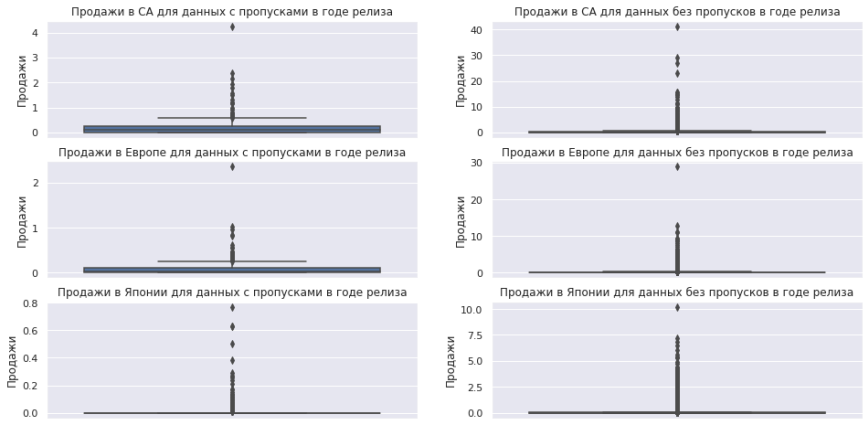
На этом проекте мы работаем с продажами, поэтому включается другой фактор, кроме процента потерь ...

**✍ Комментарий Ирины 1**

Прежде чем удалять данные посмотрим нет ли среди них вносящих существенный вклад в продажи

```
try:

        plt.subplot(3, 2, 1)
        plt.title('Продажи в СА для данных с пропусками в годе релиза')
        sns.boxplot(y= data.loc[ data['year_of_release'].isna(), 'na_sales' ])
        plt.ylabel('Продажи')


        plt.subplot(3, 2, 2)
        plt.title('Продажи в СА для данных без пропусков в годе релиза')
        sns.boxplot(y= data.loc[ ~data['year_of_release'].isna(),'na_sales' ])
        plt.ylabel('Продажи')


        plt.subplot(3, 2, 3)
        plt.title('Продажи в Европе для данных с пропусками в годе релиза')
        sns.boxplot(y= data.loc[ data['year_of_release'].isna(), 'eu_sales' ])
        plt.ylabel('Продажи')

        plt.subplot(3, 2, 4)
        plt.title('Продажи в Европе для данных без пропусков в годе релиза')
        sns.boxplot(y= data.loc[ ~data['year_of_release'].isna(), 'eu_sales' ])
        plt.ylabel('Продажи')

        plt.subplot(3, 2, 5)
        plt.title('Продажи в Японии для данных с пропусками в годе релиза')
        sns.boxplot(y= data.loc[ data['year_of_release'].isna(), 'jp_sales' ])
        plt.ylabel('Продажи')

        plt.subplot(3, 2, 6)
        plt.title('Продажи в Японии для данных без пропусков в годе релиза')
        sns.boxplot(y= data.loc[ ~data['year_of_release'].isna(), 'jp_sales' ])
```

```
        plt.ylabel('Продажи')

        plt.show()
except:
    print('данные уже удалены, попробуйте перезагрузить данные снова')
```




### ✍ Комментарий Ирины 1

Ринат, ниже попытка отобразить теже самые данные на одном графике. Для этого потребовалось переорганизовать данные, что заняло время и мне не нравится, что я использую повторяющийся код. Возможно сделать это более элегантно?

```
country_sets = ['na_sales', 'eu_sales', 'jp_sales']
country_name = ['Северная Америка','Европа','Япония']
i = [0,1,2]

for i, i_sets, i_name in zip(i, country_sets, country_name):
    print(i, i_sets, i_name)
```

```
0 na_sales Северная Америка
1 eu_sales Европа
2 jp_sales Япония
```

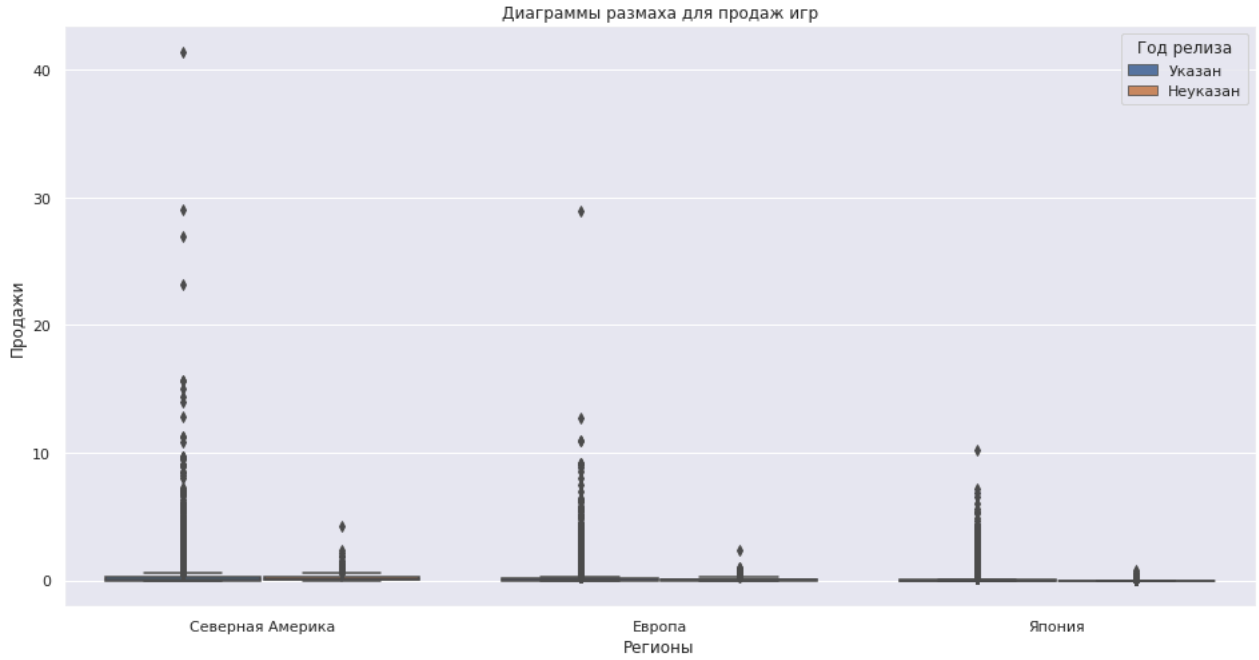
### ✔ Комментарий ревьюера в2

Кажется тебе пригодится пример использования zip

[9 Уровней применения функции zip в Python]

```
try:
    data_temp = data.copy()
    data_temp.loc[data_temp['year_of_release'].isna(),'year_empty' ] = 'Неуказан'
    data_temp.loc[~data_temp['year_of_release'].isna(),'year_empty' ] = 'Указан'

    temp = pd.DataFrame()
    temp_na = pd.DataFrame()
    temp_eu = pd.DataFrame()
    temp_jp = pd.DataFrame()

    temp_na['sales'] = data_temp['na_sales']
    temp_na['region'] = 'Северная Америка'
    temp_na['year_empty'] = data_temp['year_empty']

    temp_eu['sales'] = data_temp['eu_sales']
    temp_eu['region'] = 'Европа'
    temp_eu['year_empty'] = data_temp['year_empty']

    temp_jp['sales'] = data_temp['jp_sales']
    temp_jp['region'] = 'Япония'
    temp_jp['year_empty'] = data_temp['year_empty']

    temp=temp.append(temp_na)
    temp=temp.append(temp_eu)
```

```
        temp=temp.append(temp_jp)
    except:
        print('Данные уже удалены, попробуйте перезагрузить данные снова!')


    sns.boxplot(data=temp, y='sales', x='region', hue='year_empty')
    plt.title('Диаграммы размаха для продаж игр')
    plt.ylabel('Продажи')
    plt.xlabel('Регионы')
    plt.legend(title='Год релиза')
    plt.show()
```




### ✍ Комментарий Ирины 1

Значения выбросов в данных, которые хотим удалить состаляют менее 10% от выбросов в целом для всех данных. Для Северной Америки 4 млн копий против 40 млн копий. Для Европы 2 млн копий против 30 млн копий. В Японии 0.8млн копий против 10 млн копий. Данные можно смело удалять.

### ✔ Комментарий ревьюера в2

Сначала визуальный контроль, т.к. он самый дешевый (по ресурсам), т.е. применяем фильтрацию по суммам продаж и смотрим на сами строки с данными. Затем можно усложнять, если первичный контроль покажет значительные суммы

```
#Удалим строки с пропусками в столбце year_of_release
data = data.dropna(subset=['year_of_release'])
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16446 entries, 0 to 16714
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   name             16444 non-null  object
 1   platform         16446 non-null  object
 2   year_of_release  16446 non-null  float64
 3   genre            16444 non-null  object
 4   na_sales         16446 non-null  float64
 5   eu_sales         16446 non-null  float64
 6   jp_sales         16446 non-null  float64
 7   other_sales      16446 non-null  float64
 8   critic_score     7983 non-null   float64
 9   user_score       9839 non-null   object
 10  rating           9768 non-null   object
dtypes: float64(6), object(5)
memory usage: 1.5+ MB
```

```
#преобразуем тип данных year_of_release в int
data['year_of_release'] = data['year_of_release'].astype(int)
#выведем данные до и после преобразования типа
print(row_data['Year_of_Release'].unique())
data['year_of_release'].unique()
```

```
[2006. 1985. 2008. 2009. 1996. 1989. 1984. 2005. 1999. 2007. 2010. 2013.
 2004. 1990. 1988. 2002. 2001. 2011. 1998. 2015. 2012. 2014. 1992. 1997.
 1993. 1994. 1982. 2016. 2003. 1986. 2000.   nan 1995. 1991. 1981. 1987.
 1980. 1983.]
array([2006, 1985, 2008, 2009, 1996, 1989, 1984, 2005, 1999, 2007, 2010,
       2013, 2004, 1990, 1988, 2002, 2001, 2011, 1998, 2015, 2012, 2014,
       1992, 1997, 1993, 1994, 1982, 2016, 2003, 1986, 2000, 1995, 1991,
       1981, 1987, 1980, 1983])
```

## Преобразование типа данных для user_score - оценка пользователей

Преобразуем тип данных для user_score - оценка пользователей, максимум 10. В данных наблюдаются пропуски, поэтому придется сначала разобраться с ними, а после преобразовывать тип данных. Для начала посмотрим на характер данных

```
data['user_score'].unique()
```

```
array(['8', nan, '8.3', '8.5', '6.6', '8.4', '8.6', '7.7', '6.3', '7.4',
       '8.2', '9', '7.9', '8.1', '8.7', '7.1', '3.4', '5.3', '4.8', '3.2',
       '8.9', '6.4', '7.8', '7.5', '2.6', '7.2', '9.2', '7', '7.3', '4.3',
       '7.6', '5.7', '5', '9.1', '6.5', 'tbd', '8.8', '6.9', '9.4', '6.8',
       '6.1', '6.7', '5.4', '4', '4.9', '4.5', '9.3', '6.2', '4.2', '6',
       '3.7', '4.1', '5.8', '5.6', '5.5', '4.4', '4.6', '5.9', '3.9',
       '3.1', '2.9', '5.2', '3.3', '4.7', '5.1', '3.5', '2.5', '1.9', '3',
       '2.7', '2.2', '2', '9.5', '2.1', '3.6', '2.8', '1.8', '3.8', '0',
       '1.6', '9.6', '2.4', '1.7', '1.1', '0.3', '1.5', '0.7', '1.2',
       '2.3', '0.5', '1.3', '0.2', '0.6', '1.4', '0.9', '1', '9.7'],
      dtype=object)
```

Помимо числовых значений в данных встречаются пропуски и некоторые данные содержат строку tbd. По данным google эта абревиатура обозначает To Be Determined - будет определено, т.е. оценка еще не определена. В данном случае можно предположить, что пропуски и tbd имеют одинаковую природу - оценки пользователей нет и можно заменить на 0.

