День №3

Анализ и визуализация данных

1. Базовые вычисление

Pandas - мощный инструмент для вычислений. Пройдемся по базовым командам, позволяющим рассчитать различные статистические показатели.

1.1 Минимум, максимум, среднее, медиана, сумма и т.п.

Задача: создать датафрейм, состоящий из 5 столбцов и 5000 строк случайных чисел от 0 до 100.
Подсказка: используйте функцию np.random.randint() для создания массива данных

Ввод [220]:

```
# Импортируем нужные библиотеки и создадим датафрейм случайных чисел
import pandas as pd
import numpy as np

df = pd.DataFrame(data=np.random.randint(0, 200, (5000, 5)))
df.columns = ['a', 'b', 'c', 'd', 'e']
df.head()
```

Out[220]:

	a	b	c	d	e
0	198	35	53	96	74
1	2	12	138	147	94
2	114	2	76	188	198
3	34	38	36	97	116
4	58	113	9	35	72

Ввод [224]:

```
# Запишем в отдельные столбцы датафрейма минимальное, среднее и максимальное значение любого
df['new_a'] = df['a'].min()
df['new_b'] = df['b'].mean()
df['new_c'] = df['c'].max()
df['new_d'] = df['d'].median()

df.head()
```

Out[224]:

	a	b	c	d	e	new_a	new_b	new_c	new_d
0	198	35	53	96	74	0	99.965	199	99.0
1	2	12	138	147	94	0	99.965	199	99.0
2	114	2	76	188	198	0	99.965	199	99.0
3	34	38	36	97	116	0	99.965	199	99.0
4	58	113	9	35	72	0	99.965	199	99.0

Ввод [226]:

```
# Часть из математических методов датафреймов позволяют применять их не только к столбцам, н
df['new_rows'] = df['a'] + df['b'] + df['c'] + df['d'] + df['e']
df['new_rows_2'] = df[['a', 'b', 'c', 'd', 'e']].sum(axis=1)

df.head()
```

Out[226]:

	a	b	c	d	e	new_a	new_b	new_c	new_d	new_rows	new_rows_2
0	198	35	53	96	74	0	99.965	199	99.0	456	456
1	2	12	138	147	94	0	99.965	199	99.0	393	393
2	114	2	76	188	198	0	99.965	199	99.0	578	578
3	34	38	36	97	116	0	99.965	199	99.0	321	321
4	58	113	9	35	72	0	99.965	199	99.0	287	287

1.2 Расчеты дополнительных признаков

В ходе анализа данных к имеющимся признакам часто рассчитываются дополнительные, в том числе категориальные. Для этих операций в pandas существует несколько функций, одна из которых - функция **.where() библиотеки numpy**, а вторая - функция apply()

Ввод [228]:

```
# Создадим новый столбец. В него запишем значения 'more than 100' для тех строк, где столбец
# и 'less then 100' для тех строк, где столбец d меньше 100

df['new_c_compare'] = np.where(df['c']>100, 'more than 100', 'less then 100')

df.head()
```

Out[228]:

	a	b	c	d	e	new_a	new_b	new_c	new_d	new_rows	new_rows_2	new_c_compare
0	198	35	53	96	74	0	99.965	199	99.0	456	456	less then 100
1	2	12	138	147	94	0	99.965	199	99.0	393	393	more than 100
2	114	2	76	188	198	0	99.965	199	99.0	578	578	less then 100
3	34	38	36	97	116	0	99.965	199	99.0	321	321	less then 100
4	58	113	9	35	72	0	99.965	199	99.0	287	287	less then 100

Ввод [229]:

```
df['c']>100
```

Out[229]:

```
0       False
1        True
2       False
3       False
4       False
        ...
4995     True
4996     True
4997    False
4998    False
4999     True
Name: c, Length: 5000, dtype: bool
```

Ввод [230]:

```
# Попробуем создать новый столбец, в котором проставим "четное-нечетное" для чисел из столбц
# с помощью функции apply
df['new_c_compare_2'] = df['c'].apply(lambda x: 'more than 100' if x > 100 else 'less then 1
4
df.head()
```

Out[230]:

