# Описательная статистика. Анализ данных с помощью `Pandas`

## Цель занятия

После освоения темы вы сможете:

- получить информацию о `DataFrame`, вычислить описательные статистики для числовых данных, обратиться к элементам `DataFrame` по индексу и порядковому номеру, изменить индекс;
- выполнять поиск, фильтрацию и сортировку `DataFrame` с применением методов библиотеки `Pandas`;
- вычислять статистику по признакам, применять функции к данным, рассчитывать новые значения;
- работать с несколькими таблицами с помощью инструментов библиотеки `Pandas`;
- а также разберете построение графиков в `Pandas`;
- повторите основы математической статистики.

## План занятия

5. Построение графиков в `Pandas`
6. Основы математической статистики

# Конспект занятия

## 1. Описательная статистика

Переменные, с которыми мы будем работать, можно разделить на два типа:

- **Категориальные (неквантифицируемые характеристики)** — например, имена людей:

```
categorical_var = ['Misha','Dasha','John']
```

- **Численные (квантифицируемые характеристики)** — например, рост человека:

```
numeric_var = [187, 165, 163]
```

Каждый из типов переменных можно разделить на два подтипа:

- **Категориальные переменные:**
    - **Номинальные**: нет естественного порядка, например, знаки зодиака. 12 знаков зодиака ассоциируются с определенными характеристиками, но мы не можем установить среди них никакого порядка.
    - **Порядковые**: есть определенный порядок категорий, например, оценки. Предположим, система оценивания буквы A, B, C, D, F. Буквам соответствует определенный процент выполнения некоторой работы. Мы не можем сказать, насколько A отличается от B, а B отличается от C. Но мы понимаем, что есть определенный логический порядок категорий — социальная конвенция. Этот порядок не естественный в том смысле, что для некоторых людей удовлетворительная оценка C будет предпочтительнее оценки A.

При определении порядковых переменных важно понимать, что этот порядок и есть определенная договоренность.

- **Численные переменные:**
  - **Интервальные**: «примерно бесконечное» число реальных значений. Например, доход в России. Мы можем определить нижнюю границу, а с определением верхней границы могут возникнуть проблемы. Она будет сводиться к количеству всех денег, которые существуют. На этом промежутке существует практически бесконечное количество значений, которые возможны для этой переменной.

    В силу той или иной ограниченности инструментов измерителей мы не можем дробить наши значения до бесконечности. Поэтому в реальной жизни эмпирически интервальными переменными будем считать те, для которых число реальных значений «примерно бесконечно», то есть стремится к бесконечности.
  - **Дискретные**: конечное значение реальных значений переменных. Например, оценки на каком-нибудь соревновании. В фигурном катании были оценки от 0.0 до 6.0 с шагом в 0.1. Мы можем перечислить все возможные значения оценок.

### Используемые метрики (меры)

Для понимания трендов в переменных необходим подсчет определенных метрик или мер.

Важная характеристика для числовых переменных — различные **меры центральности**. Это меры, которые описывают «типичное» или центральное значение в распределении.

Обозначим некоторые такие меры:

- **Среднее** (среднее арифметическое): $x_{mean} = \frac{\Sigma x_i}{n}$.
- **Медиана** (центральное значение упорядоченного вариационного ряда):

  - *n* четное — значение элемента центрального ранга упорядоченного вектора чисел;
  - *n* нечетное — среднее арифметическое элементов двух центральных рангов вектора чисел.

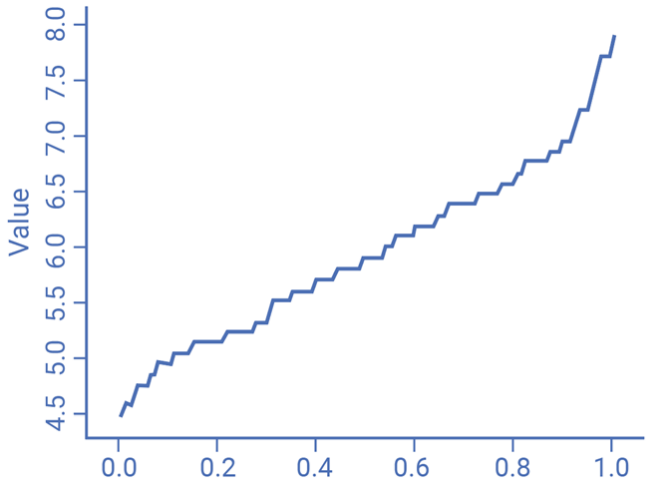
- **Мода** — самое часто встречающееся значение в выборке.

Среднее арифметическое значение зависимо от выбросов (нетипичных значений), медиана — нет. Поэтому можно использовать **усеченное среднее**:

$$x_{mean_{trimmed}} = \frac{\sum_{i=p+1}^{n-p} x_i}{n-2p}$$

Далее рассмотрим меры, которые не относятся к центральности, но они также достаточно важны.

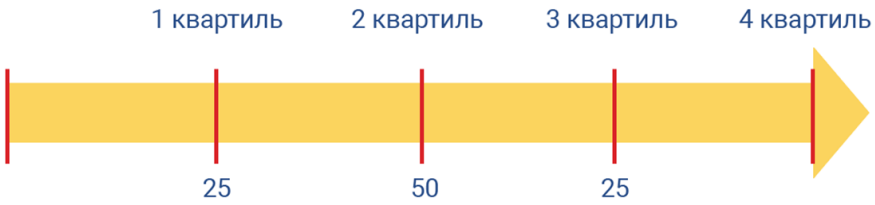
- **Процентиль** — значение упорядоченного вектора чисел, которое делит выборку в определенной пропорции. $i\%$ наблюдений имеют значения ниже $i$-го процентиля, $(100 - i)\%$ наблюдений — выше.




Частный случай процентиля — **квартили**.

- **Нижний квартиль** (25-й процентиль) — процентиль, разбивающий выборку в соотношении 25% к 75%.
- **Верхний квартиль** (75-й процентиль) — процентиль, разбивающий выборку в соотношении 75% к 25%.

Графически представить процентили можно в виде схемы:

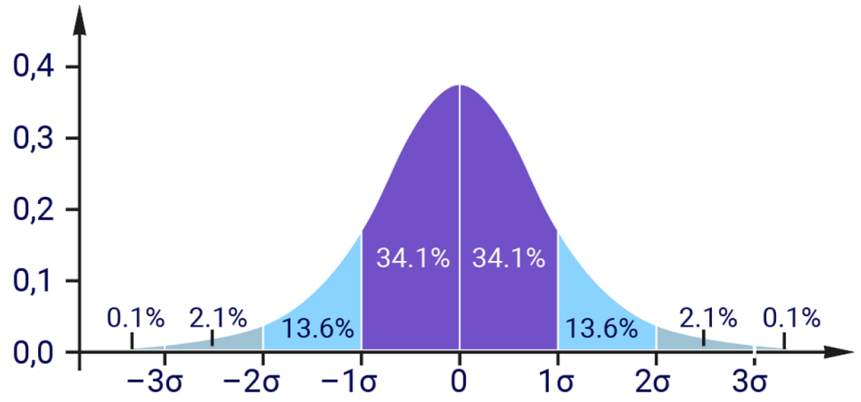



**Меры вариации** — меры, которые показывают, насколько наша переменная богата на различные значения.

Основные меры:

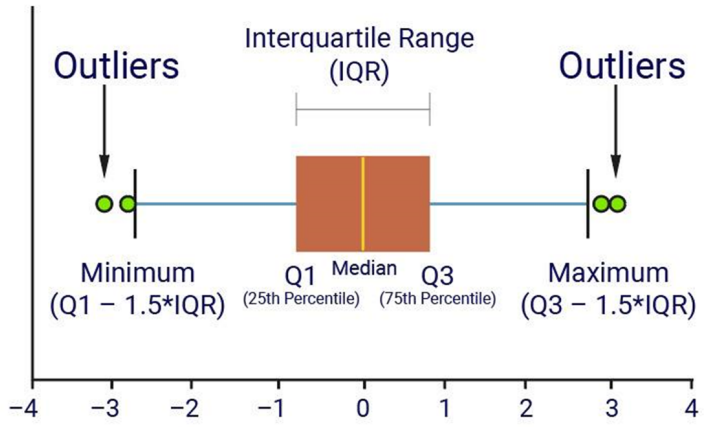
- Размах: $x_{max} - x_{min}$.
- Межквартильный размах (IQR): $\hat{x}_{0.75} - \hat{x}_{0.25}$.
- Выбросы: $(X_{outliner} \in [\hat{x}_{0.25} - 1.5IQR; \hat{x}_{0.75} + 1.5IQR])$.
- Дисперсия: $\sigma^2 = \frac{\Sigma(x_i - x_{min})^2}{n-1}$.
- Стандартное отклонение: $SD = \sqrt{\sigma^2}$.
- Среднее абсолютное отклонение: $MAD = \frac{\Sigma|x_i - x_{min}|}{n}$.

Меры вариации, в частности, стандартное отклонение, играют важную роль в статистике. Например, для стандартного нормального распределения при удалении от среднего значения выборки процент наблюдений в выборке уменьшается. Если распределение данных близко к нормальному, то практически никакие значения не будут превышать модуля суммы среднего и трех стандартных отклонений.




**Визуализация метрик**

Достаточно большое количество метрик мы можем визуализировать с помощью специального графика — «ящик с усами».




Ящик может быть как горизонтальным, так и вертикальным. Мы можем видеть на нем большое количество метрик. Вдоль ящика располагаются все возможные значения переменных, а также нанесено значение медианы переменной (желтая полоса).

Границы ящика определяются нижней и верхней квартилью. Высота ящика — межквартильный размах. «Усы» ящика — границы типичных наблюдений. Слева от

левого уса и справа от правого — выбросы или outliers. Как правило, выбросы обозначаются точками.

### Вычисление ключевых метрик в `Pandas`

Метрики можно вычислять по отдельности, например:

```
print('Mean total: ', df['total'].mean())
print('Median total: ', df['total'].median())
print('Lower Quartile of total: ', df['total'].quantile(0.25))
print('Upper Quartile of total: ', df['total'].quantile(0.75))
print('Interquartile Range of total ', df['total'].quantile(0.75) - \
      df['total'].quantile(0.25))
```

Результат:

```
Mean total:  23.727272727272727
Median total:  15.0
Lower Quartile of total:  11.0
Upper Quartile of total:  32.0
Interquartile Range of total  21.0
```

На основе этого мы можем выделить критерий для определения выбросов. В программе в отдельные переменные мы сохраняем верхнюю и нижнюю квартили, а также межквартильный размах. Далее по формуле для выброса подсчитаем его значение:

```
uq = df['total'].quantile(0.75)
lq = df['total'].quantile(0.25)
IQR = df['total'].quantile(0.75) -
df['total'].quantile(0.25)
print('Outliers thresholds: Lower - ', lq - 1.5*IQR,
'Upper - ', uq +1.5*IQR)
```

Результат:

```
Outliers thresholds: Lower -  -20.5 Upper -  63.5
```

Рассчитаем некоторые метрики вариаций:

```
print('Range of total: ', df['total'].max() - df['total'].min())
print('Variance of total: ', df['total'].var())
print('SD of total: ', df['total'].std())
print('MAD of total: ', df['total'].mad())
```

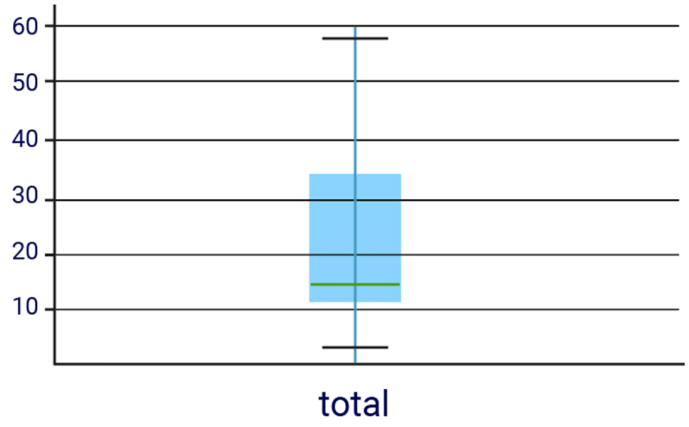
Результат:

```
Range of total:  58
Variance of total:  389.4181818181818
SD of total:  19.73368140561162
MAD of total:  15.652892561983471
```

Построим «ящик с усами»:

```
df.boxplot(column =['total'])
```

Результат:



### Асимметрия и эксцесс

При изучении распределения данных необходимо посмотреть на ряд других параметров.