Прежде, чем делать такую замену хочется посмотреть на долю пропусков и долю tbd

### ✔ Комментарий ревьюера

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

```
print('Количество пропусков в данных оценки пользователей:', data['user_score'].isna().sum() )
print(f'Доля пропусков: { data["user_score"].isna().mean():.0%}')
```

```
Количество пропусков в данных оценки пользователей: 6607
Доля пропусков: 40%
```

```
tbd_count  = data.loc[data['user_score']=='tbd', 'user_score'].value_counts()
tbd_mean = tbd_count/len(data)
print('Количество tbd в данных оценки пользователей:', tbd_count )
print('Доля tbd:',tbd_mean)
```

```
Количество tbd в данных оценки пользователей: tbd    2376
Name: user_score, dtype: int64
Доля tbd: tbd    0.144473
Name: user_score, dtype: float64
```

Заполним пропуски user_score и значения tbd нулями

```
data['user_score'] = data['user_score'].fillna(0)
data.loc[data['user_score']=='tbd','user_score' ]=0
```

```
data['user_score'].isna().sum()
```

```
0
```

```
data.query(' user_score == "tbd"')['user_score'].sum()
```

```
0
```

```
data.loc[data['user_score'] == 0,'user_score'].count()
```

```
8983
```

```
row_data.loc[row_data['User_Score']=="0",'User_Score'].count()
```

```
1
```

До удаления пропусков оценка пользователя со значением ноль была только одна. После Присвоения пропускам и tbd нуля, нулевых значений стало 8983. В дальнейшем анализе их нужно не учитывать, зато теперь строковый тип данных можно преобразовать к float

**✔ Комментарий ревьюера**

Все верно, посмотрели, оценили, заменили, молодец

**⚠ Комментарий ревьюера**

Скажи пожалуйста, какое преимущество ты получаешь от заполнения пропусков в числовых колонках на этом проекте, с этой главной задачей?

```
Это позволит сделать ставку на потенциально популярный продукт и спланировать рекламные кампании.
```

**✍ Комментарий Ирины 1**

В данном случае пропуски заполняются числовым значением, чтобы проще было работать с данными: возможно было преобразовывать типы. Но замена пропусков в данном случае нулями не значит, что эти нули имеют какую-то смысловую нагрузку, кроме как показать, что это пропуски и их нужно не учитывать в анализе, что в моей работе в дальнейшем и сделано. В принципе срезы данных можно делать и по нулю и по NaN, так что числовые значения тут только для удобства преобразования типа!

**⚠ Комментарий ревьюера в2**

На этом проекте с такой главной задачей нули нам только мешают

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

```
#проверим, какой тип данных получился
data['user_score'].dtype
```

```
dtype('float64')
```

## Преобразование типа данных для critic_score - оценка критиков

Посмотрим на характер данных critic_score и преобразуем float в int, если не встречаются значения с числом после запятой

```
data['critic_score'].sort_values().unique()
```

```
array([13., 17., 19., 20., 21., 23., 24., 25., 26., 27., 28., 29., 30.,
       31., 32., 33., 34., 35., 36., 37., 38., 39., 40., 41., 42., 43.,
       44., 45., 46., 47., 48., 49., 50., 51., 52., 53., 54., 55., 56.,
       57., 58., 59., 60., 61., 62., 63., 64., 65., 66., 67., 68., 69.,
       70., 71., 72., 73., 74., 75., 76., 77., 78., 79., 80., 81., 82.,
       83., 84., 85., 86., 87., 88., 89., 90., 91., 92., 93., 94., 95.,
       96., 97., 98., nan])
```

Все данные целочисленные и опять встречаются пропуски, посмотрим на долю пропусков.

```
print('Количество пропусков в данных год выпуска:', data['critic_score'].isna().sum() )
print(f'Доля пропусков: { data["critic_score"].isna().mean():.0%}')
```

```
Количество пропусков в данных год выпуска: 8463
Доля пропусков: 51%
```

Заменим пропуски на нули

```
data['critic_score'] = data['critic_score'].fillna(0)
data['critic_score'].isna().sum()
```

```
0
```

Преобразуем тип в int

```
data['critic_score'] = data['critic_score'].astype(int)
```

```
data['critic_score'].dtype
```

```
dtype('int64')
```

## Обработка оставшихся пропусков

### Пропуски в названии игры name

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16446 entries, 0 to 16714
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   name             16444 non-null  object
 1   platform         16446 non-null  object
 2   year_of_release  16446 non-null  int64
 3   genre            16444 non-null  object
 4   na_sales         16446 non-null  float64
 5   eu_sales         16446 non-null  float64
 6   jp_sales         16446 non-null  float64
 7   other_sales      16446 non-null  float64
 8   critic_score     16446 non-null  int64
 9   user_score       16446 non-null  float64
 10  rating           9768 non-null   object
dtypes: float64(5), int64(2), object(4)
memory usage: 1.5+ MB
```

Наблюдаются пропуски в названии игры - всего два пропуска, можно их удалить.

```
#удалим данные с пропусками в названии игры. Два пропуска вообще ни на что не влияют
data = data.dropna(subset=['name'])
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16444 entries, 0 to 16714
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   name             16444 non-null  object
 1   platform         16444 non-null  object
 2   year_of_release  16444 non-null  int64
 3   genre            16444 non-null  object
 4   na_sales         16444 non-null  float64
 5   eu_sales         16444 non-null  float64
 6   jp_sales         16444 non-null  float64
 7   other_sales      16444 non-null  float64
 8   critic_score     16444 non-null  int64
 9   user_score       16444 non-null  float64
 10  rating           9768 non-null   object
dtypes: float64(5), int64(2), object(4)
memory usage: 1.5+ MB
```

### Пропуски в rating возрастной рейтинг

```
data['rating'].value_counts()
```

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

- EC - для детей от 3 лет
- E - для всех возрастных категорий
- M - для лиц старше 17 лет
- T - для лиц старше 13 лет
- E10+ - для лиц старше 10 лет
- AO - для взрослых старше 18 лет

- RP - категория не присвоена
- K-A - для детей (был переименован в E)

```
print('Количество пропусков в данных возрастного рейтинга:', data['rating'].isna().sum() )
print(f'Доля пропусков: { data["rating"].isna().mean():.0%}')
```

```
Количество пропусков в данных возрастного рейтинга: 6676
Доля пропусков: 41%
```

Пропуски можно вообще не заполнять, т.к. данные не учавствуют в дальнейшем исследовании или заполнить отдельной категорией, например NV - not value

```
data['rating'] = data['rating'].fillna('NV')
data['rating'].isna().sum()
```

```
0
```

### ✔ Комментарий ревьюера

Отличное решение для столбца, который содержит категориальные данные, поможет определить ключевое различие в потрете пользователя

```
data['rating'] = data['rating'].fillna('NV')
```

Можно посмотреть на частотность использования категорий рейтинга, по отношению к остальным

```
EC        8
K-A       3
RP        1
AO        1
```

```
print('Частотность использования каждого вида рейтинга')
(data['rating'].value_counts()/len(data)*100).astype(int)
```

```
Частотность использования каждого вида рейтинга
NV      40
E       23
T       17
M        9
E10+     8
EC       0
K-A      0
AO       0
RP       0
Name: rating, dtype: int64
```

### ✍ Комментарий Ирины 1

Вывод:

- частота пропусков в возрастном рейтинге 40%,
- частота использования рейтинга E - для всех возрастных категорий - 23%,
- частота использования рейтинга T - для лиц старше 13 лет - 17%
- частота использования рейтинга M - для лиц старше 17 лет - 9%
- частота использования рейтинга E10+ - для лиц старше 10 лет - 8%
- частота испльзования других рейтингов менее процента

### ✔ Комментарий ревьюера в2

По второй части совета

```
unknown    6676
E          3921
T          2905
M          1536
E10+       1393
```

```
EC           8
```

```
K-A     3
AO      1
RP      1
```

стоит ответить на вопрос — нужны ли нам малочисленные категории? Все ли с ними в порядке, если их так мало?

## Поиск дубликатов

```
data.duplicated().sum()
```

```
0
```

Явных дубликатов нет

```
data.sort_values(by='name').head(15)
```

| | name | platform | year_of_release | genre | na_sales | eu_sales | jp_s |
|---|---|---|---|---|---|---|---|
| **14983** | Beyblade Burst | 3DS | 2016 | Role-Playing | 0.00 | 0.00 | |
| **1079** | Fire Emblem Fates | 3DS | 2015 | Role-Playing | 0.81 | 0.23 | |
| **3358** | Frozen: Olaf's Quest | 3DS | 2013 | Platform | 0.27 | 0.27 | |
| **3862** | Frozen: Olaf's Quest | DS | 2013 | Platform | 0.21 | 0.26 | |
| **13794** | Haikyu!! Cross Team Match! | 3DS | 2016 | Adventure | 0.00 | 0.00 | |
| **2454** | Tales of Xillia 2 | PS3 | 2012 | Role-Playing | 0.20 | 0.12 | |
| **4728** | '98 Koshien | PS | 1998 | Sports | 0.15 | 0.10 | |
| **8342** | .hack//G.U. Vol.1//Rebirth | PS2 | 2006 | Role-Playing | 0.00 | 0.00 | |
| **7087** | .hack//G.U. Vol.2//Reminisce | PS2 | 2006 | Role-Playing | 0.11 | 0.09 | |
| **8597** | .hack//G.U. Vol.2//Reminisce | PS2 | 2006 | Role-Playing | 0.00 | 0.00 | |

При выводе первых 15 строк отсортированных данных уже визуально видны дубли в 7087 и в 8597 строках. Такие строки стоит объединять.

Конечно тут нам повезло и мы увидели строки дубликаты и можем удалить их удалим вручную. Но как быть с теми строками, котрые мы не видим? Их может быть сотни и ручной способ не очень подходит.

Попробуем поискать подстроки в названиях

```
data.loc[data['name'].str.contains('sales')]
```

| | name | platform | year_of_release | genre | na_sales | eu_sales | jp_sales |
|---|---|---|---|---|---|---|---|
| **788** | Project Gotham Racing (JP weekly sales) | XB | 2002 | Action | 1.54 | 0.44 | 0.04 |
| **920** | Medal of Honor: European Assault (All Region s... | PS2 | 2005 | Shooter | 0.89 | 0.69 | 0.09 |
| **1123** | NBA Live 06 (All region sales) | PS2 | 2005 | Sports | 1.44 | 0.15 | 0.00 |
| **1136** | Tony Hawk's American Wasteland | PS2 | 2005 | Sports | 0.80 | 0.63 | 0.01 |

```
data.loc[data['name'].str.contains('National Geographic Panda') ]
```

| | name | platform | year_of_release | genre | na_sales | eu_sales | jp_sales |
|---|---|---|---|---|---|---|---|
| **13508** | National Geographic Panda (US | DS | 2008 | Simulation | 0.04 | 0.0 | 0.00 |

Подозреваю, что все строки содержащие слово sale - это дубли и их можно соединить с такими же строками, только не содержащими sale, но на данном этапе не вижу целесообразности перебирать все 142 строки и искать вручную им пару.

Поищем дубликаты среди записей в двух колонках: имя и платформа

```
data.duplicated(subset=['name','platform']).sum()
```

```
3
```

**Комментарий ревьюера**

Проверка на поиск дубликатов выполнена, молодец

Особенно это станет важным, когда мы перейдем к более сложным задачам на втором модуле курса

```
data.loc[data.duplicated(subset=['name','platform'])]
```

| | name | platform | year_of_release | genre | na_sales | eu_sales | jp_sales | other |
|---|---|---|---|---|---|---|---|---|
| **1591** | Need for Speed: Most Wanted | X360 | 2005 | Racing | 1.0 | 0.13 | 0.02 | |

Выведем дубли на экран и посмотрим, действительно это дубли?