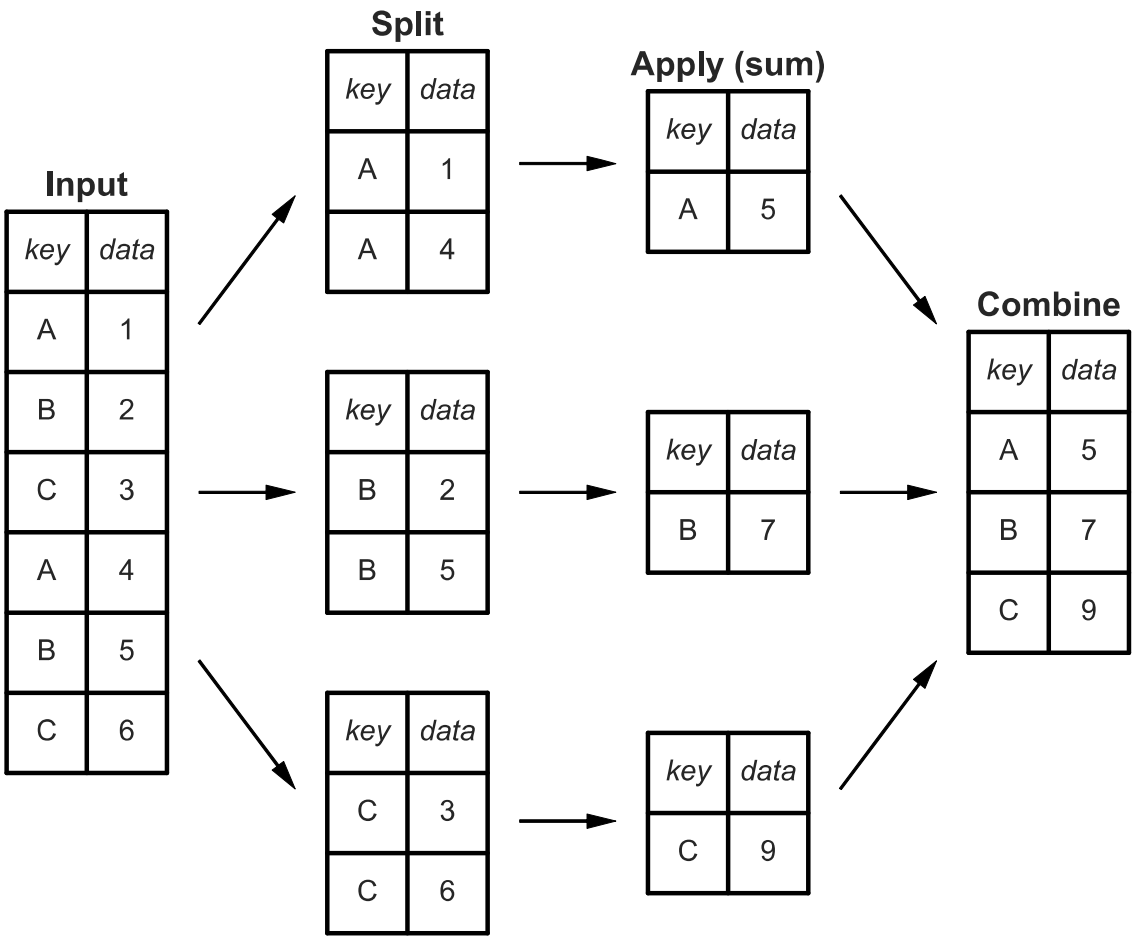
	a	b	c	d	e	new_a	new_b	new_c	new_d	new_rows	new_rows_2	new_c_compare
0	198	35	53	96	74	0	99.965	199	99.0	456	456	less then 100
1	2	12	138	147	94	0	99.965	199	99.0	393	393	more than 100
2	114	2	76	188	198	0	99.965	199	99.0	578	578	less then 100
3	34	38	36	97	116	0	99.965	199	99.0	321	321	less then 100
4	58	113	9	35	72	0	99.965	199	99.0	287	287	less then 100

Ввод [ ]:

```
1
```

2. Группировка и соединение датафреймов

Группировка данных позволяет получать агрегаты по различным группам. Фактически это процесс разделения данных на группы на основе значения в одном или нескольких столбцах и применения агрегатных функций (например, сумма, среднее значение, максимум, минимум) к этим группам.



Например, если у вас есть таблица, содержащая информацию о продажах различных товаров в различных регионах, вы можете сгруппировать данные по регионам и найти сумму продаж в каждом регионе.

Базовый синтаксис группировок выглядит следующим образом:

```
dataframe.groupby(by=column_or_columns)
[column_to_aggregate].aggregate_function()
```

Ввод [232]:

```
# Применим к первому столбцу функцию группировки, рассчитав среднее значение
df.head()
```

Out[232]:

	a	b	c	d	e	new_a	new_b	new_c	new_d	new_rows	new_rows_2	new_c_compare
0	198	35	53	96	74	0	99.965	199	99.0	456	456	less then 100
1	2	12	138	147	94	0	99.965	199	99.0	393	393	more than 100
2	114	2	76	188	198	0	99.965	199	99.0	578	578	less then 100
3	34	38	36	97	116	0	99.965	199	99.0	321	321	less then 100
4	58	113	9	35	72	0	99.965	199	99.0	287	287	less then 100

Ввод [235]:

```
df.groupby(['new_c_compare']).agg({'a':'sum', 'b':'mean', 'c':'median'})
```

Out[235]:

	a	b	c
new_c_compare			
less then 100	243602	100.539943	51
more than 100	251842	99.416569	150

3. Соединение данных

Соединение датафреймов (merge) позволяет соединять между собой данные по какому-то ключу.

Ввод [2]:

```
#Создадим 2 случайных датафрейма: один со столбцами A и B, второй со столбцами A и C.
# Соединим их по столбцу A
pass
```

3. Давайте попрактикуемся!