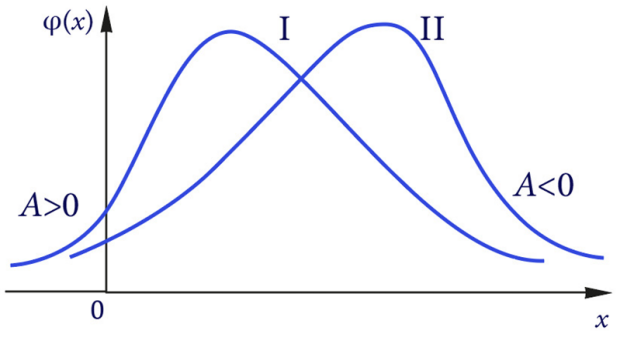
**Асимметрия** — отношение центрального момента третьего порядка к кубу стандартного отклонения. Фактически она показывает, насколько наше эмпирическое распределение отклоняется влево или вправо от **идеальной модели** — нормального распределения.

Асимметрия **положительна**, если «длинная часть» кривой распределения расположена справа от математического ожидания. Асимметрия **отрицательна**, если «длинная часть» кривой расположена слева от математического ожидания.

Формула для расчета коэффициента асимметрии:

$$a_s = \frac{\Sigma\left(x_i - x_{min}\right)^3}{\sigma^3}$$

Репрезентация различных распределений с точки зрения их коэффициента асимметрии *A*:




Коэффициент асимметрии в `Pandas` можно реализовать через метод `skew()`.

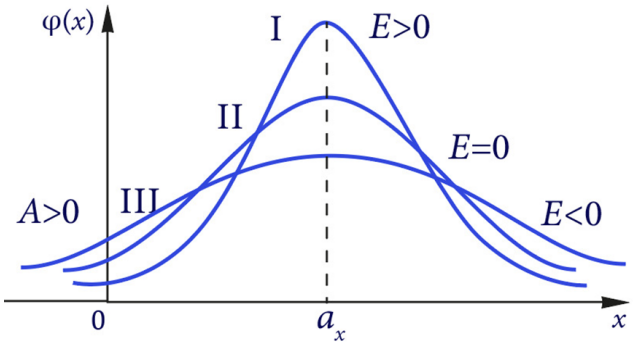
**Коэффициент эксцесса** показывает «остроту» распределения. Аналогично асимметрии, эксцесс показывает отклонение от идеальной модели нормального распределения, но уже не по горизонтали, а по вертикали (вверх-вниз). Кривые, более

островершинные, чем нормальная, обладают **положительным** эксцессом, более плосковершинные — **отрицательным** эксцессом.

Формула расчета коэффициента эксцесса:

$$e_s = \frac{\Sigma(x_i - x_{mean})^4}{\sigma^4} - 3$$

Пример:




Для расчета эксцесса в `Pandas` реализован метод `kurtosis()`.

## 2. Базовые операции с `DataFrame`

Рассмотрим работу с библиотекой `Pandas` — самой универсальной и известной библиотекой для обработки и анализа данных.

Чтобы работать с `Pandas`, необходимо знать три базовых понятия:

- `DataFrame` — двумерный неоднородный индексированный массив, таблица.
- `Series` — одномерный индексированный массив `ndarray`, столбец или строка.
- `Index` — индекс (список названий строк или столбцов).

### Настройка отображения

Первое, с чем мы познакомимся в `Pandas`, — настройка отображения. Чтобы начать работать с `Pandas`, необходимо импортировать библиотеку:

```
import pandas as pd
```

Поскольку библиотека может иметь разные версии, полезно посмотреть, какую версию мы используем:

```
print(pd.__version__)
```

Сама настройка отображения выполняется с помощью функции `set_option()`.

Пример использования функции:

```
# Максимальное количество отображаемых столбцов
pd.set_option('display.max_columns', 13)
# Максимальное количество отображаемых строк
pd.set_option('display.max_rows', 10)
# Максимальная ширина столбца
pd.set_option('display.max_colwidth', 45)
# Максимальная ширина отображения
pd.set_option('display.width', 80)
```

### Чтение данных из файла

Перейдем к работе с файлами. Как правило, файлы подгружаются в формате csv (Comma-Separated Values — «значения, разделенные запятыми»). При разборе

функций библиотеки `Pandas` воспользуемся открытым набором данных — Top 100 popular movies from 2003 to 2022 (iMDB).

Рассмотрим описание таблицы, с которой будем работать:

| Variable | Definition | Variable | Definition |
|---|---|---|---|
| **Title** | Название фильма | Stars | Актеры, сыгравшие в фильме |
| **Rating** | Оценка на iMDB | Genre/s | Жанр/ы фильма |
| **Year** | Год выхода фильма | Filming Location | Место съемки фильма |
| **Month** | Месяц выхода фильма | Budget | Бюджет фильма |
| **Certificate** | Возрастной рейтинг | Income | Сборы фильма |
| **Runtime** | Продолжительность | Country of Origin | Страна-производитель фильма |
| **Director/s** | Режиссер(ы) | — | — |

Подгружаем файл:

```
data = pd.read_csv("movies.csv", sep=',')
```

Также можно подгрузить файл с Google-диска:

```
from google.colab import drive
drive.mount('/content/drive')
```

При работе с функцией `read_csv()` указывают:

- Разделитель полей `sep` (по умолчанию ',').
- Кодировку `encoding` (например, 'utf8', в Windows часто бывает 'cp1251').
- Разделитель дробей `delimiter`.
- `usecols` — какие колонки нужно загрузить (бывает, что нужны не все. Особенно актуально, когда данных много).
- `index_col` — колонка-индекс.
- `dtype` — словарь, задающий типы данных для соответствующих полей, и т. д.

У функции `read_csv()` множество параметров. Чтобы посмотреть полный список, можно воспользоваться справкой — знак вопроса и имя функции. Или найти описание на сайте документации.

Посмотрим, какой тип данных мы считали:

```
type(data)
```

В нашем случае тип данных будет: `pandas.core.frame.DataFrame`.

Далее перейдем к просмотру данных. Чтобы просмотреть первые пять строк таблицы, необходимо прописать:

```
data.head()
```

Результат (фрагмент):

| | Title | Rating | Year | Month | Certificate | Runtime | Directors | Stars | Genre |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Avatar: The Way of Water | 8.0 | 2022 | December | PG-13 | 192 | James Cameron | Sam Worthington, Zoe Saldana, Sigourney W... | Action, Adventure, Fantasy |
| 1 | Guillermo del Toro's Pinocchio | 7.8 | 2022 | December | PG | 117 | Guillermo del Toro, Mark Gustafson | Ewan McGregor, David Bradley, Gregory Man... | Animation, Drama, Family |
| 2 | Bullet Train | 7.3 | 2022 | August | R | 127 | David Leitch | Brad Pitt, Joey King, Aaron Taylor Johnso... | Action, Comedy, Thriller |
| 3 | The Banshees of Inisherin | 8.0 | 2022 | November | R | 114 | Martin McDonagh | Colin Farrell, Brendan Gleeson, Kerry Con... | Comedy, Drama |
| 4 | M3gan | NaN | 2022 | January | PG-13 | 102 | Gerard Johnstone | Jenna Davis, Amie Donald, Allison William... | Horror, Sci-Fi, Thriller |

Если необходимо вывести определенное количество строк, можно указать их количество в скобках: `data.head(10)`.

Можно вывести последние пять строк в таблице:

```
data.tail()
```

Результат (фрагмент):

| | Title | Rating | Year | Month | Certificate | Runtime | Directors | Stars | Genre |
|---|---|---|---|---|---|---|---|---|---|
| 1995 | A Tale of Two Sisters | 7.1 | 2003 | June | R | 114 | Jee woon Kim | Lim Soo jung, Yum Jung ah, Kim Kap su, Mo... | Drama, Horror, Mystery |
| 1996 | Lara Croft Tomb Raider: The Cradle of Life | 5.5 | 2003 | July | PG-13 | 117 | Jan de Bont | Angelina Jolie, Gerard Butler, Chris Barr... | Action, Adventure, Fantasy |
| 1997 | Gothika | 5.8 | 2003 | November | R | 98 | Mathieu Kassovitz | Halle Berry, Pen lope Cruz, Robert Downey... | Horror, Mystery, Thriller |
| 1998 | Ong-Bak: The Thai Warrior | 7.1 | 2003 | February | R | 105 | Prachya Pinkaew | Tony Jaa, Phetthai Vongkumlao, Pumwaree Y... | Action, Crime, Thriller |
| 1999 | Open Water | 5.8 | 2003 | August | R | 79 | Chris Kentis | Blanchard Ryan, Daniel Travis, Saul Stein... | Adventure, Drama, Horror |

Мы можем посмотреть случайные `n` строк (например, три):

```
data.sample(n=3)
```

На практике бывает полезно знать размер данных:

```
data.shape
```

Результат работы функции: `(2000, 13)`, где 2000 — количество строк, а 13 — количество столбцов.

Перейдем к описательным статистикам. Чтобы получить общую информацию, воспользуемся методом `describe()`:

```
data.describe()
```

Результат выполнения функции:

| | Rating | Year |
|---|---|---|
| **count** | 1998.000000 | 2000.000000 |
| **mean** | 6.667618 | 2012.500000 |
| **std** | 0.913032 | 5.767723 |
| **min** | 1.900000 | 2003.000000 |
| **25%** | 6.125000 | 2007.750000 |
| **50%** | 6.700000 | 2012.500000 |
| **75%** | 7.300000 | 2017.250000 |
| **max** | 9.600000 | 2022.000000 |

Часто бывает полезно узнать информацию о `DataFrame` — воспользуемся методом `info()`:

```
data.info()
```

Результат выполнения функции:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Title              2000 non-null   object
 1   Rating             1998 non-null   float64
 2   Year               2000 non-null   int64
 3   Month              2000 non-null   object
 4   Certificate        1966 non-null   object
 5   Runtime            2000 non-null   object
 6   Directors          2000 non-null   object
 7   Stars              2000 non-null   object
 8   Genre              2000 non-null   object
 9   Filming_location   2000 non-null   object
 10  Budget             2000 non-null   object
 11  Income             2000 non-null   object
 12  Country_of_origin  2000 non-null   object
dtypes: float64(1), int64(1), object(11)
memory usage: 203.2+ KB
```

После выполнения функции `info()` мы видим количество ненулевых записей, тип колонок.

### Индексация

Теперь поговорим про индексацию. Мы можем обратиться к конкретному элементу, указав индекс и название колонки:

- элемент `data.loc[index, column]`;
- подтаблица `data.loc[list of index, list of columns]`.

Также к элементам `DataFrame` можно обращаться по номеру строк и столбцов:

- элемент `data.iloc[i, j]`;

- подтаблица `data.iloc[list of i, list of j]`.

Получить названия колонок и строк можно с помощью функций:

- `data.columns`,
- `data.index`.

Индекс можно изменять. Для этого воспользуемся методом `set_index()`. По умолчанию метод возвращает новый `DataFrame` с установленным новым индексом. Метод имеет параметр `implace`. Если `implace=True`, будет заменен текущий объект `DataFrame`.

Воспользуемся методом `set_index()` и заменим текущий индекс значением `RunTime`:

```
data.set_index('Runtime', inplace=True)
data.head(3)
```

Результат выполнения (фрагмент):

| Runtime | Title | Rating | Year | Month | Certificate | Directors | Stars | Genre |
|---|---|---|---|---|---|---|---|---|
| 192 | Avatar: The Way of Water | 8.0 | 2022 | December | PG-13 | James Cameron | Sam Worthington, Zoe Saldana, Sigourney W... | Action, Adventure, Fantasy |
| 117 | Guillermo del Toro's Pinocchio | 7.8 | 2022 | December | PG | Guillermo del Toro, Mark Gustafson | Ewan McGregor, David Bradley, Gregory Man... | Animation, Drama, Family |
| 127 | Bullet Train | 7.3 | 2022 | August | R | David Leitch | Brad Pitt, Joey King, Aaron Taylor Johnso... | Action, Comedy, Thriller |

Далее мы заново подгрузим данные из csv-файла и будем работать с первоначальным `DataFrame`.

Рассмотрим, как обращаться к конкретному элементу. Пусть нас интересует значение первой строки колонки с именем `'Stars'`:

```
data.loc[0, 'Stars']
```

Также можно обратиться по номеру. В примере мы обращаемся к первой строке первой колонки:

```
data.iloc[0, 0]
```

***Важно!*** Нумерация индексов начинается с нуля.

Полученные таблицы мы можем транспонировать. **Транспонирование** — замена столбцов на строки. Воспользуемся кодом:

```
data.T.head()
```

Результат выполнения (фрагмент):

| | 0 | 1 | 2 | 3 | 4 | 5 | ... | 1994 |
|---|---|---|---|---|---|---|---|---|
| **Title** | Avatar: The Way of Water | Guillermo del Toro's Pinocchio | Bullet Train | The Banshees of Inisherin | M3gan | Emancipation | ... | In the Cut |
| **Rating** | 8.0 | 7.8 | 7.3 | 8.0 | NaN | 5.9 | ... | 5.4 |
| **Year** | 2022 | 2022 | 2022 | 2022 | 2022 | 2022 | ... | 2003 |
| **Month** | December | December | August | November | January | December | ... | October |
| **Certificate** | PG-13 | PG | R | R | PG-13 | R | ... | R |

Теперь более подробно поговорим про объект `pd.Series`. Каждая колонка в `DataFrame` — это объект `Series`, у этого объекта тоже есть индексы. Чтобы получить всю колонку `'Title'`, нужно указать ее название в квадратных скобках:

```
data['Title'].head()
```

Второй вариант получения колонки:

```
data.Title.head()
```

Первый вариант обращения предпочтительнее.

Чтобы получить строку из данных, можно вызвать метод `loc`, указав индекс строки, например:

```
data.loc[0]
```

Точно так же можно обратиться к конкретному значению:

```
data.loc[0]['Title']
```

### Поиск и фильтрация

Поговорим про поиск и фильтрацию в `DataFrame`. Чтобы отбирать данные по конкретным столбцам, можно указывать эти столбцы. Например:

```
sub_data = data[['Title', 'Rating']]
sub_data.head()
```

Результат выполнения:

| | Title | Rating |
|---|---|---|
| 0 | Avatar: The Way of Water | 8.0 |
| 1 | Guillermo del Toro's Pinocchio | 7.8 |
| 2 | Bullet Train | 7.3 |
| 3 | The Banshees of Inisherin | 8.0 |
| 4 | M3gan | NaN |

Отбор строк осуществляется при обращении к строкам по индексу. Пусть нас интересуют строки `5`, `800`, `1234`:

```
sub_data_row = data.loc[[5, 800, 1234]]
```

Теперь объединим рассмотренный выше отбор строк и столбцов. В примере мы выберем значения строк `5`, `800`, `1234` и значения столбцов `'Title'` и `'Rating'`:

```
data.loc[[5, 800, 1234]][['Title', 'Rating']]
```

Результат выполнения:

| | Title | Rating |
|---|---|---|
| **5** | Emancipation | 5.9 |
| **800** | Interstellar | 8.6 |
| **1234** | Insidious | 6.8 |

Еще один вариант, как получить тот же самый результат:

```
data.loc[[5, 800, 1234], ['Title', 'Rating']]
```

Существуют и другие методы отбора. Например, нас интересуют строки до 5 включительно и колонки с `'Title'` по `'Year'`:

```
data.loc[:5, 'Title':'Year']
```

Результат выполнения:

| | Title | Rating | Year |
|---|---|---|---|
| 0 | Avatar: The Way of Water | 8.0 | 2022 |
| 1 | Guillermo del Toro's Pinocchio | 7.8 | 2022 |
| 2 | Bullet Train | 7.3 | 2022 |
| 3 | The Banshees of Inisherin | 8.0 | 2022 |
| 4 | M3gan | NaN | 2022 |
| 5 | Emancipation | 5.9 | 2022 |

Можно выводить значения по номерам строк и столбцов:

```
data.iloc[[10, 50, 100], [0, 5]]
```

Результат выполнения:

| | Title | Runtime |
|---|---|---|
| 10 | The Menu | 107 |
| 50 | The Wonder | 108 |
| 100 | The Fallout | 96 |

В следующем примере мы отбираем строки до 6 и столбцы с 1 до 4:

```
data.iloc[:6, 1:4]
```

Результат выполнения:

| | Rating | Year | Month |
|---|---|---|---|
| **0** | 8.0 | 2022 | December |
| **1** | 7.8 | 2022 | December |
| **2** | 7.3 | 2022 | August |
| **3** | 8.0 | 2022 | November |
| **4** | NaN | 2022 | January |
| **5** | 5.9 | 2022 | December |

Перейдем к отбору данных по условию. Воспользуемся функцией `unique()`, с помощью которой посмотрим уникальные значения в колонке, например, в `'Month'`:

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

Отберем данные только по фильмам, вышедшим в декабре. Воспользуемся маской:

```
mask = (data['Month'] == 'December')
mask.head()
```

Результат выполнения (`True` показывает, что фильм вышел в декабре):

```
0     True
1     True
2    False
3    False
4    False
Name: Month, dtype: bool
```

Подставим маску в данные и посмотрим первые пять строк:

```
data_neutral = data[mask]
data_neutral.head()
```

Результат выполнения (фрагмент):

| | Title | Rating | Year | Month | Certificate | Runtime | Directors | Stars | Genre |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Avatar: The Way of Water | 8.0 | 2022 | December | PG-13 | 192 | James Cameron | Sam Worthington, Zoe Saldana, Sigourney W... | Action, Adventure, Fantasy |
| **1** | Guillermo del Toro's Pinocchio | 7.8 | 2022 | December | PG | 117 | Guillermo del Toro, Mark Gustafson | Ewan McGregor, David Bradley, Gregory Man... | Animation, Drama, Family |
| **5** | Emancipation | 5.9 | 2022 | December | R | 132 | Antoine Fuqua | Will Smith, Ben Foster, Charmaine Bingwa,... | Action, Thriller |
| **7** | Violent Night | 6.9 | 2022 | December | R | 112 | Tommy Wirkola | David Harbour, John Leguizamo, Beverly D ... | Action, Comedy, Crime |
| **8** | The Whale | 8.2 | 2022 | December | R | 117 | Darren Aronofsky | Brendan Fraser, Sadie Sink, Ty Simpkins, ... | Drama |

Используя подобный подход, можно прописывать более сложные условия. Например, фильм должен выйти в декабре и иметь рейтинг 8 и более:

```
data[(data['Month'] == 'December') & (data['Rating'] >=
8.0)].head()
```

Результат выполнения (фрагмент):

| | Title | Rating | Year | Month | Certificate | Runtime | Directors | Stars | Genre |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Avatar: The Way of Water | 8.0 | 2022 | December | PG-13 | 192 | James Cameron | Sam Worthington, Zoe Saldana, Sigourney W... | Action, Adventure, Fantasy |
| **8** | The Whale | 8.2 | 2022 | December | R | 117 | Darren Aronofsky | Brendan Fraser, Sadie Sink, Ty Simpkins, ... | Drama |
| **51** | Freddy | 8.0 | 2022 | December | NaN | 124 | Shashanka Ghosh | Kartik Aaryan, Alaya F, Karan Pandit, Saj... | Drama, Mystery, Romance |
| **98** | An Action Hero | 8.0 | 2022 | December | NaN | 130 | Anirudh Iyer | Ayushmann Khurrana, Nora Fatehi, Akshay K... | Action, Comedy, Crime |
| **101** | Farha | 8.4 | 2021 | December | TV-14 | 92 | Darin J Sallam | Karam Taher, Ashraf Barhom, Ali Suliman, ... | Drama |

Далее поговорим про фильтрацию по индексам и колонкам с помощью метода `filter()`. Выполним отбор по названию колонок и выведем первые пять строк:

```
data.filter(items=['Title', 'Genre']).head()
```

Результат выполнения:

| | Title | Genre |
|---|---|---|
| **0** | Avatar: The Way of Water | Action, Adventure, Fantasy |
| **1** | Guillermo del Toro's Pinocchio | Animation, Drama, Family |
| **2** | Bullet Train | Action, Comedy, Thriller |
| **3** | The Banshees of Inisherin | Comedy, Drama |
| **4** | M3gan | Horror, Sci-Fi, Thriller |

Отберем колонки, которые содержат `'R'` в названии:

```
data.filter(like='R').head()
```

Результат выполнения:

| | Rating | Runtime |
|---|---|---|
| **0** | 8.0 | 192 |
| **1** | 7.8 | 117 |
| **2** | 7.3 | 127 |
| **3** | 8.0 | 114 |
| **4** | NaN | 102 |

Еще один способ отбора — по регулярному выражению. **Регулярные выражения** — это шаблоны, используемые для сопоставления последовательностей символов в строках.

Чтобы отбирать по индексам, нужно указать `axis=0`. Рассмотрим пример и отберем индексы, заканчивающиеся на 10 или содержащие 5:

```
data.filter(regex='10$|5', axis=0).head()
```

Результат выполнения:

| | Title | Rating | Year | Month | Certificate | Runtime | Directors | Stars | Genre |
|---|---|---|---|---|---|---|---|---|---|
| **5** | Emancipation | 5.9 | 2022 | December | R | 132 | Antoine Fuqua | Will Smith, Ben Foster, Charmaine Bingwa,... | Action, Thriller |
| **10** | The Menu | 7.5 | 2022 | November | R | 107 | Mark Mylod | Ralph Fiennes, Anya Taylor Joy, Nicholas ... | Comedy, Horror, Thriller |
| **15** | Spirited | 6.6 | 2022 | November | PG-13 | 127 | Sean Anders | Will Ferrell, Ryan Reynolds, Octavia Spen... | Comedy, Family, Musical |
| **25** | Don't Worry Darling | 6.2 | 2022 | September | R | 123 | Olivia Wilde | Florence Pugh, Harry Styles, Chris Pine, ... | Drama, Thriller |
| **35** | Prey for the Devil | 5.2 | 2022 | October | PG-13 | 93 | Daniel Stamm | Jacqueline Byers, Debora Zhecheva, Christ... | Horror, Thriller |

### Сортировка

Данные можно сортировать по одному или нескольким столбцам, а также по индексам. Разберем сортировку по индексам. По умолчанию сортировка всегда осуществляется в порядке возрастания:

```
data.sort_index()
```

Результат выполнения (фрагмент):

| | Title | Rating | Year | Month | Certificate | Runtime | Directors | Stars | Genre |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Avatar: The Way of Water | 8.0 | 2022 | December | PG-13 | 192 | James Cameron | Sam Worthington, Zoe Saldana, Sigourney W... | Action, Adventure, Fantasy |
| **1** | Guillermo del Toro's Pinocchio | 7.8 | 2022 | December | PG | 117 | Guillermo del Toro, Mark Gustafson | Ewan McGregor, David Bradley, Gregory Man... | Animation, Drama, Family |
| **2** | Bullet Train | 7.3 | 2022 | August | R | 127 | David Leitch | Brad Pitt, Joey King, Aaron Taylor Johnso... | Action, Comedy, Thriller |
| **3** | The Banshees of Inisherin | 8.0 | 2022 | November | R | 114 | Martin McDonagh | Colin Farrell, Brendan Gleeson, Kerry Con... | Comedy, Drama |
| **4** | M3gan | NaN | 2022 | January | PG-13 | 102 | Gerard Johnstone | Jenna Davis, Amie Donald, Allison William... | Horror, Sci-Fi, Thriller |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1995** | A Tale of Two Sisters | 7.1 | 2003 | June | R | 114 | Jee woon Kim | Lim Soo jung, Yum Jung ah, Kim Kap su, Mo... | Drama, Horror, Mystery |

Изменим порядок сортировки:

```
data.sort_index(ascending=False)
```

Результат выполнения (фрагмент):

| | Title | Rating | Year | Month | Certificate | Runtime | Directors | Stars | Genre |
|---|---|---|---|---|---|---|---|---|---|
| **1999** | Open Water | 5.8 | 2003 | August | R | 79 | Chris Kentis | Blanchard Ryan, Daniel Travis, Saul Stein... | Adventure, Drama, Horror |
| **1998** | Ong-Bak: The Thai Warrior | 7.1 | 2003 | February | R | 105 | Prachya Pinkaew | Tony Jaa, Phetthai Vongkumlao, Pumwaree Y... | Action, Crime, Thriller |
| **1997** | Gothika | 5.8 | 2003 | November | R | 98 | Mathieu Kassovitz | Halle Berry, Pen lope Cruz, Robert Downey... | Horror, Mystery, Thriller |
| **1996** | Lara Croft Tomb Raider: The Cradle of Life | 5.5 | 2003 | July | PG-13 | 117 | Jan de Bont | Angelina Jolie, Gerard Butler, Chris Barr... | Action, Adventure, Fantasy |
| **1995** | A Tale of Two Sisters | 7.1 | 2003 | June | R | 114 | Jee woon Kim | Lim Soo jung, Yum Jung ah, Kim Kap su, Mo... | Drama, Horror, Mystery |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **4** | M3gan | NaN | 2022 | January | PG-13 | 102 | Gerard Johnstone | Jenna Davis, Amie Donald, Allison William... | Horror, Sci-Fi, Thriller |

Можно выполнить сортировку по названию столбцов:

```
data.sort_index(axis=1)
```

Результат выполнения (фрагмент):

| | Budget | Certificate | Country_of_origin | Directors | Filming_location | Genre |
|---|---|---|---|---|---|---|
| **0** | $350,000,000 | PG-13 | United States | James Cameron | New Zealand | Action, Adventure, Fantasy |
| **1** | $35,000,000 | PG | United States, Mexico, France | Guillermo del Toro, Mark Gustafson | USA | Animation, Drama, Family |
| **2** | $85,900,000 | R | Japan, United States | David Leitch | Japan | Action, Comedy, Thriller |
| **3** | Unknown | R | Ireland, United Kingdom, United States | Martin McDonagh | Ireland | Comedy, Drama |
| **4** | Unknown | PG-13 | United States | Gerard Johnstone | New Zealand | Horror, Sci-Fi, Thriller |
| **...** | ... | ... | ... | ... | ... | ... |
| **1995** | Unknown | R | South Korea | Jee woon Kim | South Korea | Drama, Horror, Mystery |

Разберем пример сортировки по конкретным столбцам. Отсортируем данные по столбцу `'Year'`, выведем только колонки `'Title'` и `'Year'`:

```
data.sort_values('Year')[['Title', 'Year']].dropna()
```

Результат выполнения:

| | Title | Year |
|---|---|---|
| **1999** | Open Water | 2003 |
| **1926** | S.W.A.T. | 2003 |
| **1927** | Underworld | 2003 |
| **1928** | The Rundown | 2003 |
| **1929** | Out of Time | 2003 |
| **...** | ... | ... |
| **71** | The Stranger | 2022 |
| **72** | Elvis | 2022 |
| **73** | Weird: The Al Yankovic Story | 2022 |
| **63** | Empire of Light | 2022 |
| **0** | Avatar: The Way of Water | 2022 |

2000 rows × 2 columns

Используем более сложную конструкцию. Отсортируем данные по году и рейтингу:

```
data.sort_values(['Year', 'Rating'])[['Title', 'Year',
'Rating']].dropna()
```

Результат выполнения:

| | Title | Year | Rating |
|---|---|---|---|
| **1924** | The Room | 2003 | 3.6 |
| **1942** | The Cat in the Hat | 2003 | 4.0 |
| **1970** | Spy Kids 3: Game Over | 2003 | 4.3 |
| **1962** | Virgin Territory | 2003 | 4.7 |
| **1987** | Barely Legal | 2003 | 4.7 |
| **...** | ... | ... | ... |
| **22** | Everything Everywhere All at Once | 2022 | 8.1 |
| **8** | The Whale | 2022 | 8.2 |
| **27** | Top Gun: Maverick | 2022 | 8.4 |
| **45** | Kantara | 2022 | 8.5 |
| **84** | Drishyam 2 | 2022 | 8.6 |

1998 rows × 3 columns

### Переименование

Еще одна полезная возможность — переименование колонок. В примере мы явно пропишем новые названия колонок в виде словаря:

```
data.rename(columns={'Title':'Film title', 'Rating': 'iMDB Score'})
```

Результат выполнения:

| | Film title | iMDB Score | Year | Month | Certificate | Runtime | Directors | Stars | Genre |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Avatar: The Way of Water | 8.0 | 2022 | December | PG-13 | 192 | James Cameron | Sam Worthington, Zoe Saldana, Sigourney W... | Action, Adventure, Fantasy |
| 1 | Guillermo del Toro's Pinocchio | 7.8 | 2022 | December | PG | 117 | Guillermo del Toro, Mark Gustafson | Ewan McGregor, David Bradley, Gregory Man... | Animation, Drama, Family |
| 2 | Bullet Train | 7.3 | 2022 | August | R | 127 | David Leitch | Brad Pitt, Joey King, Aaron Taylor Johnso... | Action, Comedy, Thriller |
| 3 | The Banshees of Inisherin | 8.0 | 2022 | November | R | 114 | Martin McDonagh | Colin Farrell, Brendan Gleeson, Kerry Con... | Comedy, Drama |
| 4 | M3gan | NaN | 2022 | January | PG-13 | 102 | Gerard Johnstone | Jenna Davis, Amie Donald, Allison William... | Horror, Sci-Fi, Thriller |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 1995 | A Tale of Two Sisters | 7.1 | 2003 | June | R | 114 | Jee woon Kim | Lim Soo jung, Yum Jung ah, Kim Kap su, Mo... | Drama, Horror, Mystery |

В следующем примере переведем имена колонок в нижний регистр:

```
data.rename(columns=str.lower)
```

Результат выполнения (фрагмент):

| | title | rating | year | month | certificate | runtime | directors | stars | genre |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Avatar: The Way of Water | 8.0 | 2022 | December | PG-13 | 192 | James Cameron | Sam Worthington, Zoe Saldana, Sigourney W... | Action, Adventure, Fantasy |
| 1 | Guillermo del Toro's Pinocchio | 7.8 | 2022 | December | PG | 117 | Guillermo del Toro, Mark Gustafson | Ewan McGregor, David Bradley, Gregory Man... | Animation, Drama, Family |
| 2 | Bullet Train | 7.3 | 2022 | August | R | 127 | David Leitch | Brad Pitt, Joey King, Aaron Taylor Johnso... | Action, Comedy, Thriller |
| 3 | The Banshees of Inisherin | 8.0 | 2022 | November | R | 114 | Martin McDonagh | Colin Farrell, Brendan Gleeson, Kerry Con... | Comedy, Drama |
| 4 | M3gan | NaN | 2022 | January | PG-13 | 102 | Gerard Johnstone | Jenna Davis, Amie Donald, Allison William... | Horror, Sci-Fi, Thriller |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 1995 | A Tale of Two Sisters | 7.1 | 2003 | June | R | 114 | Jee woon Kim | Lim Soo jung, Yum Jung ah, Kim Kap su, Mo... | Drama, Horror, Mystery |

Переименовывать столбцы можно, применив к названию некоторую функцию:

```
data.rename(columns=lambda col: 'film' + '_' + col.lower())
```

Результат выполнения:

| | film_title | film_rating | film_year | film_month | film_certificate | film_runtime | film_directors | film_stars |
|---|---|---|---|---|---|---|---|---|
| 0 | Avatar: The Way of Water | 8.0 | 2022 | December | PG-13 | 192 | James Cameron | Sam Worthington, Zoe Saldana, Sigourney W... |
| 1 | Guillermo del Toro's Pinocchio | 7.8 | 2022 | December | PG | 117 | Guillermo del Toro, Mark Gustafson | Ewan McGregor, David Bradley, Gregory Man... |
| 2 | Bullet Train | 7.3 | 2022 | August | R | 127 | David Leitch | Brad Pitt, Joey King, Aaron Taylor Johnso... |
| 3 | The Banshees of Inisherin | 8.0 | 2022 | November | R | 114 | Martin McDonagh | Colin Farrell, Brendan Gleeson, Kerry Con... |

## 3. Работа с пропусками и операции над данными

Снова воспользуемся датасетом Top 100 popular movies from 2003 to 2022 (iMDB).

### Статистики по признакам

Для расчета частоты событий используют:

- `.sum()` вектора из нулей и единиц = количество единиц;
- `.mean()` вектора из нулей и единиц = доля единиц.

Подсчитаем средний рейтинг фильмов:

```
data['Rating'].mean()
```

Результат выполнения: `6.667617617617617`.

Также мы можем подсчитать количество фильмов по определенным жанрам:

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

Результат выполнения:

```
Action, Adventure, Sci-Fi       92
Animation, Adventure, Comedy    77
Comedy, Drama, Romance          76
Drama                           65
Action, Adventure, Fantasy      59
                                ..
Comedy, Western                  1
Drama, Music, Thriller           1
Action, Fantasy, Sci-Fi          1
Mystery                          1
Adventure, Horror, Mystery       1
Name: Genre, Length: 244, dtype: int64
```

Воспользовавшись расчетом частоты событий и средним значением для вектора из нулей и единиц, мы можем посчитать долю фильмов жанра `'Action, Adventure, Sci-Fi'`:

```
(data['Genre'] == 'Action, Adventure, Sci-Fi').mean()
```

Результат выполнения: `0.046`.

Также, используя эти функции, мы можем получить средний рейтинг фильмов в жанре драма:

```
data[data['Genre'] == 'Drama']['Rating'].mean()
```

Результат выполнения: `7.052307692307692` — рейтинг таких фильмов немного выше 7.

Важно уточнить, что метод `count()` возвращает число заполненных строк:

```
data[data['Genre'] == 'Drama']['Rating'].count()
```

Результат выполнения: `65`.

Далее поговорим про числовые признаки для наших данных: минимальное, максимальное, медианное и среднее значения. Найдем их для колонки `'Rating'`:

```
data['Rating'].min()
data['Rating'].max()
data['Rating'].median()
data['Rating'].mean()
```

Если нужно оценить сразу все признаки:

```
data.min()
data.max()
data.median()
data.mean()
```

### Работа с пропусками и дубликатами

Далее рассмотрим работу с пропусками. Не обязательно все колонки и значения в данных будут заполнены — могут быть пустые или некорректные значения. Поэтому нужно оценить, сколько пропусков в данных. После чего можно сделать вывод, допустимо ли такое количество пропусков или их необходимо чем-то заполнить.

Посмотрим на общее количество пропусков:

```
data.isnull().sum()
```

Количество незаполненных значений мало информативно. Важнее понимать, какую долю составляют пропуски:

```
data.isnull().mean().sort_values(ascending=False)
```

Результат выполнения:

```
Certificate          0.017
Rating               0.001
Title                0.000
Year                 0.000
Month                0.000
                     ...
Genre                0.000
Filming_location     0.000
Budget               0.000
Income               0.000
Country_of_origin    0.000
Length: 13, dtype: float64
```

Следующий шаг — отбросить пропуски. Для этого мы используем метод `dropna()`. Отбросим пропуски и посмотрим долю незаполненных значений в датасете:

```
data.dropna(subset=['Certificate',
'Rating']).isnull().mean()
```

Результат выполнения — доля незаполненных значений по всем колонкам равна нулю:

```
Title               0.0
Rating              0.0
Year                0.0
Month               0.0
Certificate         0.0
                   ...
Genre               0.0
Filming_location    0.0
Budget              0.0
Income              0.0
Country_of_origin   0.0
Length: 13, dtype: float64
```

Отбрасывание пропусков не всегда подходит, так как это может сказаться на итогах исследования. Пропуски можно заполнять с помощью метода `fillna()`.

Чем можно заполнять пропуски:

- Каким-то значением, которое мы считаем «нормальным»:

  ```
  data['Rating'].fillna(value=0)
  ```

  Такой вариант может нас устроить, но если в дальнейшем мы будем проводить исследование рейтинга, значение 0 будет явным выбросом.

- Средним значением или медианой:

  ```
  data['Rating'].fillna(value=data['Rating'].median())
  ```

- Предыдущим или следующим значением:

```
data['Rating'].fillna(method='ffill')
```

Следующий важный момент, о котором стоит поговорить — удаление дубликатов. Дубликаты в данных появляются по разным причинам.

Первое, что стоит сделать, — проверить, как мы получаем данные. Возможно, мы подгружаем несколько файлов, в них есть пересечения, поэтому строки могут дублироваться. Если технические ошибки в сборе данных отсутствуют, можно воспользоваться методом `drop_duplicates()`:

```
data.drop_duplicates(inplace=True)
```

### Применение функции к данным, расчет новых значений

Мы можем создавать некоторые новые признаки, которые нам могут быть удобны. Например, мы хотим разделить фильмы по трем категориям в зависимости от рейтинга. Напишем функцию, которая будет определять категорию фильма:

```
def rating_group(rating):
    if rating >= 8:
        return 'Good'
    if rating <= 6:
        return 'Bad'
    else:
        return 'Medium'
```

Применим написанную функцию к данным с помощью метода `apply()`:

```
data['Rating_group'] = data['Rating'].apply(rating_group)
data['Rating_group'].head()
```

Мы создали новую колонку, в которую прописали группу по рейтингу:

```
0      Good
1    Medium
2    Medium
3      Good
4    Medium
Name: Rating_group, dtype: object
```

Напишем более сложное условие. Мы хотим посмотреть фильмы с рейтингом от 7,5 до 10 и имеющие жанр Драма:

```
data['New_feature'] = data.apply(lambda row:
    1 if 7.5 < row['Rating'] < 10.0 and row['Genre'] ==
'Drama'
    else 0, axis=1)
data[data['New_feature'] == 1].head()
```

Результат выполнения запроса (фрагмент):

| | Title | Rating | Year | Month | Certificate | Runtime | ... | Filming_location | Budget |
|---|---|---|---|---|---|---|---|---|---|
| **8** | The Whale | 8.2 | 2022 | December | R | 117 | ... | USA | Unknown |
| **9** | The Fabelmans | 7.8 | 2022 | November | PG-13 | 151 | ... | USA | $40,000,000 |
| **66** | Aftersun | 7.7 | 2022 | November | R | 102 | ... | Turkey | Unknown |
| **88** | Women Talking | 7.7 | 2022 | January | PG-13 | 104 | ... | Unknown | Unknown |
| **101** | Farha | 8.4 | 2021 | December | TV-14 | 92 | ... | Jordan | Unknown |

Далее рассмотрим работу со строками:

- Строки можно разделять:

```
data['Stars'].str.split(', ', n=3, expand=True)
```

- Можно вычислять длину строки:

```
data['Stars'].str.len()
```

- Можно вывести строки, содержащие определенное значение:

```
data['Stars'].str.contains('Brad Pitt')
```

Заметим, что в данных есть бинарный признак `Rating_group`, который можно закодировать с помощью 0 и 1:

```
data['Rating_group_new'] = data['Rating_group'].map({'Good':
0, 'Medium': 1})

data.head()
```

Чтобы удалить созданные признаки (это необходимо для дальнейшего изложения материала) воспользуемся методом `drop()`:

```
data.drop(['Rating_group', 'New_feature',
'Rating_group_new'], inplace=True, axis=1)
```

### Группировка и агрегация с GroupBy

Поговорим про группировку и агрегацию данных. Под «группировкой» мы подразумеваем процесс, включающий один или несколько из следующих шагов:

- Splitting the data — разбиение на группы.
- Applying a function — применение функции к каждой группе.
- Combining the results — объединение результата.

Рассчитаем среднюю оценку фильмов в каждом месяце:

```
data.groupby('Month')['Rating'].mean()
```

Результат выполнения запроса:

```
Month
2008         6.100000
2014         2.100000
April        6.643802
August       6.576510
December     6.754435
               ...
March        6.594000
May          6.686806
November     6.741923
October      6.703209
September    6.743709
Name: Rating, Length: 14, dtype: float64
```

Написанный код аналогичен SQL-запросу:

```
SELECT
        Month,
        AVG(Rating)
    FROM data
```

```
        GROUP BY
            Month;
```

Разберем разбиение (split). `GroupBy` хранит исходный `DataFrame` (или `Series`) и разбиение на группы, т. е. соответствие «название группы (значение колонки, по которой группируем) — список индексов». Давайте выведем тип:

```
splits = data.groupby('Genre')
print(type(splits))
```

Также мы можем воспользоваться `get_group()`:

```
splits.get_group('Drama')
```

Мы получим тот же результат, что и `data.loc[data.Genre == 'Drama']`. Можно применять несколько функций:

```
splits['Rating'].agg(['mean', 'std', 'count'])
```

Результат выполнения запроса:

| Genre | mean | std | count |
|---|---|---|---|
| Action | 8.500000 | NaN | 1 |
| Action, Adventure | 6.250000 | 0.070711 | 2 |
| Action, Adventure, Biography | 7.000000 | 0.141421 | 2 |
| Action, Adventure, Comedy | 6.330909 | 0.970657 | 55 |
| Action, Adventure, Crime | 6.083333 | 0.740802 | 12 |
| ... | ... | ... | ... |
| Mystery, Thriller | 6.633333 | 1.048173 | 6 |
| Romance, Drama, Family | 9.600000 | NaN | 1 |
| Romance, Sci-Fi, Thriller | 7.000000 | NaN | 1 |
| Sci-Fi, Thriller | 7.300000 | 0.141421 | 2 |
| Thriller | 5.654545 | 0.777642 | 11 |

## 4. Работа с несколькими таблицами

Для соединения нескольких таблиц в Python можно пользоваться функциями `merge()` и `join()`, а также конкатенацией.

Чтобы продемонстрировать работу с указанными функциями, создадим несколько таблиц. В первой таблице будут содержаться идентификаторы, имя и фамилия некоторых людей:

```
raw_data = {
        'user_id': ['1', '2', '3', '4', '5'],
        'first_name': ['Jin', 'Alex', 'Shelly', 'Anna',
'Troy'],
        'last_name': ['Kazama', 'Lewis', 'Tenant',
'Ivanova', 'Brown']}
df_a = pd.DataFrame(raw_data, columns = ['user_id',
'first_name', 'last_name'])
```

Индексы строк будут отличаться на единицу от идентификаторов (`user_id`).

Вторая таблица будет аналогична первой, при этом часть пользователей будет пересекаться с первой таблицей:

```
raw_data = {
        'user_id': ['4', '5', '6', '7', '8'],
        'first_name': ['Anna', 'Troy', 'Rio', 'Terry',
'Peter'],
        'last_name': ['Ivanova', 'Brown', 'Ferdinand',
'Parker', 'Parker']}
df_b = pd.DataFrame(raw_data, columns = ['user_id',
'first_name', 'last_name'])
```

Третья таблица будет представлять некий словарь: `user_id` по всем значениям из первых двух таблиц (и даже двум новым, которого не было ранее) и некий внешний `outer_id`:

```
raw_data = {
        'user_id': ['1', '2', '3', '4', '5', '7', '8', '9',
'10'],
        'outer_id': [223, 1134, 89, 671, 278, 17, 1931, 216,
81]}
df_c = pd.DataFrame(raw_data, columns =
['user_id','outer_id'])
```

Первое, с чего мы начнем — конкатенация. Сделаем конкатенацию по строкам:

```
df_new = pd.concat([df_a, df_b], ignore_index=True)
```

В данном случае произошло слияние двух таблиц:

| | user_id | first_name | last_name |
|---|---|---|---|
| 0 | 1 | Jin | Kazama |
| 1 | 2 | Alex | Lewis |
| 2 | 3 | Shelly | Tenant |
| 3 | 4 | Anna | Ivanova |
| 4 | 5 | Troy | Brown |
| 5 | 4 | Anna | Ivanova |
| 6 | 5 | Troy | Brown |
| 7 | 6 | Rio | Ferdinand |
| 8 | 7 | Terry | Parker |
| 9 | 8 | Peter | Parker |

Если строка по одному разу присутствует в первой таблице и во второй, при конкатенации она дважды включится в получившуюся таблицу.

Конкатенацию можно выполнять и по столбцам:

```
pd.concat([df_a, df_b], axis=1)
```

В получившуюся таблицу вошли три столбца из первой таблицы и три из второй:

| | user_id | first_name | last_name | user_id | first_name | last_name |
|---|---|---|---|---|---|---|
| **0** | 1 | Jin | Kazama | 4 | Anna | Ivanova |
| **1** | 2 | Alex | Lewis | 5 | Troy | Brown |
| **2** | 3 | Shelly | Tenant | 6 | Rio | Ferdinand |
| **3** | 4 | Anna | Ivanova | 7 | Terry | Parker |
| **4** | 5 | Troy | Brown | 8 | Peter | Parker |

Давайте теперь поговорим про функцию `merge()`. В примере функция `merge()` будет выполняться по колонке `user_id`:

```
pd.merge(df_new, df_c, on='user_id')
```

В результате мы к таблице `df_new` добавили значения из словаря `df_c`:

| | user_id | first_name | last_name | outer_id |
|---|---|---|---|---|
| **0** | 1 | Jin | Kazama | 223 |
| **1** | 2 | Alex | Lewis | 1134 |
| **2** | 3 | Shelly | Tenant | 89 |
| **3** | 4 | Anna | Ivanova | 671 |
| **4** | 4 | Anna | Ivanova | 671 |
| **5** | 5 | Troy | Brown | 278 |
| **6** | 5 | Troy | Brown | 278 |
| **7** | 7 | Terry | Parker | 17 |
| **8** | 8 | Peter | Parker | 1931 |

`Megre()` можно выполнять по полю из левой и полю из правой таблиц. В примере названия колонок совпадают:

```
pd.merge(df_new, df_c, left_on='user_id',
right_on='user_id')
```

Результат:

| | user_id | first_name | last_name | outer_id |
|---|---|---|---|---|
| **0** | 1 | Jin | Kazama | 223 |
| **1** | 2 | Alex | Lewis | 1134 |
| **2** | 3 | Shelly | Tenant | 89 |
| **3** | 4 | Anna | Ivanova | 671 |
| **4** | 4 | Anna | Ivanova | 671 |
| **5** | 5 | Troy | Brown | 278 |
| **6** | 5 | Troy | Brown | 278 |
| **7** | 7 | Terry | Parker | 17 |
| **8** | 8 | Peter | Parker | 1931 |

С помощью `merge()` можно осуществлять `OUTER JOIN`. Для этого в блоке `how` мы прописываем «`outer`»:

```
pd.merge(df_a, df_b, on='user_id', how='outer')
```

В результате получаем объединение таблиц `df_a` и `df_b`:

| | user_id | first_name_x | last_name_x | first_name_y | last_name_y |
|---|---|---|---|---|---|
| **0** | 1 | Jin | Kazama | NaN | NaN |
| **1** | 2 | Alex | Lewis | NaN | NaN |
| **2** | 3 | Shelly | Tenant | NaN | NaN |
| **3** | 4 | Anna | Ivanova | Anna | Ivanova |
| **4** | 5 | Troy | Brown | Troy | Brown |
| **5** | 6 | NaN | NaN | Rio | Ferdinand |
| **6** | 7 | NaN | NaN | Terry | Parker |
| **7** | 8 | NaN | NaN | Peter | Parker |

Значение `NaN` указывает, что в первоначальных таблицах не содержались данные.

Аналогично выполняется `INNER JOIN`:

```
pd.merge(df_a, df_b, on='user_id', how='inner')
```

Получилось пересечение всего по двум пользователям:

| | user_id | first_name_x | last_name_x | first_name_y | last_name_y |
|---|---|---|---|---|---|
| **0** | 4 | Anna | Ivanova | Anna | Ivanova |
| **1** | 5 | Troy | Brown | Troy | Brown |

Мы можем проделать `LEFT` и `RIGHT JOIN`. Для `LEFT JOIN` мы к левой таблице присоединяем правую:

```
pd.merge(df_a, df_b, on='user_id', how='left')
```

Результат:

| | user_id | first_name_x | last_name_x | first_name_y | last_name_y |
|---|---|---|---|---|---|
| **0** | 1 | Jin | Kazama | NaN | NaN |
| **1** | 2 | Alex | Lewis | NaN | NaN |
| **2** | 3 | Shelly | Tenant | NaN | NaN |
| **3** | 4 | Anna | Ivanova | Anna | Ivanova |
| **4** | 5 | Troy | Brown | Troy | Brown |

Пример с `RIGHT JOIN`:

```
pd.merge(df_a, df_b, on='user_id', how='right')
```

Получаем обратный результат:

| | user_id | first_name_x | last_name_x | first_name_y | last_name_y |
|---|---|---|---|---|---|
| **0** | 4 | Anna | Ivanova | Anna | Ivanova |
| **1** | 5 | Troy | Brown | Troy | Brown |
| **2** | 6 | NaN | NaN | Rio | Ferdinand |
| **3** | 7 | NaN | NaN | Terry | Parker |
| **4** | 8 | NaN | NaN | Peter | Parker |

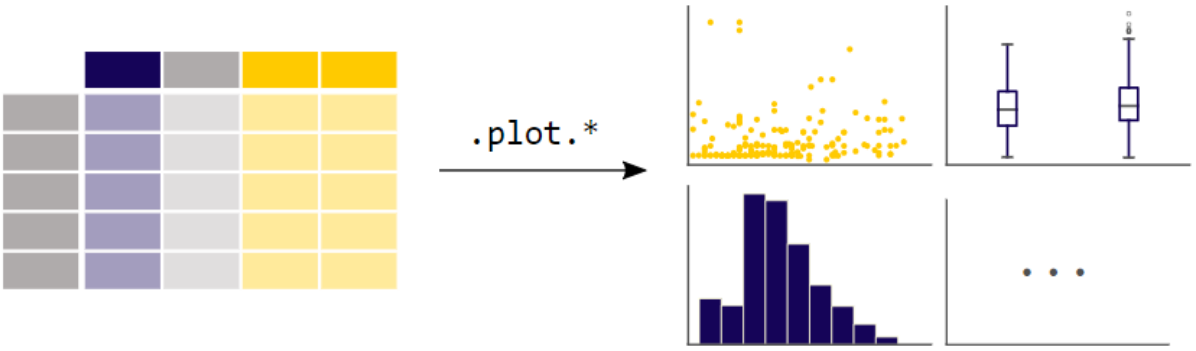
## 5. Построение графиков в Pandas

Рассмотрим построение графиков с использованием библиотеки `Pandas`.

В библиотеке `Pandas` есть набор механизмов, которые позволяют не только представлять данные в виде таблице, но и визуализировать. Важно отметить, что сам `Pandas` не строит графики. Он для этого использует библиотеку MatPlotLib. Но в библиотеке `Pandas` существует ряд методов, которые позволяют нам быстро

визуализировать некоторые элементы, основываясь на таблице `DataFrame` или на объекте типа Series.

**График**



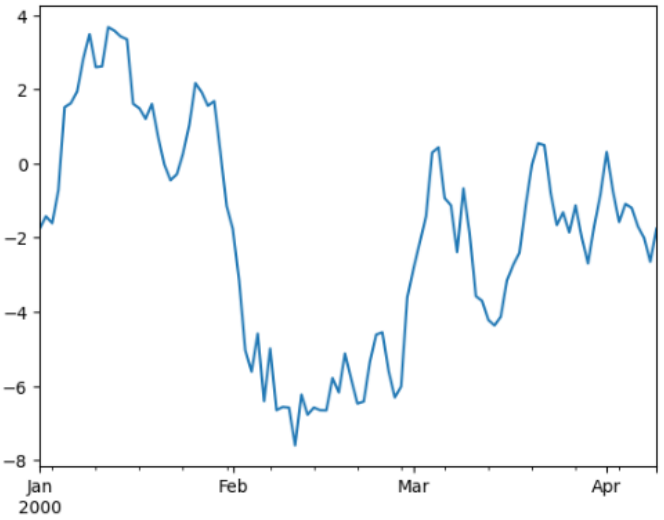

Рассмотрим пример.

```
import pandas as pd

import numpy as np
```

В этом случае NumPy нужен, чтобы создавать рандомные объекты. Создадим объект типа Series, чтобы протестировать визуализацию:

```
ts = pd.Series(np.random.randn(100), index=pd.date_range("1/1/2000", periods=100))

ts = ts.cumsum()

ts.plot()
```

Будем строить график для кумулятивной суммы, чтоб было более наглядно:




Для каждого индекса мы получили сумму в данный период времени.

Создадим `DataFrame`:

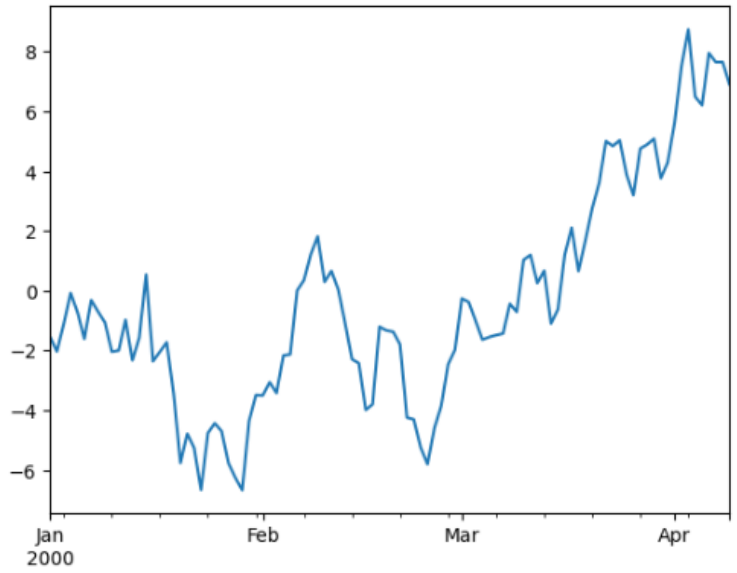
```
df = pd.DataFrame(np.random.randn(100, 4), index=ts.index,
columns=list("ABCD"))

df = df.cumsum()

df.head()
```

>>>

| | A | B | C | D |
|---|---|---|---|---|
| **2000-01-01** | 0.011252 | 0.106307 | 0.559970 | -0.207491 |
| **2000-01-02** | 0.788400 | -0.292333 | 0.478013 | 1.099615 |
| **2000-01-03** | 2.067247 | -0.806673 | -0.119324 | 2.277504 |
| **2000-01-04** | 1.843743 | -0.443222 | -0.251505 | 2.627936 |
| **2000-01-05** | 0.488043 | -0.804468 | -2.331758 | 1.743868 |

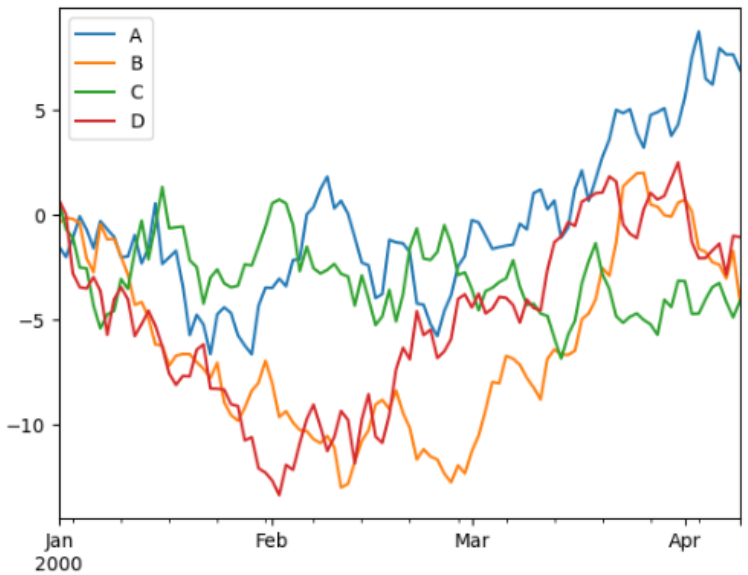
Построим кумулятивную сумму для одной из колонок:

```
df['A'].plot()
```



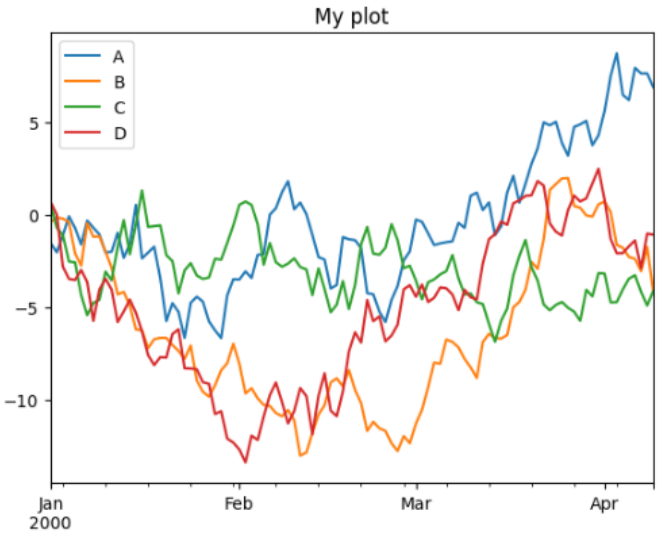

Если хотим получить график, который отображает все колонки, то:

```
df.plot()
```



Зададим название для графика:

```
df.plot(title="My plot")
```



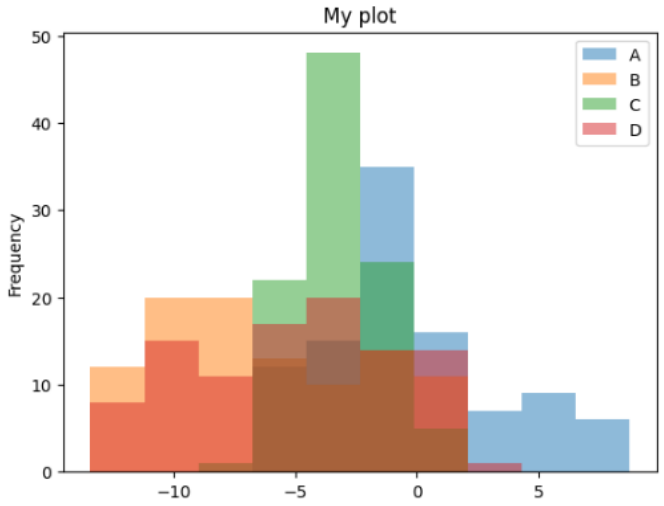

**Гистограмма**

По нашим нарисуем данным гистограмму:

```
df.plot(kind='hist', title="My plot")
```

`Pandas` рисует отдельные гистограммы для каждой колонки. Можно указать прозрачность:

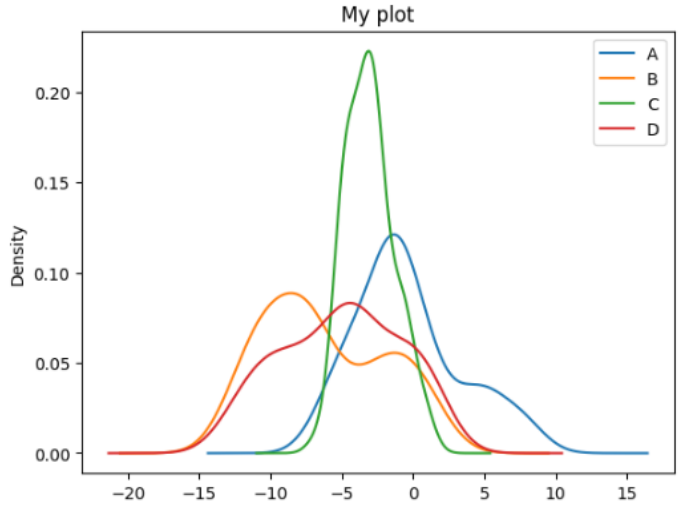
```
df.plot(kind='hist', alpha=.5, title="My plot")
```




### Плотность распределения

Нарисуем плотность распределения:

```
df.plot(kind='density', title="My plot")
```



### Диаграмма рассеяния

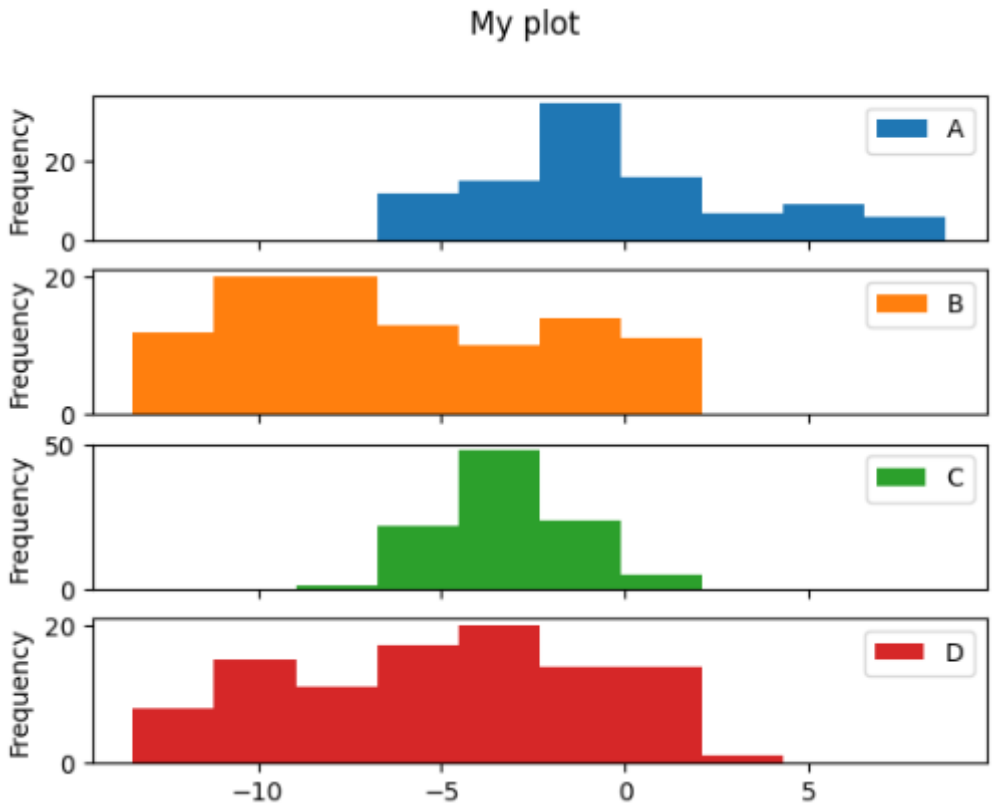
```
df.plot(kind='scatter', x='A', y='B', title="My plot")
```


Обязательные параметры для диаграммы рассеяния — указание двух колонок с данными, по которым строится диаграмма.

### Построение нескольких областей

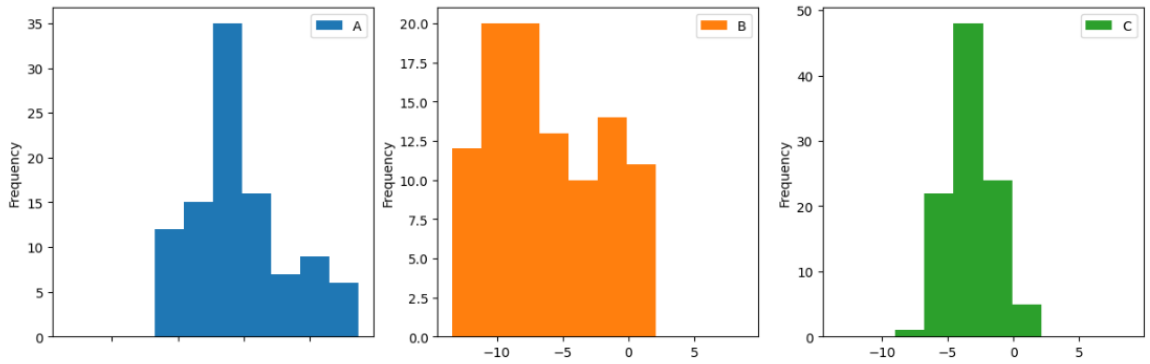
Построение нескольких областей необходимо, когда, например, на одном графике мы хотим увидеть несколько гистограмм, не накладывающихся друг на друга.

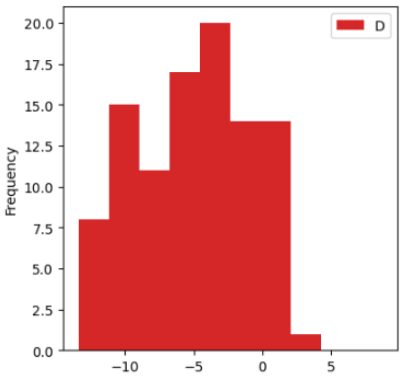
```
df.plot(kind='hist', title="My plot", subplots=True) #, figsize=(6, 6)
```




Еще вариант построения:

```
df.plot(kind='hist', title="My plot", subplots=True, layout=(2, 3),
figsize=(15, 10))
```

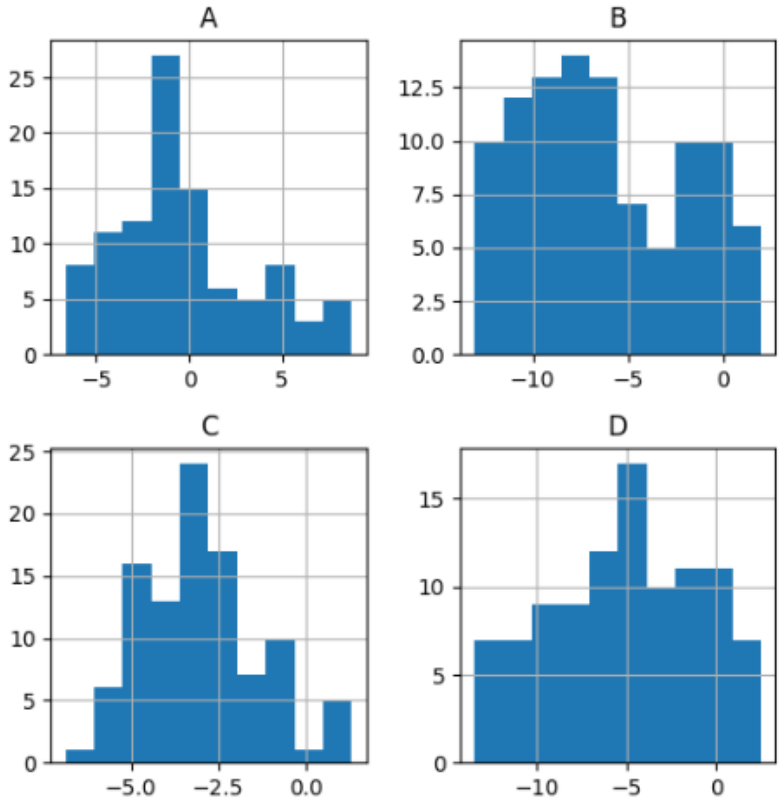





### Метод hist

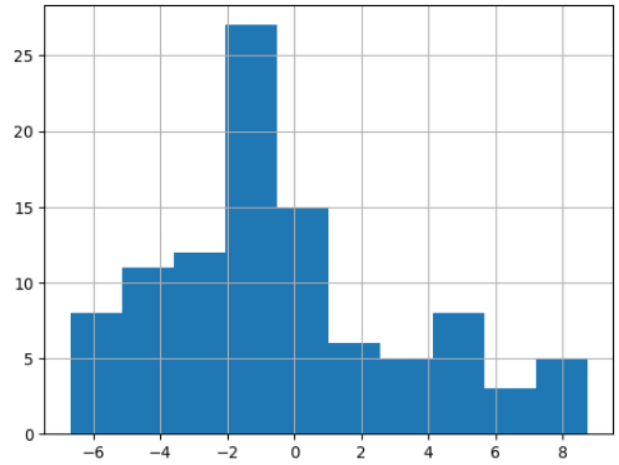
Метод hist также позволяет строить гистограмму, но немного по-другому.

```
df.hist(figsize=(6, 6))
```



Для одной колонки:

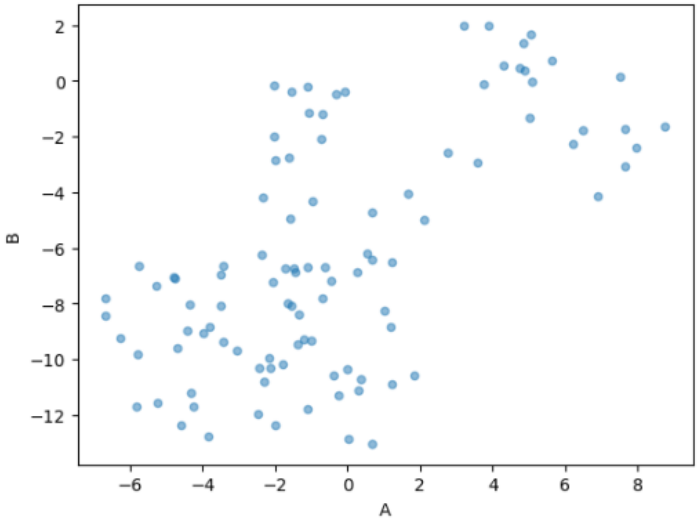
```
df['A'].hist()
```




### Метод scatter

Диаграмму рассеяния можно строить отдельным методом scatter.

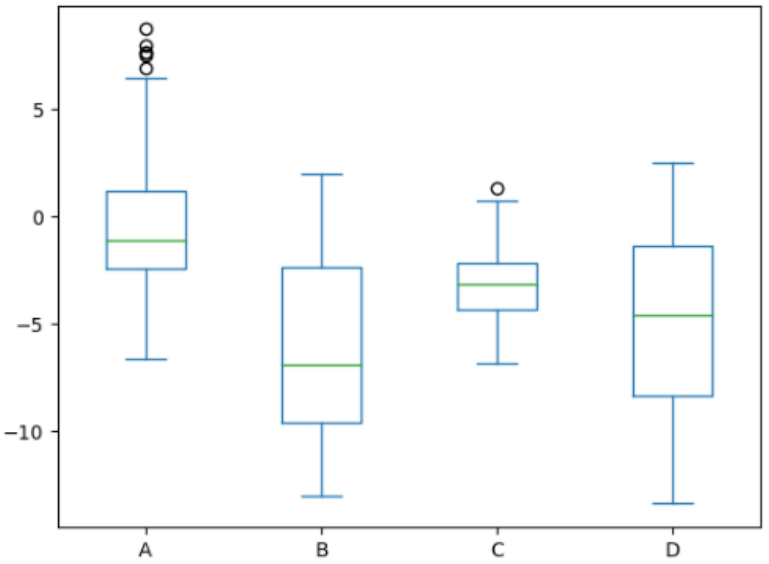
```
df.plot.scatter(x="A", y="B", alpha=0.5)
```



### Метод box

Метод box позволяет отрисовать «ящик с усами».

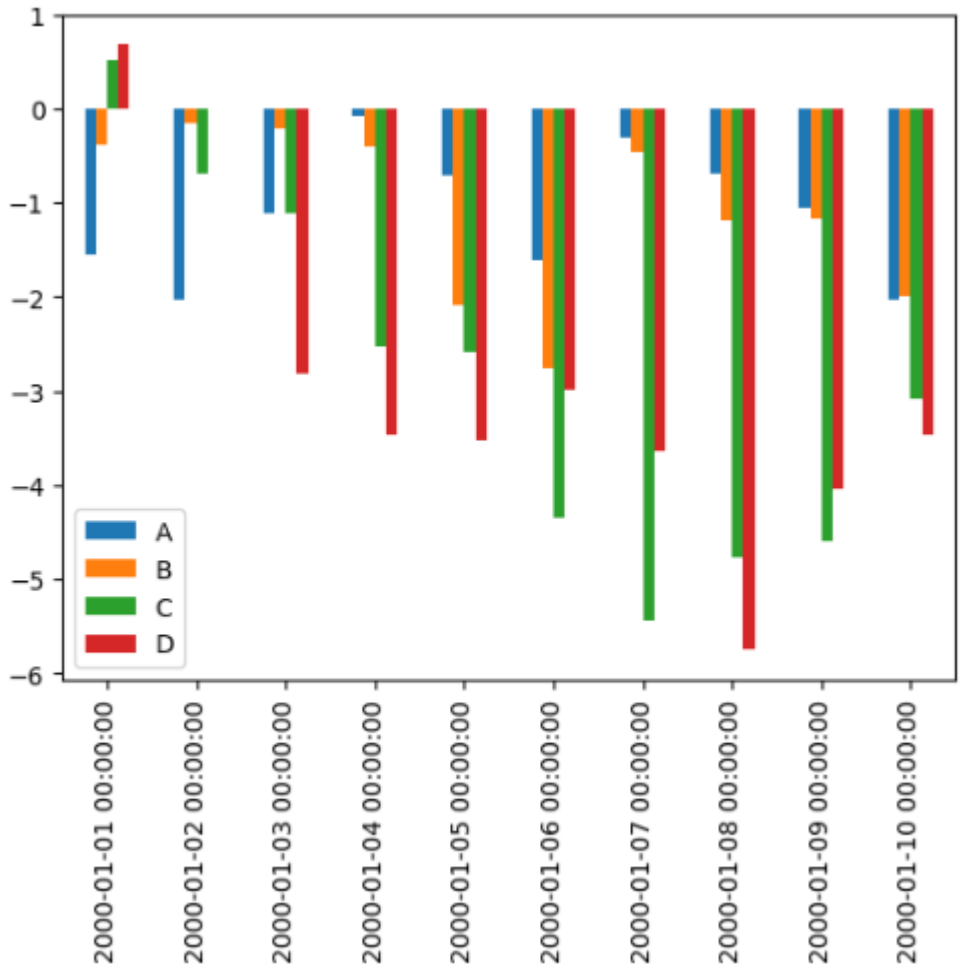
```
df.plot.box() #subplots=True, layout=(2, 2), figsize=(10, 10)
```




«Ящик с усами» показывает нам медианное значение (зеленая линия). «Коробка» ящика — 25% квартиль и 75% квартиль. «Усы» — минимальное и максимальное значения. Отдельные точки — выбросы.

### Столбчатая диаграмма