```
display(data.query('name == "Need for Speed: Most Wanted" & platform == "X360"'))
display(data.query('name == "Need for Speed: Most Wanted" & platform == "PC"'))
display(data.query('name == "Madden NFL 13" & platform == "PS3"'))
```

| | name | platform | year_of_release | genre | na_sales | eu_sales | jp_sales | other_ |
|---|---|---|---|---|---|---|---|---|
| **1190** | Need for Speed: Most Wanted | X360 | 2012 | Racing | 0.62 | 0.78 | 0.01 | |
| **1591** | Need for Speed: Most Wanted | X360 | 2005 | Racing | 1.00 | 0.13 | 0.02 | |

| | name | platform | year_of_release | genre | na_sales | eu_sales | jp_sales | other_ |
|---|---|---|---|---|---|---|---|---|

Одна и та же игра была выпущена на одной и той же платформе в 2005 и в 2012 годах. Возможно ее нужно было переименовать? Добавить номер релиза? Пока оставим как есть.

А вот строку 16230 сложим с 604

```
#данные о продажах в Японии переместим в стоку 7087

#удалим строку 8597
try:
    data.loc[604,'eu_sales']+=data.loc[16230,'eu_sales']
    data.drop(axis=0, index=16230, inplace=True)
except:
    print('Строка с номером 8597 уже удалена')
display(data.query('name == "Madden NFL 13" & platform == "PS3"'))
```

| | name | platform | year_of_release | genre | na_sales | eu_sales | jp_sales | other_sa |
|---|---|---|---|---|---|---|---|---|
| **604** | Madden | PS3 | 2012 | Sports | 2.11 | 0.23 | 0.0 | |

## Суммарные продажи sum_sales

Посчитаем суммарные продажи во всех регионах и запишем их в отдельный столбец

```
data.head(10)
```

| | name | platform | year_of_release | genre | na_sales | eu_sales | jp_sales | oth |
|---|---|---|---|---|---|---|---|---|
| **0** | Wii Sports | Wii | 2006 | Sports | 41.36 | 28.96 | 3.77 | |
| **1** | Super Mario Bros. | NES | 1985 | Platform | 29.08 | 3.58 | 6.81 | |
| **2** | Mario Kart Wii | Wii | 2008 | Racing | 15.68 | 12.76 | 3.79 | |
| **3** | Wii Sports Resort | Wii | 2009 | Sports | 15.61 | 10.93 | 3.28 | |
| **4** | Pokemon Red/Pokemon Blue | GB | 1996 | Role-Playing | 11.27 | 8.89 | 10.22 | |
| **5** | Tetris | GB | 1989 | Puzzle | 23.20 | 2.26 | 4.22 | |
| **6** | New Super Mario Bros. | DS | 2006 | Platform | 11.28 | 9.14 | 6.50 | |

```
data[['na_sales','eu_sales','jp_sales', 'other_sales']].describe()
```

| | na_sales | eu_sales | jp_sales | other_sales |
|---|---|---|---|---|
| **count** | 16443.000000 | 16443.000000 | 16443.000000 | 16443.000000 |
| **mean** | 0.264028 | 0.145939 | 0.078492 | 0.047597 |
| **std** | 0.818400 | 0.506731 | 0.311109 | 0.188011 |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **25%** | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **50%** | 0.080000 | 0.020000 | 0.000000 | 0.010000 |
| **75%** | 0.240000 | 0.110000 | 0.040000 | 0.030000 |
| **max** | 41.360000 | 28.960000 | 10.220000 | 10.570000 |

Порядок данных похож, можно расчитывать суммарные продажи

```
data['sum_sales'] = data['na_sales'] + data['eu_sales']+ data['jp_sales']+data['other_sales']
data[['na_sales','eu_sales','jp_sales', 'other_sales', 'sum_sales']].head()
```

| | na_sales | eu_sales | jp_sales | other_sales | sum_sales |
|---|---|---|---|---|---|
| **0** | 41.36 | 28.96 | 3.77 | 8.45 | 82.54 |
| **1** | 29.08 | 3.58 | 6.81 | 0.77 | 40.24 |
| **2** | 15.68 | 12.76 | 3.79 | 3.29 | 35.52 |
| **3** | 15.61 | 10.93 | 3.28 | 2.95 | 32.77 |
| **4** | 11.27 | 8.89 | 10.22 | 1.00 | 31.38 |

### ❌ Комментарий ревьюера

Стоит добавить вывод по результатам выполнения раздела проекта

### ✍️ Комментарий Ирины 1

**Выводы:**

В ходе предобработки данных потребовалось осущебыли следующие операции:

- Названия столбцов таблицы данных были приведены к единообразному виду – к нижнему регистру
- Преобразованы типы данных:
  - Год выпуска преобразован к целочисленному типу
  - Оценнки пользователей 0-10 приведены к числовым значениям с запятой вместо строковых

    - Оценки критиков 0-100 приведены к целочисленным, т.к. нет данных со значениями после запятой
- Удалены данные с пропусками в годе выпуска. Всего удалено 2% данных
- Помечены пропуски в данных оценки пользователя и оценки критиков нулями. Оценки пользователей со строковыми данными tbd преобразованы в нули.
- В дальнейшем анализе данные с нулями в оценках не учитывались.
- Пропуски в названиях игр удалены (всего два пропуска)
- Доля пропусков в данных рейтинга 41 процент помечена значением NV – not value
- Явные дубликаты не обнаружены
- Обнаружены дубликаты с разделением данных по регионам на две строки, например: National Geographic Panda (JP sales) и National Geographic Panda (US sales). Т.к. дальнейшее исследование предполагает оценку суммарных продаж по всем регионам данные дубликаты не объеденены в одну строку. Строк с возможными дубликатами со словом sales обнаружено 142 строки.
- Обнаружены дубликаты среди записей name+platform. Только одна запись из найденных признана дублем.
- Добавлен столбец с суммарными продажами по всем регионам

## Исследовательский анализ данных

### Количество игр выпускаемых в разные годы

Посмотрим количество игр, выпускающихся в разные годы. Посмотрим, важны ли данные за все периоды

```
data.pivot_table(index='year_of_release', values='name', aggfunc='count').plot(
    kind = 'bar',
    y='name',
    xlabel='год выпуска',
    ylabel='количество игр',
    grid=True,
    figsize=(15,4),
    label='количество игр',
    title = 'Количество игр, выпускающихся в разные годы',
    legend=False
)
plt.show()
```


**✔ Комментарий ревьюера**

Классный график, все элементы добавлены, молодец

```
легенду можно убрать, это дублирование информации
```

**✍ Комментарий Ирины 1**

Убрала легенду

```
legend=False
```

По графику видно, что до 1995 года выпускалось крайне мало игр, данные по этим годам в дальнейшем можно точно не учитывать.

С 1995 года начинается наращивание выпуска игр и пик разнообразных игр приходится на 2008-2009 годы. Далее разнообразие выпускаемых игр снижается и в 2012-2016 достигает уровня как в 2001 году в промежутке от 400 до 600 в год.

## Лидирующие платформы по сумме продаж

- Посмотрим, как менялись продажи по платформам.
- Выберем платформы с наибольшими суммарными продажами
- построим распределение по годам.

Ответим на вопрос: За какой характерный срок появляются новые и исчезают старые платформы?

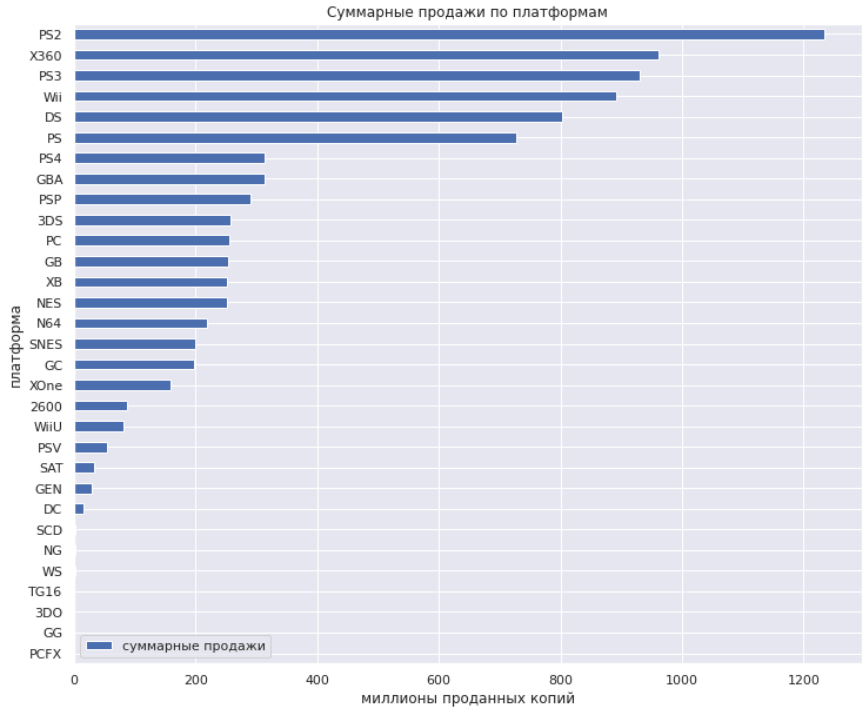
Выведем список всех платформ

```
print(data['platform'].unique())
```

```
['Wii' 'NES' 'GB' 'DS' 'X360' 'PS3' 'PS2' 'SNES' 'GBA' 'PS4' '3DS' 'N64'
 'PS' 'XB' 'PC' '2600' 'PSP' 'XOne' 'WiiU' 'GC' 'GEN' 'DC' 'PSV' 'SAT'
 'SCD' 'WS' 'NG' 'TG16' '3DO' 'GG' 'PCFX']
```

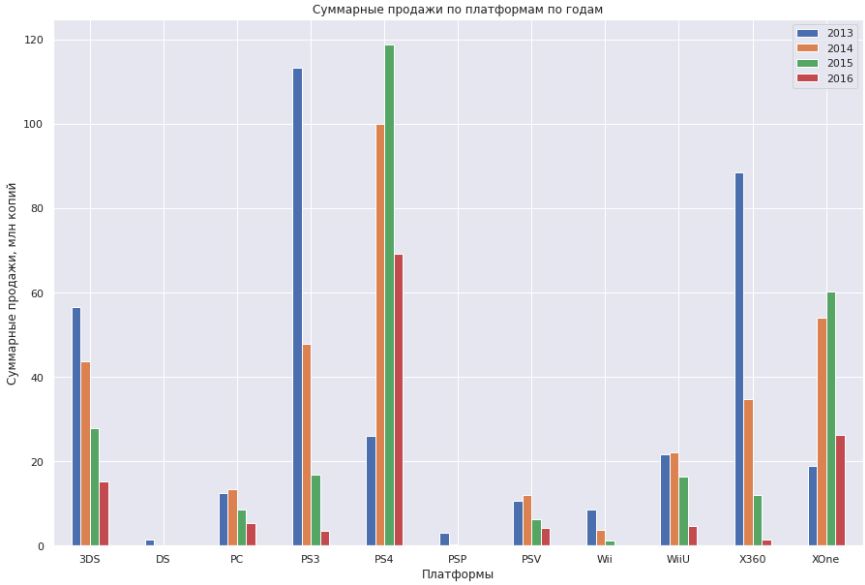
```
data.pivot_table(index='platform', values = 'sum_sales', aggfunc = 'sum').sort_values(by='sum_sales').plot(
    kind='barh',
    y='sum_sales',
    xlabel='платформа',
    grid=True,
    figsize=(12,10),
    label = 'суммарные продажи',
    title = 'Суммарные продажи по платформам'
)
plt.xlabel('миллионы проданных копий')
```