Давай представим, что мы хотим создать собственный фильм, который войдет в топ рейтингов. Для этого мы должны определиться, в каком жанре создавать фильм, на какую продолжительность ориентироваться, с каким режиссером нам стоит работать и какой актерский состав выбрать.

Перед тем, как приступать к анализу любых данных, стоит провести **мозговой штурм** и ответьте на важные вопросы:

1. Какую бизнес-проблему мы собираемся решить и какова моя цель?
2. Какой анализ я хочу провести и почему?
3. Какое влияние на бизнес я могу оказать своими результатами?
4. Кто мои конечные пользователи или потребители проведенного анализа?

В основе анализа будут лежать данные о 1000 самых популярных фильмов в истории по версии IMDB

Полезные функции для чтения данных из файлов:

1. pd.read_excel() - чтение данных из файлов с расширениями *.xlsx, *.xls
2. pd.read_csv() - чтение данных из файлов с форматами *.csv, *.txt
3. pd.read_sql() - чтения данных из результата SQL запроса в базу данных

Ввод [240]:

```
# С помощью функции pd.read_excel() загрузите данные в датафрейм и выведите на экран первые
# Важно корректно указать путь до файла. Самый легкий способ - "положить" файл в ту же папку
# запущен jupyter-notebook, и передайте названия файла в функцию pd.read_excel()
df = pd.read_excel('data_for_lesson/imdb_top_1000.xlsx')
df.head()
```

Out[240]:

	Poster_Link	Series_Title	Released_Year	Certificate	Runtime	Genre	IM
0	https://m.media-amazon.com/images/M/MV5BMDFkYT...	The Shawshank Redemption	1994	A	142 min	Drama	
1	https://m.media-amazon.com/images/M/MV5BM2MyNj...	The Godfather	1972	A	175 min	Crime, Drama	
2	https://m.media-amazon.com/images/M/MV5BMTMxNT...	The Dark Knight	2008	UA	152 min	Action, Crime, Drama	
3	https://m.media-amazon.com/images/M/MV5BMWMwMG...	The Godfather: Part II	1974	A	202 min	Crime, Drama	
4	https://m.media-amazon.com/images/M/MV5BMWU4N2...	12 Angry Men	1957	U	96 min	Crime, Drama	

Ввод [239]:

```
# С помощью функции info() посмотрим типы данных нашего датасета
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 16 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Poster_Link    1000 non-null   object 
 1   Series_Title   1000 non-null   object 
 2   Released_Year  1000 non-null   object 
 3   Certificate    899 non-null    object 
 4   Runtime        1000 non-null   object 
 5   Genre          1000 non-null   object 
 6   IMDB_Rating    1000 non-null   float64
 7   Overview       1000 non-null   object 
 8   Meta_score     843 non-null    float64
 9   Director       1000 non-null   object 
 10  Star1          1000 non-null   object 
 11  Star2          1000 non-null   object 
 12  Star3          1000 non-null   object 
 13  Star4          1000 non-null   object 
 14  No_of_Votes    1000 non-null   int64  
 15  Gross          831 non-null    object 
dtypes: float64(2), int64(1), object(13)
memory usage: 125.1+ KB
```

3.2. Преобразование данных

Практически всегда при анализе данных вы будете сталкиваться с тем, что данные представлены в нестандартном формате. Например, заполнены не все ячейки, в каких-то столбцах присутствуют странные символы, не несущие полезной информации. По этим причинам данные необходимо предобработать

> **Предобработка занимает до 80% времени при анализе данных и построение различных моделей**

Для первого представления о том, какими данными нам придется манипулировать, используют функцию info(). Функция дает базовое представление о том, какие типы данных есть в нашем датафрейме, сколько их, есть ли пропуски и какой объем памяти занимают наши данные.

Ввод [244]:

```
# Названия столбцов начинаются с заглавных букв. Давайте приведем все названия к нижнему ре
my_columns_list = df.columns.tolist()
my_columns_list_1 = []

for col in my_columns_list:
    my_columns_list_1.append(col.lower())

df.columns = my_columns_list_1
df.head()
```

Out[244]:

s_title	released_year	certificate	runtime	genre	imdb_rating	overview	meta_score	director	s
The vshank mption	1994	A	142 min	Drama	9.3	Two imprisoned men bond over a number of years...	80.0	Frank Darabont	Rob
The dfather	1972	A	175 min	Crime, Drama	9.2	An organized crime dynasty's aging patriarch t...	100.0	Francis Ford Coppola	Ma Bra
e Dark Knight	2008	UA	152 min	Action, Crime, Drama	9.0	When the menace known as the Joker wreaks havo...	84.0	Christopher Nolan	Chri
The lfather: Part II	1974	A	202 min	Crime, Drama	9.0	The early life and career of Vito Corleone in ...	90.0	Francis Ford Coppola	Pa
! Angry Men	1957	U	96 min	Crime, Drama	9.0	A jury holdout attempts to prevent a miscarria...	96.0	Sidney Lumet	H Fo