```
Text(0.5, 0, 'миллионы проданных копий')
```




```
data_year = pd.DataFrame()

def tempd(data, years):
    for y in years:
        data_temp=data.query('year_of_release == @y').pivot_table(index='platform', values='sum_sales', aggfunc = 'sum')
        data_year.insert(loc= len(data_year.columns) , column=y, value=data_temp['sum_sales'] )

years = [2013,2014,2015,2016]
tempd(data, years)
display(data_year)
data_year.reset_index().plot(x='platform', kind='bar', grid=True, figsize=(15,10), rot=0)
plt.title('Суммарные продажи по платформам по годам')
plt.ylabel('Суммарные продажи, млн копий')
plt.xlabel('Платформы')
plt.show()
```

| platform | 2013 | 2014 | 2015 | 2016 |
|---|---|---|---|---|
| **3DS** | 56.57 | 43.76 | 27.78 | 15.14 |
| **DS** | 1.54 | NaN | NaN | NaN |
| **PC** | 12.38 | 13.28 | 8.52 | 5.25 |
| **PS3** | 113.25 | 47.76 | 16.82 | 3.60 |
| **PS4** | 25.99 | 100.00 | 118.90 | 69.25 |
| **PSP** | 3.14 | 0.24 | 0.12 | NaN |
| **PSV** | 10.59 | 11.90 | 6.25 | 4.25 |
| **Wii** | 8.59 | 3.75 | 1.14 | 0.18 |
| **WiiU** | 21.65 | 22.03 | 16.35 | 4.60 |
| **X360** | 88.58 | 34.74 | 11.96 | 1.52 |
| **XOne** | 18.96 | 54.07 | 60.14 | 26.15 |




### ⚠️ Комментарий ревьюера

Стоит развернуть подписи на оси X и добавить название для оси У

Подписи осей на графиках и название добавят ясности и читабельности. Это важные элементы любой визуализации. Как добавить подписи и названия, смотри здесь

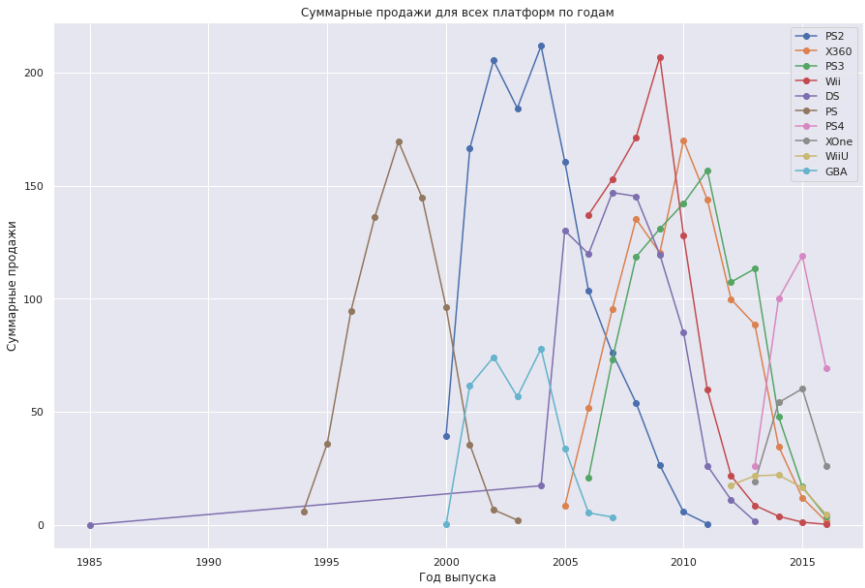
### ✍️ Комментарий Ирины 1

Сделано

Из графиков видно, что год от года суммарные продажи падают для всех платформ, но для некотроых видем сначала рост, а уже потом спад:PC, PS4, PSV, WiiU, XOne

Выберем платформы с наибольшими суммарными продажами. И посмотрим как менялись продажи по платформам по годам.

```
#создаем объект рисунка
fig,ax = plt.subplots()
#выбираем платформы
plat = ['PS2', 'X360', 'PS3', 'Wii', 'DS', 'PS', 'PS4', 'XOne', 'WiiU', 'GBA']
for pl in plat:
    data.loc[ data['platform']==pl ].pivot_table(index='year_of_release', values='sum_sales', aggfunc='sum').plot(
        y='sum_sales',
        style='o-',
        label=pl,
        ax=ax,
        figsize=(15,10)
    )
    plt.ylabel('Суммарные продажи')
    plt.xlabel('Год выпуска')
plt.title('Суммарные продажи для всех платформ по годам')
plt.show()
```




Выводы: лидирующие платформы по сумме продаж за все годы: PS2, X360, PS3, Wii, но эти платформы к 2016 году практически не имеют продаж. Существенные продажи в 2014-2016гг у платформ: PS4, XOne, WiiU, X360, PS3

## Определение актуального периода для исследования. Период жизни игровой платформы

Старые платформы:

- платформа PS с 1994 до 2003
- платформа GBA с 2000 по 2007
- платформа PS2 с 2000 по 2011
- платформа DS с 2004-2013

Относительно новые:

- платформа PS3 с 2006-2016
- платформа X360 с 2005-2016
- платформа Wii с 2006-2016

Новые платформы:

- платформа PS4 с 2013-2016
- платформа WiiU с 2012-2016
- платформа XOne с 2013-2016

Старые платформы пропадают к 2005 году по большей части. Среди них выделяется платформа PS2, у которой продажи спадают к 2010 году и DS у которой продажи до 2013 года. Срок жизни 8 - 10(прям уже много) лет

Относительно новые платформы с 2005 по 2016 года. Срок жизни 10 и еще не совсем окончен.

Самые новые платформы выделяются PS4, WiiU, Xone, возникшие совсем недавно в 2012-2013 годах. Срок их жизни 4 года до н.в.

Выберем за актуальный период данные с 2008 года по 2016, когда все старые платформы отходят на второй план и новые набирают силу. Если вернуться к графику количество игр в год, то было бы логично взять за актуальный период с 2012 по 2016, где число выпускаемых игр принимает стабильное число между 400-600. Возъмем именно такой период

Актуальный период 2012-2016 года.

**✔ Комментарий ревьюера**

Отличное определение жизненного цикла продаж у платформ, молодец

**✘ Комментарий ревьюера**

Странный выброс 1985 года у DS, можно посмотреть когда платформу выпустили на рынок, стоит удалить аномалию

**✍ Комментарий Ирины 1**

Кстати, меня тоже смутил этот выброс вначале, но почему-то решила, что он мне не мешает и не стала его удалять. Посмотрим на данные соответствующие странному выбросу для платформы DS для года 1985 и удалим их. По данным из сети платформа DS возникла в 2004 году, т.к. в 1985 году не могло быть никаких продаж.

```
data.loc[ (data['platform']=='DS') & (data['year_of_release']==1985) ]
```

| | name | platform | year_of_release | genre | na_sales | eu_sales | jp_sales | othe |
|---|---|---|---|---|---|---|---|---|
| | Strongest | | | | | | | |

```
data=data.drop(labels = data.loc[ (data['platform']=='DS') & (data['year_of_release']==1985) ].index, axis=0)
data.loc[15955:15959]
```

| | name | platform | year_of_release | genre | na_sales | eu_sales | jp_sales | ot |
|---|---|---|---|---|---|---|---|---|
| **15955** | Kiss Bell | PSV | 2014 | Adventure | 0.00 | 0.0 | 0.02 | |
| **15956** | San Goku Shi DS 2 | DS | 2007 | Strategy | 0.00 | 0.0 | 0.02 | |
| | Shin | | | | | | | |

```
#далее будем работать с масивом данных за выбранный период
data_p=data.query('(year_of_release>=2012)&(year_of_release<=2016)')
```

**✘ 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

```
data_p=data.query('(year_of_release>=2012)&(year_of_release<=2016)')
```

**✍ Комментарий Ирины 1**

Возъмем в качестве актуального периода для исследования возьмет 2015 и 2016 годы, т.к. в 2014 еще велико влияние отживающих платформ, таких как PS3 и X360

**⚠ Комментарий ревьюера в2**

за 2016 год данные неполные, фактически у нас для анализа взят только один полный год — 2015, на рабочих проектах можем упустить фактор сезонности

```
#далее будем работать с масивом данных за выбранный период
data_p=data.query('(year_of_release>=2015)&(year_of_release<=2016)')
```

## Определение лидирующих по продажам платформ

Лидируют по максимальным продажам в период 2012-2016 платформы за счет бесцеллеров:

- X360
- PS3
- Wii

У всех платформ начиная с определенного момента продажи начинают падать.

Но на 2013-2016 год продажи еще сохраняются у следующих платформ:

- PS4
- PS3
- X360
- WiiU
- Xone

Выберем именно эти платформы как потенциально прибыльные: PS4, PS3, X360, XOne, WiiU, Wii

### ❌ Комментарий ревьюера

X360, Wii и PS3 уже исполнилось 11 лет и они с малой вероятностью станут перспективными или сохранят продажи в 2017 году

### ✍ Комментарий Ирины 1

Выберем в качестве лидирующих по продажам новые платформы:

- платформа PS4 с 2013-2016
- платформа WiiU с 2012-2016
- платформа XOne с 2013-2016

### ❌ Комментарий ревьюера в2

Стоит дополнить перечень перспективных платформ

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

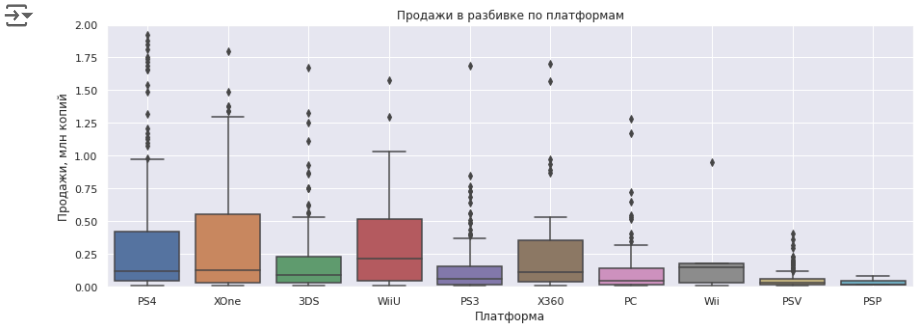
30-тилетняя история персональных компьютеров говорит, что игры для PC можно включить в рекомендацию

```
#далее будем работать с выбранными платформами
data_platform = data_p.query('platform == "PS4"|platform == "XOne"|platform == "WiiU"')
```

## Диаграмма размаха продаж в разбивке по платформам

Построим диаграммы размаха для всех данных для периуда 2012-2016 год

```
plt.figure(figsize=(15,5))
plt.ylim(0, 2)
sns.boxplot(data = data_p, x='platform', y='sum_sales')
plt.title('Продажи в разбивке по платформам')
plt.ylabel('Продажи, млн копий')
plt.xlabel('Платформа')
plt.grid('True')
plt.show()
```



По диаграмме размаха для периода 2012-2016 наиболее прибыльные платформы: X360, XOne, WiiU, PS3, PS4,Wii. У них максимальная медиана из всех.

### ⚠ 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

### ✍ Комментарий Ирины 1

Выведем диаграммы размаха в 100% масштабе

```
plt.figure(figsize=(15,5))
sns.boxplot(data = data_p, x='platform', y='sum_sales')
plt.title('Продажи в разбивке по платформам')
plt.ylabel('Продажи, млн копий')
plt.xlabel('Платформа')
plt.grid('True')
plt.show()
```


### ✔ 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...

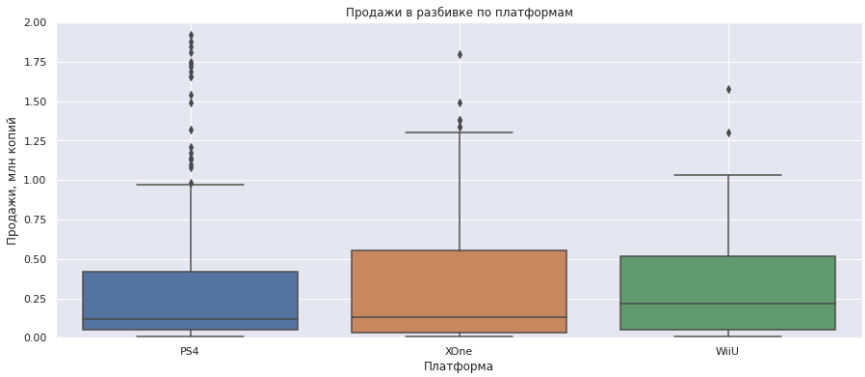
Две хорошие статьи про диаграмму размаха

Ящики, усы и скрипки

Исследуем отношение между переменными

Построим ящик с усами для потенциально прибыльных платформ для периода 2015-2016