Ввод [254]:

```
df = pd.read_excel('data_for_lesson/imdb_top_1000.xlsx')

my_columns_list = df.columns.tolist()
my_columns_list_1 = []

for col in my_columns_list:
    my_columns_list_1.append(col.lower())

df.columns = my_columns_list_1


# В столбце "runtime" присутствует время
df['runtime'] = df['runtime'].str.replace(' min', '')
df['runtime'] = df['runtime'].astype(int)

# В столбце "gross" разряды разделены запятыми, а сам столбец - текстовый
df['gross'] = df['gross'].str.replace(',', '')
df['gross'] = df['gross'].astype(float)

# Столбец "released_year" в странном формате
df.head()
```

Out[254]:

s_title	released_year	certificate	runtime	genre	imdb_rating	overview	meta_score	director	s
The shank mption	1994	A	142	Drama	9.3	Two imprisoned men bond over a number of years...	80.0	Frank Darabont	Rob
The lfather	1972	A	175	Crime, Drama	9.2	An organized crime dynasty's aging patriarch t...	100.0	Francis Ford Coppola	Ma Bra
e Dark Knight	2008	UA	152	Action, Crime, Drama	9.0	When the menace known as the Joker wreaks havo...	84.0	Christopher Nolan	Chris E
The father: Part II	1974	A	202	Crime, Drama	9.0	The early life and career of Vito Corleone in ...	90.0	Francis Ford Coppola	Pa
Angry Men	1957	U	96	Crime, Drama	9.0	A jury holdout attempts to prevent a miscarria...	96.0	Sidney Lumet	He Fo

3.3 Базовые описательные статистики

Ввод [255]:

```
# Выведем общие статистики по данным с помощью функции describe
df.describe()
```

Out[255]:

	runtime	imdb_rating	meta_score	no_of_votes	gross
count	1000.000000	1000.000000	843.000000	1.000000e+03	8.310000e+02
mean	122.891000	7.949300	77.971530	2.736929e+05	6.803475e+07
std	28.093671	0.275491	12.376099	3.273727e+05	1.097500e+08
min	45.000000	7.600000	28.000000	2.508800e+04	1.305000e+03
25%	103.000000	7.700000	70.000000	5.552625e+04	3.253559e+06
50%	119.000000	7.900000	79.000000	1.385485e+05	2.353089e+07
75%	137.000000	8.100000	87.000000	3.741612e+05	8.075089e+07
max	321.000000	9.300000	100.000000	2.343110e+06	9.366622e+08

Ввод [260]:

```
# Посмотрим, все ли данные заполнены. Для этого используем функцию isna() и функцию sum()
df.isna().sum()
```

Out[260]:

```
poster_link        0
series_title       0
released_year      0
certificate      101
runtime            0
genre              0
imdb_rating        0
overview           0
meta_score       157
director           0
star1              0
star2              0
star3              0
star4              0
no_of_votes        0
gross              0
dtype: int64
```

Ввод [259]:

```
# Что делать с незаполненными данными?
df['gross'] = df['gross'].fillna(df['gross'].mean())
df.head()
```

Out[259]:

	poster_link	series_title	released_year	certificate	runtime	genre	imdb
0	https://m.media-amazon.com/images/M/MV5BMDFkYT...	The Shawshank Redemption	1994	A	142	Drama	
1	https://m.media-amazon.com/images/M/MV5BM2MyNj...	The Godfather	1972	A	175	Crime, Drama	
2	https://m.media-amazon.com/images/M/MV5BMTMxNT...	The Dark Knight	2008	UA	152	Action, Crime, Drama	
3	https://m.media-amazon.com/images/M/MV5BMWMwMG...	The Godfather: Part II	1974	A	202	Crime, Drama	
4	https://m.media-amazon.com/images/M/MV5BMWU4N2...	12 Angry Men	1957	U	96	Crime, Drama	

3.4 Базовые выборки

Ввод [264]:

```
# Посмотрим топ фильмов по различным показателям. В pandas это можно сделать множеством спос
df[df['gross']==df['gross'].max()]
```

Out[264]:

eries_title	released_year	certificate	runtime	genre	imdb_rating	overview	meta_score	director	s
Star Wars: :pisode VII The Force Awakens	2015	U	138	Action, Adventure, Sci-Fi	7.9	As a new threat to the galaxy rises, Rey, a de...	80.0	J.J. Abrams	[R

Ввод [265]:

```
df[df['gross']==df['gross'].min()]
```

Out[265]:

ter_link	series_title	released_year	certificate	runtime	genre	imdb_rating	overview	meta_score	c
ı.media-ΓU4NT...	Adams æbler	2005	R	94	Comedy, Crime, Drama	7.8	A neo-nazi sentenced to community service at a...	51.0	-

3.5 Базовые группировки

Ввод [267]:

```
# Попробуем посмотреть на динамику производства фильмов по годам
# Сгруппируем данные по столбцу Released_Year и применим агрегатную функцию
df.groupby(['released_year']).agg({'series_title':'count'})
```

Out[267]:

released_year	series_title
1920	1
1921	1
1922	1
1924	1
1925	2
...	...
2017	22
2018	19
2019	23
2020	6
PG	1

100 rows × 1 columns

Ввод [271]:

```
df = df[df['released_year']!='PG']
df.head(2)
```

Out[271]:

	poster_link	series_title	released_year	certificate	runtime	genre	imdb_r
0	https://m.media-amazon.com/images/M/MV5BMDFkYT...	The Shawshank Redemption	1994	A	142	Drama	
1	https://m.media-amazon.com/images/M/MV5BM2MyNj...	The Godfather	1972	A	175	Crime, Drama	

Ввод [273]:

```
# Посмотрим на режиссеров фильмом. Кто из них выпустил больше всего фильмов, которые попали
df.groupby(['director']).agg({'series_title':'count'}).sort_values('series_title')
```

Out[273]:

director	series_title
Aamir Khan	1
Mikael Håfström	1
Michel Hazanavicius	1
Michel Gondry	1
Michael Radford	1
...	...
Martin Scorsese	10
Akira Kurosawa	10
Hayao Miyazaki	11
Steven Spielberg	13
Alfred Hitchcock	14

548 rows × 1 columns

Ввод [275]:

```
# Какой из актеров и сколько раз снимался в фильмах на первых ролях?
df.groupby(['star1', 'director']).agg({'series_title':'count'}).sort_values('series_title')
```

Out[275]:

star1	director	series_title
Aamir Khan	Ashutosh Gowariker	1
Matt Damon	James Mangold	1
	Ridley Scott	1
Matthew Broderick	Edward Zwick	1
	John Hughes	1
...	...	...
Ethan Hawke	Richard Linklater	4
Robert De Niro	Martin Scorsese	6
Charles Chaplin	Charles Chaplin	6
Ethan Coen	Joel Coen	6
Toshirô Mifune	Akira Kurosawa	7

897 rows × 1 columns

Ввод [152]:

```
# Какие фильмы оказались самыми успешными в расчете на 1 минуту показа?
pass
```

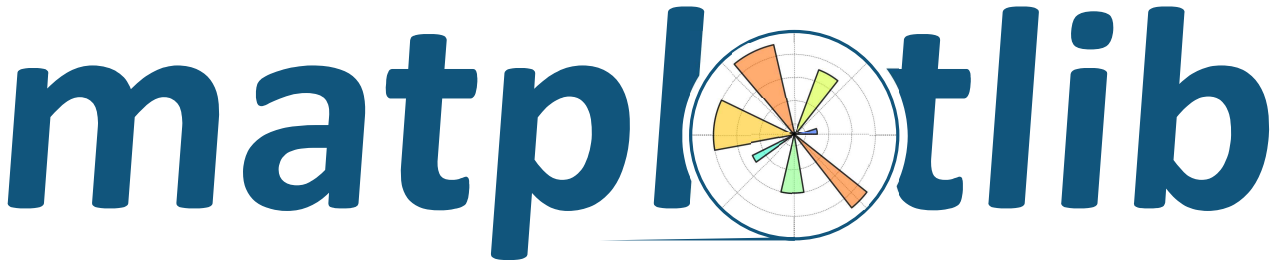
Ввод [167]:

```
# в pandas есть удобная функция value_counts(), которая возвращает уникальные значения из
# серии и количество в исходной серии
pass
```