```
plt.figure(figsize=(15,6))
plt.ylim(0,2)
sns.boxplot(data = data_platform, x='platform', y='sum_sales')
plt.title('Продажи в разбивке по платформам')
plt.ylabel('Продажи, млн копий')
plt.xlabel('Платформа')
plt.grid('True')
plt.show()
```




Выводы: для актуального периода 2015-2016 гг. для всех платформ 75% данных суммарных продаж составляет до 0,6 млн проданных копий. Медианное значение для наиболее прибыльных платформ находится в районе 0,1 млн копий. Самые экстремальные выбросы уникально-большие продажи у платформ PS4 более 14млн проданных копий

## Влияние отзывов пользователей и критиков на продажи для одной популярой платформы

Выберем для анализа платформу PS4. Возьмем актуальный период 2015-2016

```
data_ps4 = data_platform.query('platform=="PS4"')
data_ps4
```

| | name | platform | year_of_release | genre | na_sales | eu_sales | jp_sale |
|---|---|---|---|---|---|---|---|
| **31** | Call of Duty: Black Ops 3 | PS4 | 2015 | Shooter | 6.03 | 5.86 | 0.3 |
| **77** | FIFA 16 | PS4 | 2015 | Sports | 1.12 | 6.12 | 0.( |
| **87** | Star Wars Battlefront (2015) | PS4 | 2015 | Shooter | 2.99 | 3.49 | 0.2 |
| **94** | FIFA 17 | PS4 | 2016 | Sports | 0.66 | 5.75 | 0.( |
| **105** | Fallout 4 | PS4 | 2015 | Role-Playing | 2.53 | 3.27 | 0.2 |
| **...** | ... | ... | ... | ... | ... | ... | |
| **16500** | Root Letter | PS4 | 2016 | Adventure | 0.00 | 0.00 | 0.( |
| **16503** | Shin Hayarigami 2 | PS4 | 2016 | Adventure | 0.00 | 0.00 | 0.( |
| **16526** | Dungeons 2 | PS4 | 2016 | Role- | 0.01 | 0.00 | 0.( |

```
data_ps4.plot(
    style='o',
    x='user_score',
    y='sum_sales',
    figsize=(15,3),
    grid=True,
    alpha=0.3,
    legend=False

)
plt.title('Диаграмма рассеяния. Влияние отзывов пользователей на суммарные продажи')
plt.xlabel('Оценка пользователей')
plt.ylabel('Продажи')
plt.xlim(1,10)
plt.ylim(-0.3,3)
plt.show()
```


Для рассчета корреляции не будем учитывать оценки пользователей 0, т.к. пропуски в данных были заполнены нулями и они создают неестественный пик в данных.

```
print(data_ps4.query('user_score != 0')['sum_sales'].corr(data_ps4['user_score']))
```

```
-0.05973845712638215
```

Больше всего продаж для игр с отзывами пользователй от 5 до 8 балов. При этом рассеяние равномерное, линейная зависимость не наблюдается. Коэффициент линейной корреляции меньше 6%. Линейная корреляция отсутствует.

```
data_ps4.plot(
    style='o',
    x='critic_score',
    y='sum_sales',
    figsize=(15,3),
    alpha=0.3,
    grid=True
)
plt.title('Диаграмма рассеяния. Влияние отзывов критиков на суммарные продажи')
plt.xlabel('Оценка критиков')
plt.ylabel('Суммарные продажи')
plt.xlim(1,100)
plt.ylim(-0.3,4)
plt.show()
```


Для рассчета корреляции с оценкой критиков поступим так же, как и в предыдущем случае. Уберем из анализа нулевые значения оценки, т.к. нулями мы заполнили пропуски в данных.

```
print(data_ps4.query('critic_score!=0')['sum_sales'].corr(data_ps4['critic_score']))
```

```
0.3928485555130601
```

Больше всего продаж для игр с отзывами критиков от 60 до 85 балов. Суммы продаж растут с ростом оценки критиков. При этом наблюдается тенденция к линейной зависимости. Коэффициент корреляции уже 39%. Средняя корреляция. Явно оценки критиков имеют большее значение для людей, чем оценки пользователей.

Выводы:

- сумма продаж линейно не коррелирует с оценкой пользоватлей на популярой игровой платформе. Коэффициент менее 6%
- Отзывам критиков явно доверяют больше, наблюдается средняя корреляция суммарных продаж с отзывами критиков - 39%.

## Влияние отзывов на продажи для остальных потенциально прибыльных платформ

```
#анализируем успешные платформы за период 2015-2016 'XOne', 'WiiU'

list_data_corr = {
    'platform': [],
    'user_corr': [],
    'critic_corr': []
}

# рассчитаем  коэффициенты корреляции для других потенциально прибыльных платформ
# не забываем не учитывать нулевые значения, которые были подставлены вместо пропусков
for pl in data_platform['platform'].unique():
    list_data_corr['platform'].append(pl)

    u_corr=data_platform.query('platform==@pl & user_score!=0')['sum_sales'].corr(data_platform['user_score'])
    list_data_corr['user_corr'].append(round(u_corr,2))

    c_corr=data_platform.query('platform==@pl & critic_score!=0')['sum_sales'].corr(data_platform['critic_score'])
    list_data_corr['critic_corr'].append(round(c_corr,2))

data_corr = pd.DataFrame(list_data_corr)
display(data_corr)
```

| | platform | user_corr | critic_corr |
|---|---|---|---|
| **0** | PS4 | -0.06 | 0.39 |
| **1** | XOne | -0.04 | 0.43 |
| **2** | WiiU | 0.36 | 0.32 |

**⚠ Комментарий ревьюера**

на проектах стоит снижать размерность выводимой информации, где не требуется максимальная точность, до одного или двух знаков после запятой

```
0    PS3    -0.006218    0.334166
```

**✍ Комментарий Ирины 1**

Округление произведено

Выводы:

- для платформ: XOne, PS4 наблюдается средняя линейная корреляция в районе 39%-43% суммарных продаж от оценки критиков и отсутствие линейной корреляции с отзывами пользоватлей.
- для платформы WiiU суммарные продажи средне-коррелируют как с отзывами критиков: 32%, так и с отзывами пользователей 36%. Чем-то эта платформа отличается от остальных. Возможно такой факт влияет на ее отличительные данные по этой консоли: "Одной из целей WiiU было привлечь более серьёзную игровую аудиторию"

**✔ Комментарий ревьюера**

Сравнение показателей на нескольких платформах, позволяет набрать вес твоему исследованию, молодец

**⚠ Комментарий ревьюера**

Стоит добавить парочку диаграмм рассеяния, коэф. корреляции отображает лишь возможное наличие линейной связи

**✍ Комментарий Ирины 1**

Парочка диаграмм рассеяния была выше, или я что-то путаю? Диаграммы были для x360, теперь переделаны для ps4

**⚠ Комментарий ревьюера в2**

Стоит добавить ЕЩЕ парочку диаграмм рассеяния, коэф. корреляции отображает лишь возможное наличие линейной связи

**⚠ 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**

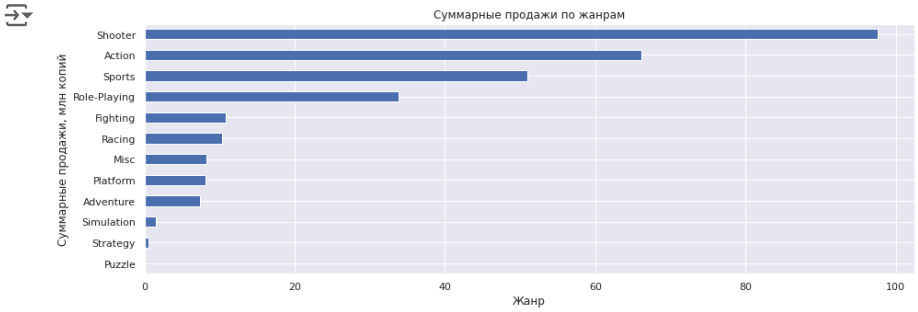
Приведу пример ложной корреляции, весьма известный в статистической литературе. Была исследована корреляционная связь между числом аистов, свивших гнезда в южных районах Швеции, и рождаемостью в эти же годы в Швеции. Расчёты, выполненные ради шутки, показали существенную положительную корреляцию между 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) .

## Распределение игр по жанрам. Определение самых прибыльных жанров

```
#анализируем успешные платформы за период 2012-2016 'X360', 'XOne', 'WiiU', 'PS3', 'PS4', 'Wii'
data_platform.pivot_table(index='genre', values='sum_sales', aggfunc='sum').\
sort_values(by='sum_sales',ascending = True).\
plot(
    y='sum_sales',
    kind='barh',
    figsize=(15,5),
    legend=False
)
plt.ylabel('Суммарные продажи, млн копий')
plt.xlabel('Жанр')
plt.title('Суммарные продажи по жанрам')
plt.show()
```




### ⚠ Комментарий ревьюера

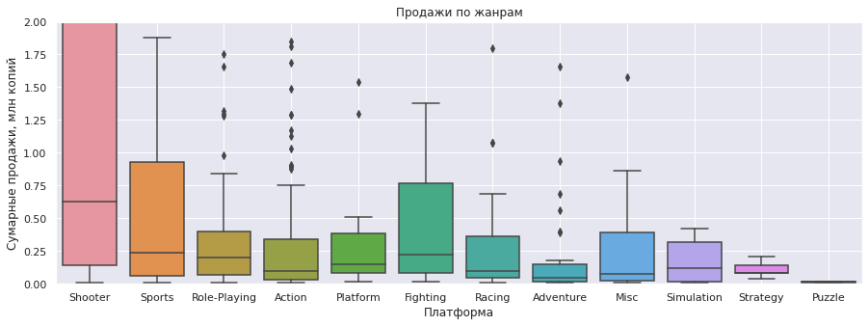
Стоит сортировать значения для графиков и развернуть график в горизонтальные бары, если надписи оси X не умещаются в горизонтальном виде

### ✍ Комментарий Ирины 1

Данные уже были отсортированы. А график я перевернула

👍