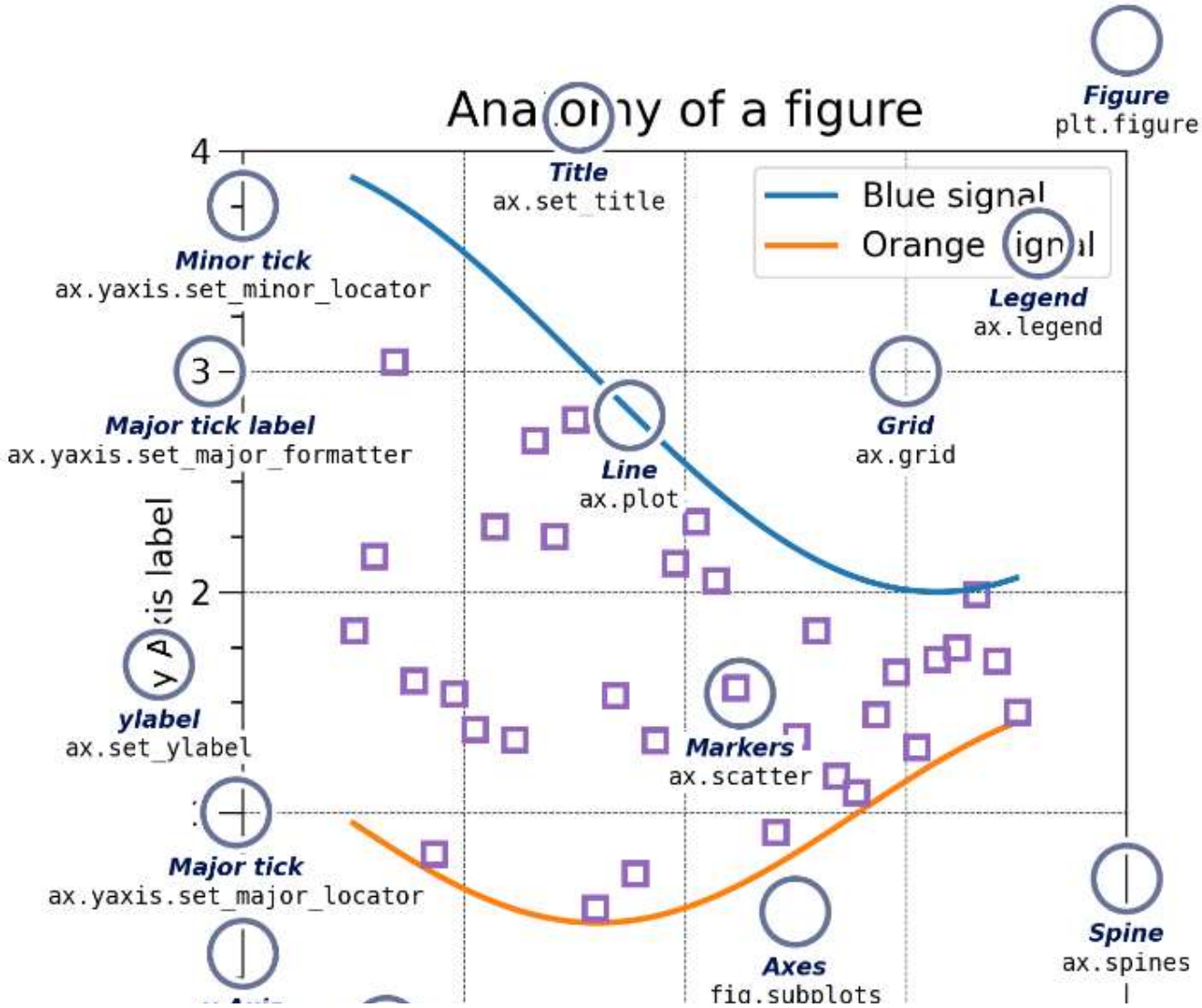
4. Визуализация данных

Визуализация результатов анализа данных позволяет нам наглядно продемонстрировать выводы, полученные в ходе анализа. Для визуализации данных в python существует множество библиотек, мы с вами познакомимся с базовыми: **matplotlib** и **seaborn**

Библиотека Matplotlib - одна из самых первых библиотек визуализации данных в python. Она была создана в 2003 году в качестве open-source аналога Matlab. Несмотря на возраст, библиотека по-прежнему является довольно популярным инструментом для визуализации данных.



Matplotlib состоит из множества модулей. Модули наполнены различными классами и функциями, которые иерархически связаны между собой. Каждый рисунок, созданный с помощью библиотеки matplotlib, состоит из нескольких элементов, каждый из которых мы можем редактировать.



Ввод [276]:

```
# де-факто стандарт вызова pyplot в python
import matplotlib.pyplot as plt

# отображение объектов matplotlib в jupyter без создания отдельного окна
%matplotlib inline

plt.style.use('ggplot')
```

4.1. Figure и Axes - рисунок и область рисования

Самым главным объектом в matplotlib является рисунок **Figure**. Создать рисунок в matplotlib означает создать экземпляр класса Figure с помощью метода **plt.figure()**. После создания объекта Figure с помощью различных методов (метода plt.add_axes() - тонкая настройка, метода **plt.subplots()** - грубая) можно создать различные объекты **Axes** (по умолчанию создается axes прямоугольной формы).

Axes является одним из самых важных классов matplotlib. **Axes** это, прежде всего, область рисования множества объектов, которая включает в себя много методов (plot(), text(), hist() и др.). Эти методы принимают в качестве параметров данные и автоматически создают необходимые экземпляры **Artist**.

Как создать рисунок и область рисования?

Самая удобная команда для одновременного создания и объекта Figure, и Axes - **fig, ax = plt.subplots()**. Возвращает коржет из объетов figure и axes.

plt.subplots() может принимать на себя следующие (основные) параметры:

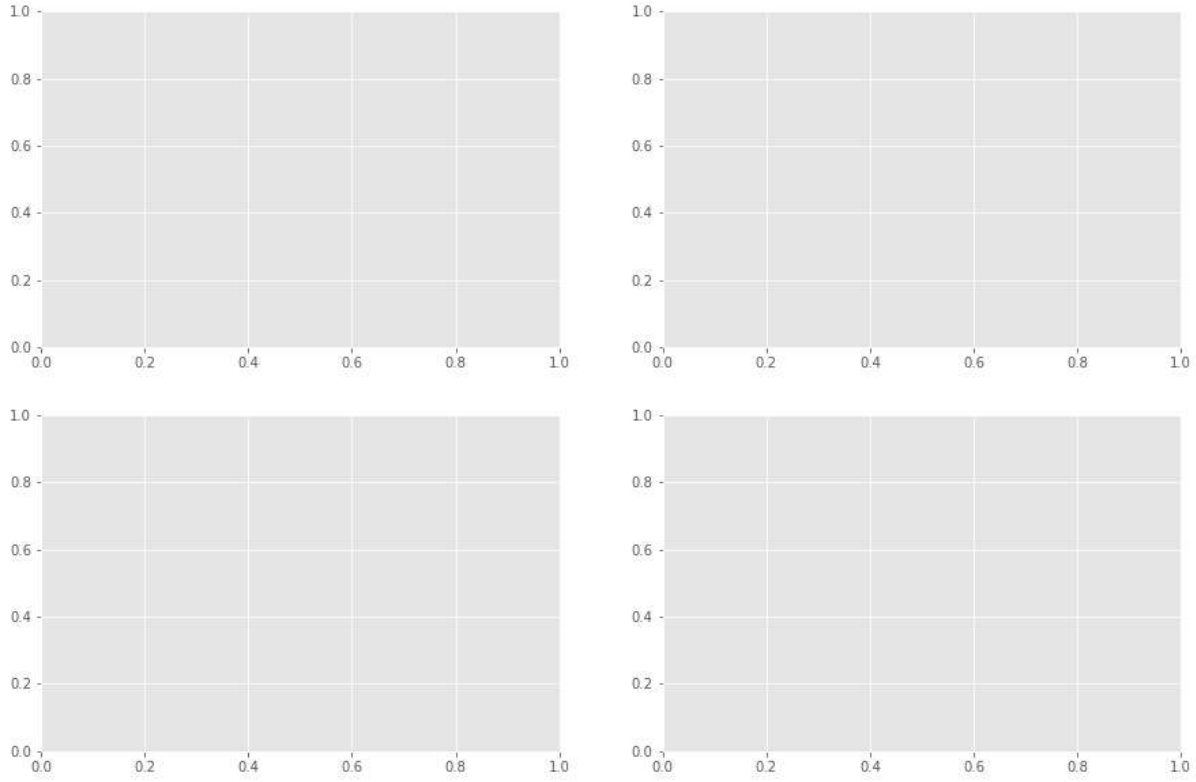
- nrows, ncols - количество строк и стоблцов областей рисования;
- sharex, sharey - объединение координатный осей для областей рисования.

Ввод [278]:

```
fig, ax = plt.subplots(nrows=2, ncols=2, figsize=(15, 10))

# for i in range(len(ax)):
#     for j in range(len(ax[i])):
#         ax[i][j].text(0.35, 0.5, f'ax[{i}][{j}]', fontsize=15)

plt.show()
```



4.2. Scatter, Plot, Bar, Hist, Pie, fill_between

Каждая из функций принимат в качестве параметров последовательность значений координат y и x (x не всегда), а также ряд параметров форматирования (**необязательно**). Параметры форматирования включают в себя работу с цветом, формой, типом линии или маркера, толщиной линий, степенью прозрачности элементов, размером и типом шрифта и другими свойствами.

Каждая из функций обладает рядом свойств, основные из которых:

1. plot()
   - color - цвет линии;
   - linestyle - тип линии;
   - linewidth - толщина линии;
   - alpha - прозрачность;
2. scatter()
   - color, alpha
   - marker - тип маркера
   - edgecolors - цвет обрамления маркера
   - s - размер маркера
3. bar()
   - color, alpha
   - width - толщина столбцов
4. hist()
   - bins - кол-во полос (бинов), по которым будет разбит массив (10 по умолчанию)
   - orientation - ориентация (horizontal, vertical)
5. pie()
   - labels - наименование кусочков
   - radius - радиус пирога
   - explode - выступ кусочков

Полный список см. в документации.

Ввод [280]:

```
# Первая визуализация большинства проектов анализа данных - матрица корреляций
# импортируем библиотеку seaborn (import seaborn as sns) и с помощью функции heatmap()
# построим матрицу корреляций
df.corr()
```

Out[280]:

	runtime	imdb_rating	meta_score	no_of_votes	gross
runtime	1.000000	0.244112	-0.031399	0.173304	0.124073
imdb_rating	0.244112	1.000000	0.268641	0.495361	0.091350
meta_score	-0.031399	0.268641	1.000000	-0.018519	-0.031429
no_of_votes	0.173304	0.495361	-0.018519	1.000000	0.549904
gross	0.124073	0.091350	-0.031429	0.549904	1.000000

Ввод [284]:

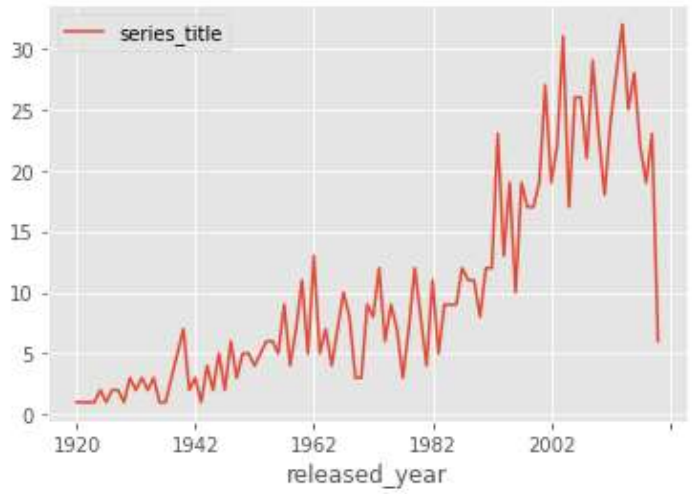
```
1 import seaborn as sns
2 fig, ax = plt.subplots(figsize=(10, 10))
3 sns.heatmap(df.corr(), annot=True)
4 plt.show()
```

Ввод [286]:

```
# Попробуем визуализировать данные по количеству произведенных фильмов по годам
df.groupby(['released_year']).agg({'series_title':'count'}).plot()
```

Out[286]:

```
<AxesSubplot:xlabel='released_year'>
```

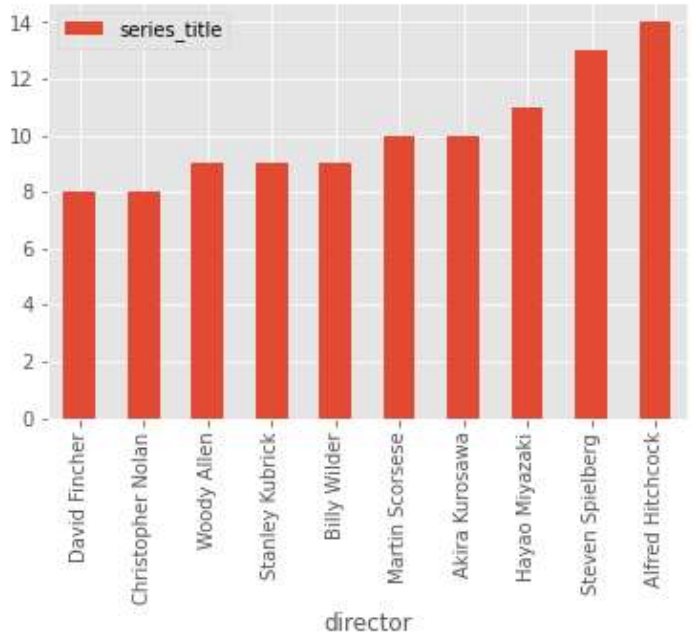


Ввод [292]:

```
# Попробуем визуализировать данные по топ-5 режиссерам, которые произвели наибольшее число ф
df.groupby(['director']).agg({'series_title':'count'}).sort_values('series_title')[-10:].pl
```

Out[292]:

```
<AxesSubplot:xlabel='director'>
```



Ввод [ ]:

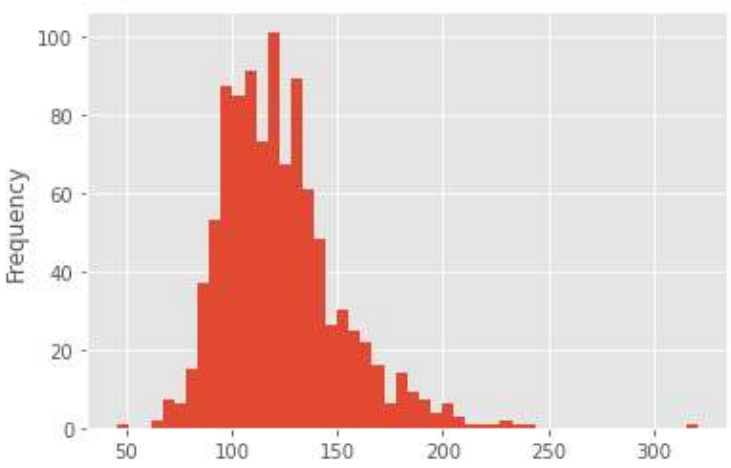
```
# Выведем кол-во фильмов по рейтингу и отобразим результаты на графике в виде баров
pass
```

Ввод [297]:

```
# А что по распределению продолжительности фильмов? Построем его
df['runtime'].plot(kind='hist', bins=50)
```

Out[297]:

```
<AxesSubplot:ylabel='Frequency'>
```



5. Пробуем решить нашу задачу и ответить на вопросы

5.1. С каким режиссером стоит работать?

Ввод [181]:

```
pass
```

5.2. В каком жанре создавать фильм?

Ввод [182]:

```
pass
```

5.3. На какую продолжительность фильма ориентироваться?

Ввод [183]:

```
1 pass
```

5.4. Каких актеров выбрать на главные роли?

Ввод [184]:

```
1 pass
```