```
plt.figure(figsize=(15,5))
plt.ylim(0, 2)
sns.boxplot(data = data_platform, x='genre', y='sum_sales')
plt.title('Продажи по жанрам')
plt.ylabel('Сумарные продажи, млн копий')
plt.xlabel('Платформа')
plt.grid('True')
plt.show()
```




Вывод:

- Самые большие продажи среди жанров: Action, Shooter, Sports, Role-Playing
- Причем максимум Action добивается за счет продаж некоторых бесцеллеров, если эти бесцеллеры не учитывать, то Shooter выходит на первое место

### ❌ Комментарий ревьюера

Стоит добавить вывод по результатам выполнения раздела проекта

### ✍ Комментарий Ирины 1

Вывод:

- Количество игр выпускаемых в разные годы:

до 1995 года выпускалось крайне мало игр, данные по этим годам в дальнейшем можно точно не учитывать. С 1995 года начинается наращивание выпуска игр и пик разнообразных игр приходится на 2008-2009 годы. Далее разнообразие выпускаемых игр снижается и в 2012-2016 достигает уровня как в 2001 году в промежутке от 400 до 600 в год.

- Лидирующие платформы по сумме продаж:

лидирующие платформы по сумме продаж за все годы: PS2, X360, PS3, Wii, но эти платформы к 2016 году практически или не имеют продаж совсем или сильно снижают. Существенные продажи в 2014-2016гг у платформ: PS4, XOne, WiiU, X360, PS3, Wii

- Определение актуального периода исследования:

За актуальный период выбран период 2015-2016г когда имеют продажи новые платформы. В 2014 году еще существенное влияние оказывают старые платформы.

Определен средний срок жизни игровых платформ:10-11 лет, что свидетельствует о завершении срока жизни платформ X360, PS3, Wii к 2017 году

- Определение наиболее прибыльных платформ

Выделены самые новые платформы PS4, WiiU, Xone, возникшие в 2012-2013 годах. Срок их жизни 4 года до н.в., следовательно есть еще куда им развиваться.

Для актуального периода 2015-2016 гг. для всех платформ 75% данных суммарных продаж составляет до 0,6 млн проданных копий. Медианное значение для наиболее прибыльных платформ находится в районе 0,1 млн копий.

Самые экстремальные выбросы уникально-большие продажи у платформ PS4 более 14млн проданных копий.

- Исследование влияния отзывов пользователей и критиков для платформы PS4:

сумма продаж линейно не коррелирует с оценкой пользоватлей на популярой игровой платформе. Коэффициент менее 6%. Отзывам критиков явно доверяют больше, наблюдается средняя корреляция суммарных продаж с отзывами критиков - 39%.

- Исследование влияния отзывов для других потенциально прибыльных платформ: X360, PS4, WiiU

для платформ: XOne, PS4 наблюдается средняя линейная корреляция в районе 39%-43% суммарных продаж от оценки критиков и отсутствие линейной корреляции с отзывами пользоватлей. для платформы WiiU суммарные продажи средне-коррелируют как с отзывами критиков: 32%, так и с отзывами пользователей 36%. Чем-то эта платформа отличается от остальных. Возможно такой факт влияет на ее отличительные данные: "Одной из целей WiiU было привлечь более серьёзную игровую аудиторию"

## Портрет пользователя каждого региона

## Топ-5 популярных жанров для каждого региона

Сделаем разбивку данных по регионам

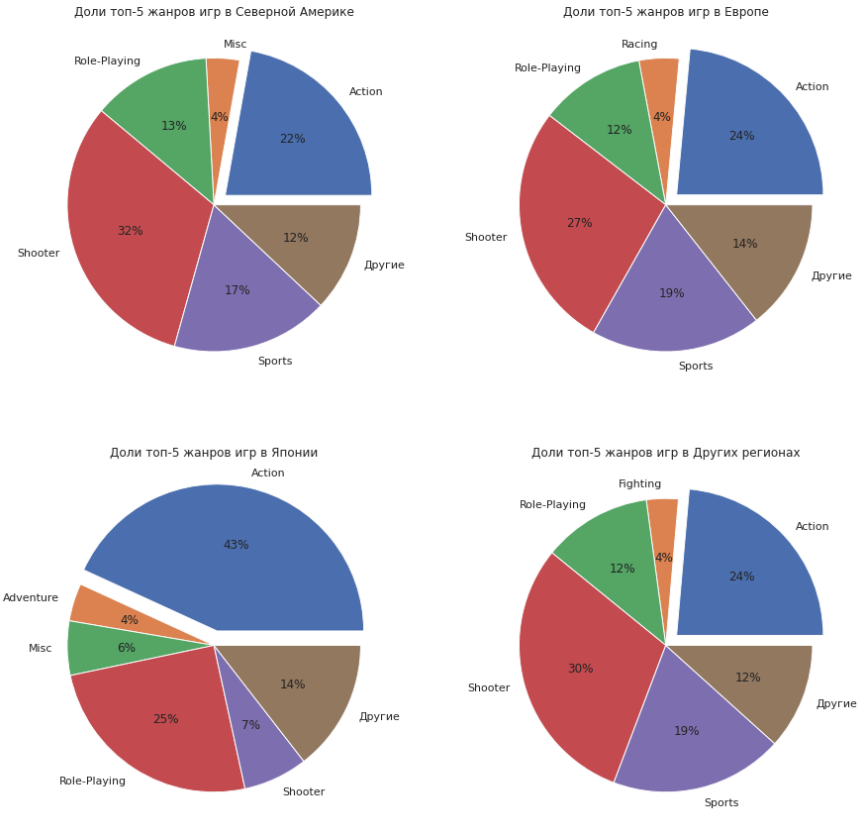
- NA_sales — продажи в Северной Америке (миллионы проданных копий)
- EU_sales — продажи в Европе (миллионы проданных копий)
- JP_sales — продажи в Японии (миллионы проданных копий)

Для анализа будем использовать данные для актуального периода. Платформы же оставим все, чтобы отобразить всю картину целиком

```
# сводные таблицы по жанрам для трех регионов
data_na_genre = (data_p
    .pivot_table(index='genre', values='na_sales', aggfunc='sum')
    .sort_values(by='na_sales',ascending = False).reset_index())
data_eu_genre = (data_p
    .pivot_table(index='genre', values='eu_sales', aggfunc='sum')
    .sort_values(by='eu_sales',ascending = False).reset_index())
data_jp_genre = (data_p
    .pivot_table(index='genre', values='jp_sales', aggfunc='sum')
    .sort_values(by='jp_sales',ascending = False).reset_index())
data_other_genre = (data_p
    .pivot_table(index='genre', values='other_sales', aggfunc='sum')
    .sort_values(by='other_sales',ascending = False).reset_index())
#сумма продаж для всех жанров не входящих в топ-5
others_na_sales=data_na_genre.loc[5:].sum()
others_eu_sales=data_eu_genre.loc[5:].sum()
others_jp_sales=data_jp_genre.loc[5:].sum()
others_other_sales=data_other_genre.loc[5:].sum()

#оставляем только топ5 значений, остальные удаляем
top5_na = data_na_genre.drop(data_na_genre.index[5:len(data_na_genre)], axis = 0)
top5_eu = data_eu_genre.drop(data_eu_genre.index[5:len(data_eu_genre)], axis = 0)
top5_jp = data_jp_genre.drop(data_jp_genre.index[5:len(data_jp_genre)], axis = 0)
top5_other = data_other_genre.drop(data_other_genre.index[5:len(data_other_genre)], axis = 0)

#добавляем запись о сумме продаж остальных жанров объеденив, как "Другие"
top5_na.loc[ len(top5_na.index )] = ['Другие', others_na_sales['na_sales']]
top5_eu.loc[ len(top5_eu.index )] = ['Другие', others_eu_sales['eu_sales']]
top5_jp.loc[ len(top5_jp.index )] = ['Другие', others_jp_sales['jp_sales']]
top5_other.loc[ len(top5_other.index )] = ['Другие', others_other_sales['other_sales']]
```

```
f,ax = plt.subplots(2,2)
f.set_size_inches(15, 15)
plt.suptitle('Топ-5 жанров по регионам', fontsize=20)
explode = (0.1, 0, 0, 0, 0, 0)

top5_na.groupby(['genre']).sum().plot(
    kind='pie',
    y='na_sales',
    autopct='%1.0f%%',
    title='Доли топ-5 жанров игр в Северной Америке',
    ylabel='',
    ax=ax[0,0],
    legend=False,
    explode = explode
)
top5_eu.groupby(['genre']).sum().plot(
    kind='pie',
    y='eu_sales',
    autopct='%1.0f%%',
    title='Доли топ-5 жанров игр в Европе',
    ylabel='',
    ax=ax[0,1],
    legend=False,
    explode = explode
)
top5_jp.groupby(['genre']).sum().plot(
    kind='pie',
    y='jp_sales',
    autopct='%1.0f%%',
    title='Доли топ-5 жанров игр в Японии',
    ylabel='',
    legend=False,
    ax=ax[1,0],
    explode = explode
)
top5_other.groupby(['genre']).sum().plot(
    kind='pie',
    y='other_sales',
    autopct='%1.0f%%',
    title='Доли топ-5 жанров игр в Других регионах',
    ylabel='',
    legend=False,
    ax=ax[1,1],
    explode = explode
)

plt.show()
```

Топ-5 жанров по регионам




Северная Америка топ-5 жанров:

- Shooter - 32%
- Action - 22%
- Sports - 17%
- Role-Playing - 13%
- Misc - 4%
- Другие - 12%

Европа топ-5 жанров:

- Shooter - 27%
- Action - 24%
- Sports - 19%
- Role-Playing - 12%
- Racing - 4%
- Другие - 14%

Япония топ-5 жанров:

- Action - 43%
- Role-Playing - 25%
- Shooter - 13%
- Adventure - 4%
- Misc - 6%
- Другие - 14%

Выводы: популярные жанры игр в Америке и в Европе схожи, но отличаются от Японии, что еще раз доказывает различие в менталитете с восточными народами.

В Северной Америке и в Европе самые популярные игровые жанры Shooter. Но в Америке популярность этого жанра несколько выше (32%), чем в Европе (27%). На втором месте жанр Action, доля игр с этим жанром схода - 22-24%. На третьем месте жанр Sports - 17%-19%, так же примерно одинаковой долей. Role-Playing занимает четвертое место: 12-13%. Доля других игр около 14%

В Японии наблюдается совсем иная картина. На первое место выходят игры с жанром Action и занимают без малого почти половину рынка. А вот популярный на западе жанр Shooter спускается н третье место и составляет всего 13%. Второе место занимает жанр Role-Playing - 25%, примерно столькоже, сколько Shooter в Европе, тогда как на западе этот жанр лиш на четвертом месте.

## Топ-5 популярных платформ для каждого региона

Сделаем разбивку данных по регионам

- NA_sales — продажи в Северной Америке (миллионы проданных копий)
- EU_sales — продажи в Европе (миллионы проданных копий)
- JP_sales — продажи в Японии (миллионы проданных копий)

```
data_p.platform.unique()
```

```
array(['PS4', 'XOne', '3DS', 'WiiU', 'PS3', 'X360', 'PC', 'Wii', 'PSV',
       'PSP'], dtype=object)
```

### ❌ Комментарий ревьюера

Срезая часть платформ,

мы не узнаем реальную долю рынка, которую занимают остальные платформы, т.к. мы не будем знать 100% объем продаж

При создании актуальной выборки стоит удалить только старые игры, не изменяя перечень платформ, как например в data_p

**Стоит выполнить раздел с обновленной актуальной выборкой, выводы можно подправить**

### ✍️ Комментарий Ирины 1

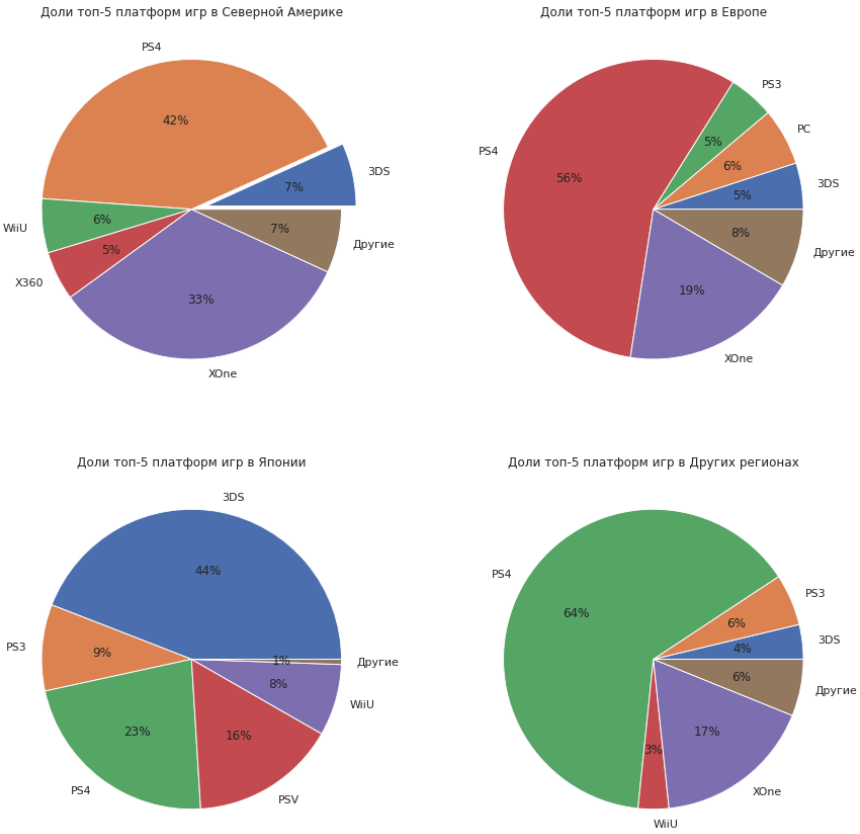
Исправила, вернула платформы на место.

### ✔️ Комментарий ревьюера в2

Отлично, молодец

```
# сводные таблицы по платформам для трех регионов
data_na_platform = (data_p
    .pivot_table(index='platform', values='na_sales', aggfunc='sum')
    .sort_values(by='na_sales',ascending = False).reset_index())
data_eu_platform = (data_p
    .pivot_table(index='platform', values='eu_sales', aggfunc='sum')
    .sort_values(by='eu_sales',ascending = False).reset_index())
data_jp_platform = (data_p
    .pivot_table(index='platform', values='jp_sales', aggfunc='sum')
    .sort_values(by='jp_sales',ascending = False).reset_index())
data_other_platform = (data_p
    .pivot_table(index='platform', values='other_sales', aggfunc='sum')
    .sort_values(by='other_sales',ascending = False).reset_index())

#сумма продаж для всех платформ не входящих в топ-5
others_na_sales_p=data_na_platform.loc[5:].sum()
others_eu_sales_p=data_eu_platform.loc[5:].sum()
others_jp_sales_p=data_jp_platform.loc[5:].sum()
others_other_sales_p=data_other_platform.loc[5:].sum()

#оставляем только топ5 значений, остальные удаляем
top5_na_p = data_na_platform.drop(data_na_platform.index[5:len(data_na_platform)], axis = 0)
top5_eu_p = data_eu_platform.drop(data_eu_platform.index[5:len(data_eu_platform)], axis = 0)
top5_jp_p = data_jp_platform.drop(data_jp_platform.index[5:len(data_jp_platform)], axis = 0)
top5_other_p = data_other_platform.drop(data_other_platform.index[5:len(data_other_platform)], axis = 0)

#добавляем запись о сумме продаж остальных платформ объеденив, как "Другие"
top5_na_p.loc[ len(top5_na_p.index )] = ['Другие', others_na_sales_p['na_sales']]
top5_eu_p.loc[ len(top5_eu_p.index )] = ['Другие', others_eu_sales_p['eu_sales']]
top5_jp_p.loc[ len(top5_jp_p.index )] = ['Другие', others_jp_sales_p['jp_sales']]
top5_other_p.loc[ len(top5_other_p.index )] = ['Другие', others_other_sales_p['other_sales']]


f,ax = plt.subplots(2,2)
f.set_size_inches(15, 15)
plt.suptitle('Топ-5 платформ по регионам', fontsize=20)

top5_na_p.groupby(['platform']).sum().plot(
    kind='pie',
    y='na_sales',
    autopct='%1.0f%%',
    ax=ax[0,0],
    ylabel='',
    title='Доли топ-5 платформ игр в Северной Америке',
    legend=False,
    explode=explode
)

top5_eu_p.groupby(['platform']).sum().plot(
    kind='pie',
    y='eu_sales',
    autopct='%1.0f%%',
    ax=ax[0,1],
    ylabel='',
    title='Доли топ-5 платформ игр в Европе',
    legend=False
)
top5_jp_p.groupby(['platform']).sum().plot(
    kind='pie',
    y='jp_sales',
    autopct='%1.0f%%',
    ylabel='',
    title='Доли топ-5 платформ игр в Японии',
    legend=False,
    ax=ax[1,0]
)
top5_other_p.groupby(['platform']).sum().plot(
    kind='pie',
    y='other_sales',
    autopct='%1.0f%%',
    ylabel='',
    title='Доли топ-5 платформ игр в Других регионах',
    legend=False,
    ax=ax[1,1]
)
plt.show()
```

Топ-5 платформ по регионам




### ⚠️ Комментарий ревьюера

Стоит убрать легенду, она дублирует информацию

### ✍️ Комментарий Ирины 1

Далее исправления процентов и выводов, после возврата срезанных платформ

Северная Америка топ-5 платформ:

- PS4 - 42%
- XOne - 33%
- 3DS - 7%
- WiiU - 6%
- X360 - 5%
- Другие - 7%

Европа топ-5 платформ:

- PS4 - 56%
- XOne - 19%
- PC - 6%
- PS3 - 5%
- 3DS - 5%
- Другие - 8%

Япония топ-5 платформ:

- 3DS - 44%
- PS4 - 23%
- PSV - 16%
- PS3 - 9%
- WiiU - 8%
- Другие - 1%

Выводы:

В Северной Америке пользователи отдают предпочтения 2 платформам: PS4, XOne, причем PS4 занимает лидирующее место 42%, а XOne - 33%. Остальные платформы составляют незначительную долю (5-7%)

В Вероре с существенным отрывом лидирует платформа PS4 - 56%, больше половины всех платформ. На втором месте так же как и в Америке XOne, до ее доля 19%. Остальные платформы так же занимают незначтительный процент 5-8%

В Японии картина совершенно другая. На первом месте платформа 3DS- 44%, на втором PS4 - 23%, на третьем PS3- 16%. Остальные платформи 8-9%

## Влияние возрастного рейтинга на продажи в разных регионах

Сделаем разбивку данных по регионам

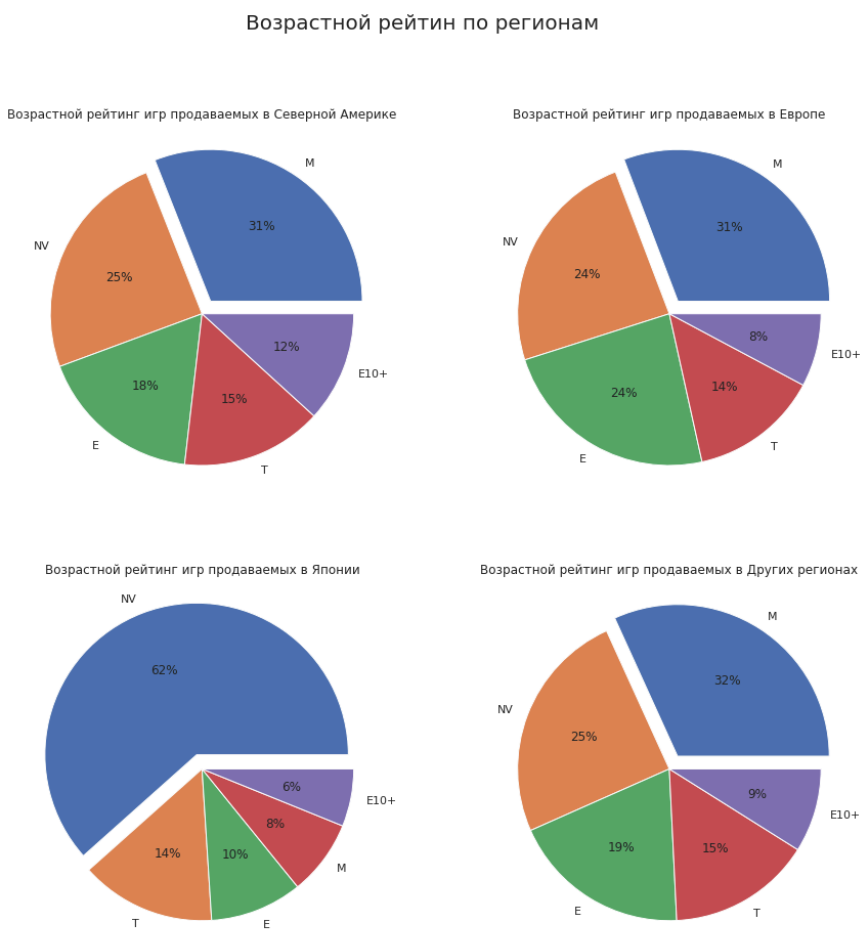
- NA_sales — продажи в Северной Америке (миллионы проданных копий)
- EU_sales — продажи в Европе (миллионы проданных копий)
- JP_sales — продажи в Японии (миллионы проданных копий)

Возрастной рейтинг:

- EC - для детей от 3 лет
- E - для всех возрастных категорий
- M - для лиц старше 17 лет
- T - для лиц старше 13 лет
- E10+ - для лиц старше 10 лет
- AO - для взрослых старше 18 лет
- RP - категория не присвоена
- K-A - для детей (был переименован в E)
- NV - тип не определен

```
# сводные таблицы по платформам для трех регионов с данными без рейтинга
data_na_rating = data_p.\
    pivot_table(index='rating', values='na_sales', aggfunc='sum').\
    sort_values(by='na_sales',ascending = False)
data_eu_rating = data_p.\
    pivot_table(index='rating', values='eu_sales', aggfunc='sum').\
    sort_values(by='eu_sales',ascending = False)
data_jp_rating = data_p.\
    pivot_table(index='rating', values='jp_sales', aggfunc='sum').\
    sort_values(by='jp_sales',ascending = False)
data_other_rating = data_p.\
    pivot_table(index='rating', values='other_sales', aggfunc='sum').\
    sort_values(by='other_sales',ascending = False)
```

```
f,ax = plt.subplots(2,2)
f.set_size_inches(15, 15)
plt.suptitle('Возрастной рейтин по регионам', fontsize= 20)
explode = (0.1, 0, 0, 0, 0)

data_na_rating.plot(
    kind='pie',
    y='na_sales',
    autopct='%1.0f%%',
    ax=ax[0,0],
    title='Возрастной рейтинг игр продаваемых в Северной Америке',
    legend=False,
    ylabel='',
    explode=explode
)
data_eu_rating.plot(
    kind='pie',
    y='eu_sales',
    autopct='%1.0f%%',
    ax=ax[0,1],
    title='Возрастной рейтинг игр продаваемых в Европе',
    legend=False,
    ylabel='',
    explode=explode

)
data_jp_rating.plot(
    kind='pie',
    y='jp_sales',
    autopct='%1.0f%%',
    title='Возрастной рейтинг игр продаваемых в Японии',
    legend=False,
    ylabel='',
    ax=ax[1,0],
    explode=explode
)
data_other_rating.plot(
    kind='pie',
    y='other_sales',
    autopct='%1.0f%%',
    title='Возрастной рейтинг игр продаваемых в Других регионах',
    legend=False,
    ylabel='',
    ax=ax[1,1],
    explode=explode
)
plt.show()
```




Северная Америка:

- продажи без рейтинга - 25%

Европа:

- продажи без рейтинга - 24%

Япония:

- продажи без рейтинга - 62%

### ✔ Комментарий ревьюера

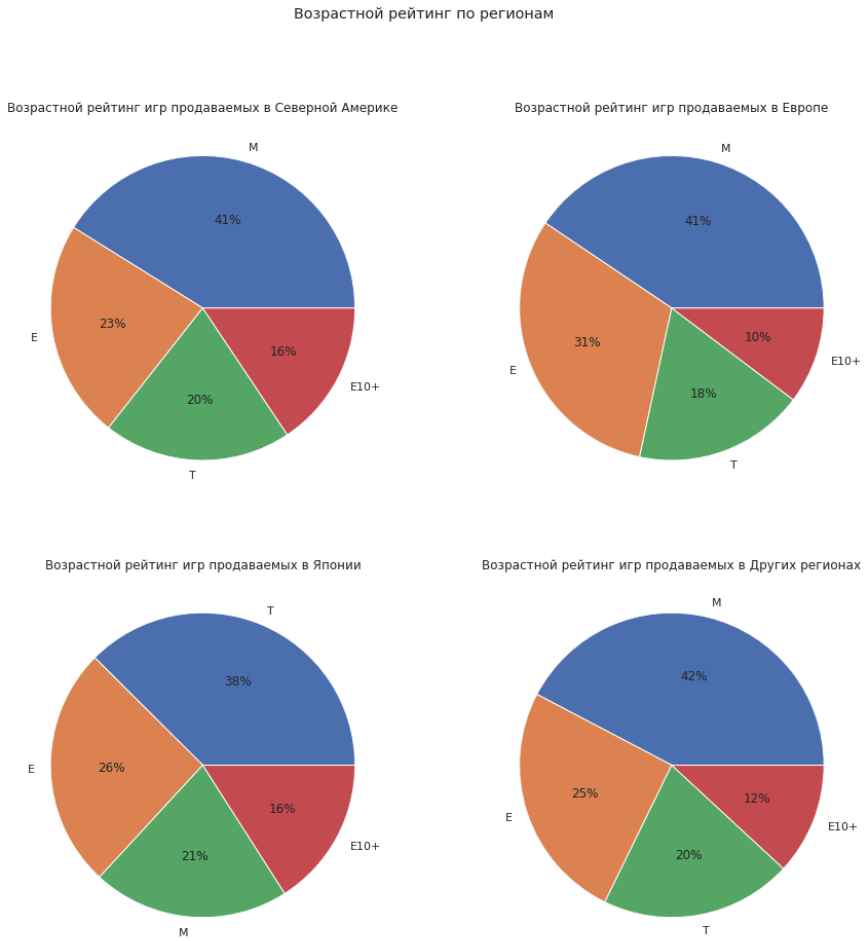
Интересно в чем причина, что на рынке Японии продается так много игр без рейтинга ESRB

### ✍ Комментарий Ирины 1

Возможно в Японии просто действует другая возрастная рейтинговая система, и рейтинговые оценки, которые работают на Западе ни на что не влияю в Японии. В Японии рейтинги выставляются по другой системе и обозначаются по другому.

👍

```
# сводные таблицы по платформам для трех регионов исключив данные без рейтинга
data_na_rating = data_p.query('rating!="NV"').\
    pivot_table(index='rating', values='na_sales', aggfunc='sum').\
    sort_values(by='na_sales',ascending = False)
data_eu_rating = data_p.query('rating!="NV"').\
    pivot_table(index='rating', values='eu_sales', aggfunc='sum').\
    sort_values(by='eu_sales',ascending = False)
data_jp_rating = data_p.query('rating!="NV"').\
    pivot_table(index='rating', values='jp_sales', aggfunc='sum').\
    sort_values(by='jp_sales',ascending = False)
data_other_rating = data_p.query('rating!="NV"').\
    pivot_table(index='rating', values='other_sales', aggfunc='sum').\
    sort_values(by='other_sales',ascending = False)


f,ax = plt.subplots(2,2)
f.set_size_inches(15, 15)
plt.suptitle('Возрастной рейтинг по регионам')
data_na_rating.plot(
    kind='pie',
    y='na_sales',
    autopct='%1.0f%%',
    ax=ax[0,0],
    title='Возрастной рейтинг игр продаваемых в Северной Америке',
    ylabel='',
    legend=False
)
data_eu_rating.plot(
    kind='pie',
    y='eu_sales',
    autopct='%1.0f%%',
    ax=ax[0,1],
    title='Возрастной рейтинг игр продаваемых в Европе',
    ylabel='',
    legend=False
)
data_jp_rating.plot(
    kind='pie',
    y='jp_sales',
    autopct='%1.0f%%',
    title='Возрастной рейтинг игр продаваемых в Японии',
    ylabel='',
    legend=False,
    ax=ax[1,0]
)
data_other_rating.plot(
    kind='pie',
    y='other_sales',
    autopct='%1.0f%%',
    title='Возрастной рейтинг игр продаваемых в Других регионах',
    ylabel='',
    legend=False,
    ax=ax[1,1]
)
plt.show()
```




Северная Америка:

- M (для лиц старше 17) - 41%
- E (для всех возрастных категорий) - 23%
- T (для лиц старше 13 лет) - 20%
- E10+ (для лиц старше 10 лет) - 16%

Европа:

- M - 41%
- E - 31%
- T - 18%
- E10+ - 10%

Япония:

- T - 38%
- E - 26%
- M - 21%
- E10+ - 16%

Выводы: В Северной Америке, Европе самыми продаваемыми являются игры с рейтингом для взрослых -41% и без возрастных ограничений E - 23-31%. В Японии самыми продаваемыми являются игры для аудитории старше 13 лет - 38% и без возрастных ограничений - 26%.

Пользователь из Северной Америки играет в игры жанра Shooter на платформе PS4 и XOne

Пользователь из Европы играет в игры жанра Shooter на платформе PS4

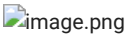
Пользователь из Японии играет в игры Action и Role-Plaing на платформе 3DS

**⚠ Комментарий ревьюера**

Сможешь добавить 4-ый круг с продажами в другом регионе → other_sales

и стоит оформить графики раздела ТОП-5:

- оформить "двухуровневый заголовок" - и у всех трех или 4-х графиков вместе, и у каждого по отдельности;



**✍ Комментарий Ирины 1**

Сделала

## Общий вывод

ЗАДАЧА

выявить определяющие успешность игры закономерности. Это позволит сделать ставку на потенциально популярный продукт и спланировать рекламные кампании на 2017 год. (Сейчас декабрь 2016 года)

ОПИСАНИЕ ДАННЫХ

данные из открытых источников, доступны исторические данные о продажах игр, оценки пользователей и экспертов, жанры и платформы (например, Xbox или PlayStation). Данные до 2016 года. Путь к файлу: /datasets/games.csv. • Name — название игры • Platform — платформа • Year_of_Release — год выпуска • Genre — жанр игры • NA_sales — продажи в Северной Америке (миллионы проданных копий) • EU_sales — продажи в Европе (миллионы проданных копий) • JP_sales — продажи в Японии (миллионы проданных копий) • Other_sales — продажи в других странах (миллионы проданных копий) • Critic_Score — оценка критиков (максимум 100) • User_Score — оценка пользователей (максимум 10) • Rating — рейтинг от организации ESRB (англ. Entertainment Software Rating Board). Эта ассоциация определяет рейтинг компьютерных игр и присваивает им подходящую возрастную категорию.

ПРЕДОБРАБОТКА

Первое знакомство с данными показывает:

- объем данных 16715 строк
- в данных наблюдаются пропуски в стобцах: наименование, год релиза, жанр, оценки пользователей и критиков, возрастной рейтинг
- возможно понадобится замена типов данных в стобцах на более подходящие: год выпуска, оценка пользователя и оценка критиков

В ходе предобработки данных потребовалось осущебыли следующие операции:

- Названия столбцов таблицы данных были приведены к единообразному виду – к нижнему регистру
- Преобразованы типы данных:
  - Год выпуска преобразован к целочисленному типу
  - Оценнки пользователей 0-10 приведены к числовым значениям с запятой вместо строковых
  - Оценки критиков 0-100 приведены к целочисленным, т.к. нет данных со значениями после запятой
- Удалены данные с пропусками в годе выпуска. Всего удалено 2% данных
- Помечены пропуски в данных оценки пользователя и оценки критиков нулями. Оценки пользователей со строковыми данными tbd преобразованы в нули.
- В дальнейшем анализе данные с нулями в оценках не учитывались.
- Пропуски в названиях игр удалены (всего два пропуска)
- Доля пропусков в данных рейтинга 41 процент помечена значением NV – not value
- Явные дубликаты не обнаружены
- Обнаружены дубликаты с разделением данных по регионам на две строки, например: National Geographic Panda (JP sales) и National Geographic Panda (US sales). Т.к. дальнейшее исследование предполагает оценку суммарных продаж по всем регионам данные дубликаты не объеденены в одну строку. Строк с возможными дубликатами со словом sales обнаружено 142 строки.
- Обнаружены дубликаты среди записей name+platform. Только одна запись из найденных признана дублем.
- Добавлен столбец с суммарными продажами по всем регионам

ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ

- Количество игр выпускаемых в разные годы:

до 1995 года выпускалось крайне мало игр, данные по этим годам в дальнейшем можно точно не учитывать. С 1995 года начинается наращивание выпуска игр и пик разнообразных игр приходится на 2008-2009 годы. Далее разнообразие выпускаемых игр снижается и в 2012-2016 достигает уровня как в 2001 году в промежутке от 400 до 600 в год.

- Лидирующие платформы по сумме продаж:

лидирующие платформы по сумме продаж за все годы: PS2, X360, PS3, Wii, но эти платформы к 2016 году практически или не имеют продаж совсем или сильно снижают. Существенные продажи в 2014-2016гг у платформ: PS4, XOne, WiiU, X360, PS3, Wii

- Определение актуального периода исследования:

За актуальный период выбран период 2015-2016г когда имеют продажи новые платформы. В 2014 году еще существенное влияние оказывают старые платформы.

Определен средний срок жизни игровых платформ:10-11 лет, что свидетельствует о завершении срока жизни платформ X360, PS3, Wii к 2017 году

- Определение наиболее прибыльных платформ

Выделены самые новые платформы PS4, WiiU, Xone, возникшие в 2012-2013 годах. Срок их жизни 4 года до н.в., следовательно есть еще куда им развиваться.

Для актуального периода 2015-2016 гг. для всех платформ 75% данных суммарных продаж составляет до 0,6 млн проданных копий. Медианное значение для наиболее прибыльных платформ находится в районе 0,1 млн копий.

Самые экстремальные выбросы уникально-большие продажи у платформ PS4 более 14млн проданных копий.

- Исследование влияния отзывов пользователей и критиков для платформы PS4:

сумма продаж линейно не коррелирует с оценкой пользоватлей на популярой игровой платформе. Коэффициент менее 6%. Отзывам критиков явно доверяют больше, наблюдается средняя корреляция суммарных продаж с отзывами критиков - 39%.

- Исследование влияния отзывов для других потенциально прибыльных платформ: X360, PS4, WiiU

для платформ: XOne, PS4 наблюдается средняя линейная корреляция в районе 39%-43% суммарных продаж от оценки критиков и отсутствие линейной корреляции с отзывами пользоватлей. для платформы WiiU суммарные продажи средне-коррелируют как с отзывами критиков: 32%, так и с отзывами пользователей 36%. Чем-то эта платформа отличается от остальных. Возможно такой факт влияет на ее отличительные данные: "Одной из целей WiiU было привлечь более серьёзную игровую аудиторию"

- Распределение игр по жанрам.

Самые большие продажи среди жанров: Action, Shooter, Sports, Role-Playing. Причем максимум Action добивается за счет продаж некоторых бесцеллеров, если эти бесцеллеры не учитывать, то Shooter выходит на первое место

- Исследование Топ5 жанров по регионам:

Популярные жанры игр в Америке и в Европе схожи, но отличаются от Японии, что еще раз доказывает различие в менталитете с восточными народами.

В Северной Америке и в Европе самые популярные игровые жанры Shooter. Но в Америке популярность этого жанра несколько выше (32%), чем в Европе (27%). На втором месте жанр Action, доля игр с этим жанром схода - 22-24%. На третьем месте жанр Sports - 17%-19%, так же примерно одинаковой долей. Role-Playing занимает четвертое место: 12-13%. Доля других игр около 14%

В Японии наблюдается совсем иная картина. На первое место выходят игры с жанром Action и занимают без малого почти половину рынка. А вот популярный на западе жанр Shooter спускается н третье место и составляет всего 13%. Второе место занимает жанр Role-Playing - 25%, примерно столькоже, сколько Shooter в Европе, тогда как на западе этот жанр лиш на четвертом месте.

- Топ-5 популярных платформ по регионам:

В Северной Америке пользователи отдают предпочтения 2 платформам: PS4, XOne, причем PS4 занимает лидирующее место 42%, а XOne - 33%. Остальные платформы составляют незначительную долю (5-7%)

В Веропе с существенным отрывом лидирует платформа PS4 - 56%, больше половины всех платформ. На втором месте так же как и в Америке XOne, до ее доля 19%. Остальные платформы так же занимают незначтительный процент 5-8%

В Японии картина совершенно другая. На первом месте платформа 3DS- 44%, на втором PS4 - 23%, на третьем PS3- 16%. Остальные платформи 8-9%

- Портреты пользователей:

Пользователь из Северной Америки играет в игры жанра Shooter на платформе PS4 и XOne

Пользователь из Европы играет в игры жанра Shooter на платформе PS4

Пользователь из Японии играет в игры Action и Role-Plaing на платформе 3DS

- Возрастной рейтинг:

Северная Америка: продажи без рейтинга - 25%

Европа:продажи без рейтинга - 24%

Япония:продажи без рейтинга - 62%

**❌ Комментарий ревьюера**

Итоговый вывод технически составлен грамотно

стоит перепроверить результаты после определения актуального периода и исправления всех комментариев, можно обновить названия самых актуальных платформ, жанров и рейтингов, какую долю они занимают на исследуемых рынках

## ✔ Комментарий ревьюера

Ты выполнила практически все пункты проекта, молодец! Проведен значительный объем исследования

Критические ❌ комментарии связаны с неточностями:

- переопределить актуальный период
- поправить перечень перспективных платформ
- поправить графики в ТОП-5
- перепроверить промежуточные и итоговый выводы после всех исправлений

Стоит обратить внимание на ⚠ комментарии...

Если будут вопросы про мои комментарии - задавай, если какой-то формат взаимодействия не устраивает или есть какие-то другие