```
df.iloc[:10,:].plot.bar()
```




Здесь мы взяли первые 10 строк и все столбцы. Данных слишком много, поэтому, если возьмем все строки, график получится нечитаемым:

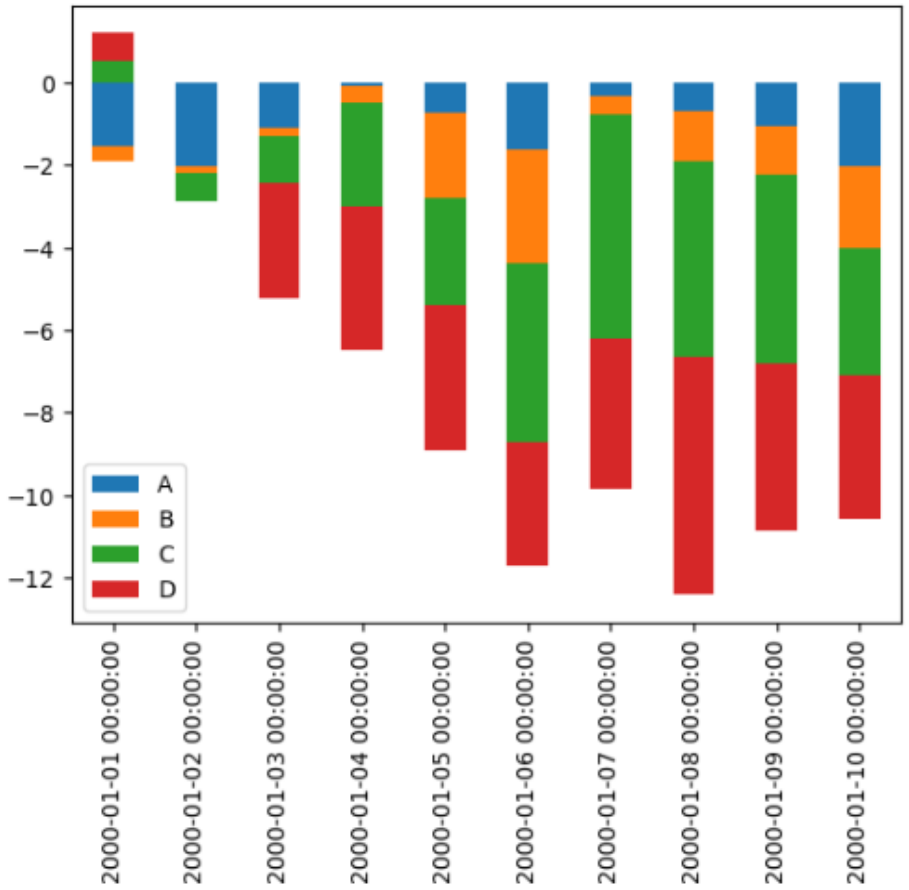
```
df.shape
```

```
>>> (100, 4)
```

В этой диаграмме значения по оси $x$ — значения индекса. По оси $y$ — значения, которые лежат в ячейке `DataFrame`.

Мы можем включить параметр stacked, и тогда все колонки будут в одной прямоугольной области:

```
df.iloc[:10,:].plot.bar(stacked=True)
```

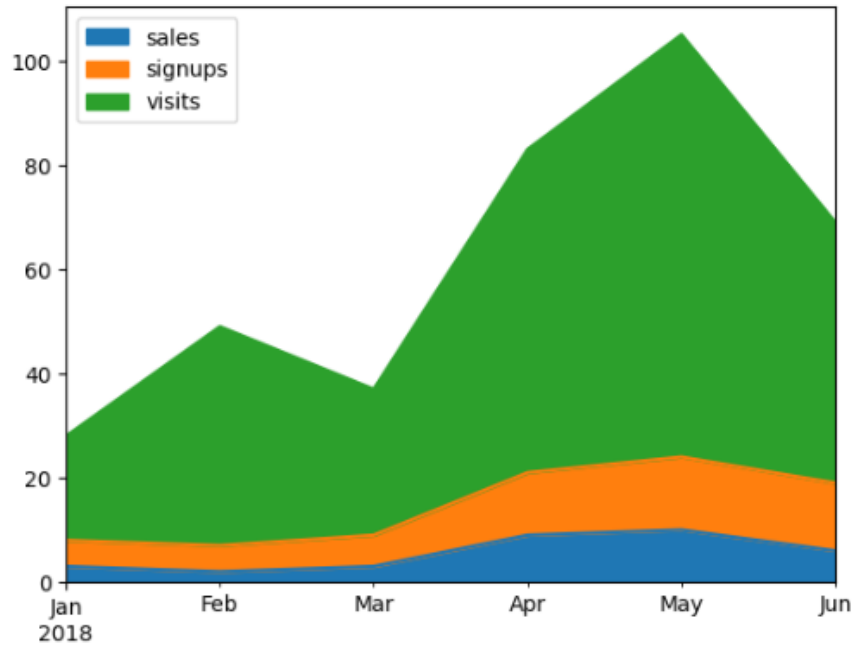



### Зарисовка области

Рассмотрим пример зарисовки на графике некоторой области:

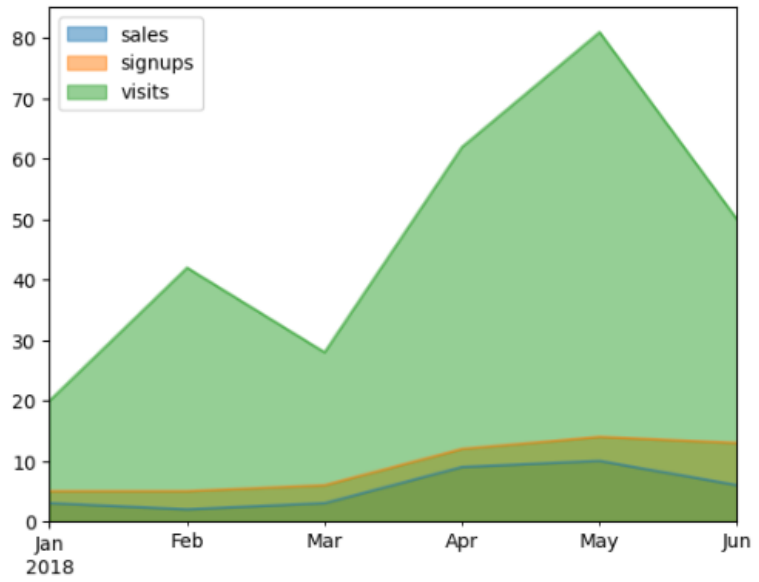
```
df = pd.DataFrame({

    'sales': [3, 2, 3, 9, 10, 6],
```

```
    'signups': [5, 5, 6, 12, 14, 13],

    'visits': [20, 42, 28, 62, 81, 50],

    }, index=pd.date_range(start='2018/01/01', end='2018/07/01',
freq='M'))

df.plot.area()
```



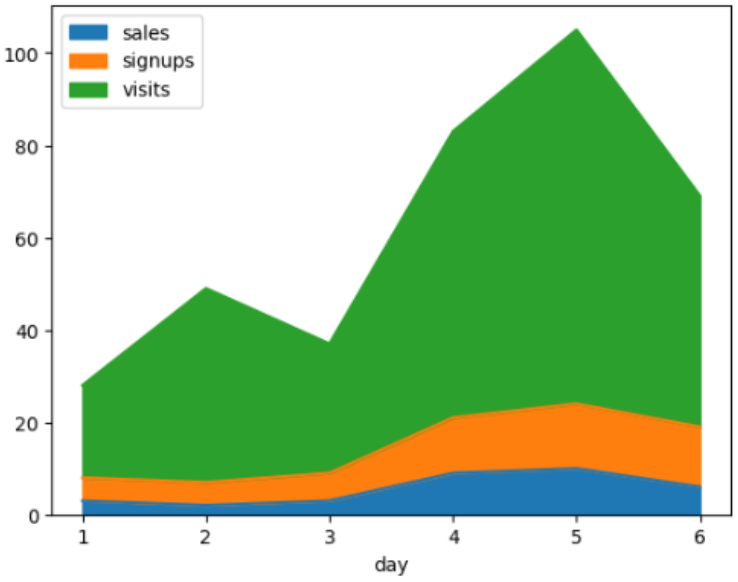

Полупрозрачная зарисовка:

```
df.plot.area(stacked=False)
```




Можно строить только для конкретной колонки:

```
df['day'] = [1, 2, 3, 4, 5, 6]

df.plot.area(x='day')
```



## 6. Основы математической статистики

**Математическая статистика** — это раздел математики, изучающий математические методы сбора, систематизации, обработки и интерпретации результатов наблюдений с целью выявления статистических закономерностей.

Способы сбора информации влияют на смысловое понимание статистических параметров и оценок. Поэтому математическая статистика нужна не только, чтобы описывать и сравнивать данные в ходе анализа, но и для интерпретации получаемых результатов.

**Для чего нужна математическая статистика:**

- Правильный сбор и обработка анализируемых данных.
- Описание свойств и явлений, содержащихся в больших массивах данных.
- Визуализация и систематизация данных.
- Получение выводов по наблюдаемым данным.
- Решение задач прогнозирования и проверки статистических гипотез.
- Оценка отклонений результатов прогнозирования от фактических наблюдений.

### Структурирование данных

Данные поступают к нам из многочисленных источников. Ими могут быть показания счетчиков, результаты измерения температуры, биохимического состава крови. Это могут быть события: посещаемость сайта, загруженность клиентского зала. Также это может быть текстовая или графическая информация: статьи, фото или видеосъемки.

Значительная часть поступающих данных обычно не структурирована. Например, тексты состоят из последовательности словарных и несловарных символов, часто разбиты на разделы, подразделы и т. д., но при этом лишены четкой структуры, которая позволяла бы сравнивать и анализировать их. Поэтому первичные данные обычно называют сырыми данными.

Основная задача науки о данных — перерабатывать поток сырых данных в структурированную форму, имеющую практическое значение. Одна из наиболее часто встречающихся форм структурированных данных — таблица со строками и столбцами. Иногда можно увидеть графовые структуры или временные ряды.

**Типы структурированных данных:**

- Числовые:
    - Непрерывные (скорость ветра, продолжительность времени, концентрация вещества).
    - Дискретные (количество возникновений события, численность населения).
- Категориальные (тип экрана телевизора, название города). Обычно представлены в виде текста.

### Статистические свойства данных

Для описания измеряемых или количественных данных обычно используются числовые признаки — типичные значения для некоторого признака.

**Среднее значение (mean)** — сумма всех значений, деленная на число значений:

$$\text{среднее } = \bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

**Медиана (median)** — число, расположенное в отсортированном списке данных ровно посередине. Медиана зависит только от отсортированных данных в середине.

**Выброс (outlier)** — значение данных, которое сильно отличается от большинства данных.

По сравнению с медианой среднее значение намного более чувствительно к данным. Чаще всего медиана — наиболее подходящая метрика.

**Пример**. Мы хотим рассмотреть типичные доходы жителей городов A и B. Если в одном из городов живет один из представителей списка Forbs, то при сравнении средних значений доходов в этих городах будут совершенно разные результаты.

Медиана называется **робастной** оценкой центрального положения, поскольку она не находится под влиянием выбросов, которые могут исказить результаты.

Когда выбросы являются результатом неправильных данных, среднее значение будет показывать плохую оценку центрального положения, тогда как медиана будет по-прежнему допустимой.

Центральное положение не единственная размерность в обобщении признака.

**Дисперсия (variance)** — сумма квадратичных отклонений от среднего, деленная на $n - 1$, где $n$ — число значений данных:

$$s^2 = \frac{\sum(x-\bar{x})^2}{n-1}.$$

Дисперсия показывает, сгруппировано ли значение данных плотно или же они разбросаны.

**Стандартное отклонение (standard deviation)** — квадратный корень из дисперсии:

$$s = \sqrt{\text{Дисперсия}}$$

Стандартное отклонение интерпретируется намного проще, чем дисперсия, поскольку оно находится на той же шкале измерений, что и исходные данные. Дисперсия и стандартное отклонение чувствительны к выбросам больше всего, поскольку они основаны на квадратичных отклонениях.

Другой подход к оцениванию дисперсии основан на рассмотрении разброса сортированных данных.

**Элементарная мера** — это размах, разница между самым крупным и самым малым числом. Однако он чрезвычайно чувствителен к выбросам и не очень полезен в качестве общей меры дисперсности данных.

Во избежание этой чувствительности существуют такие характеристики как процентиль.

**Процентиль (percentile)** — значение, отражающее, что $P$ процент значений принимает данное значение или меньшее:

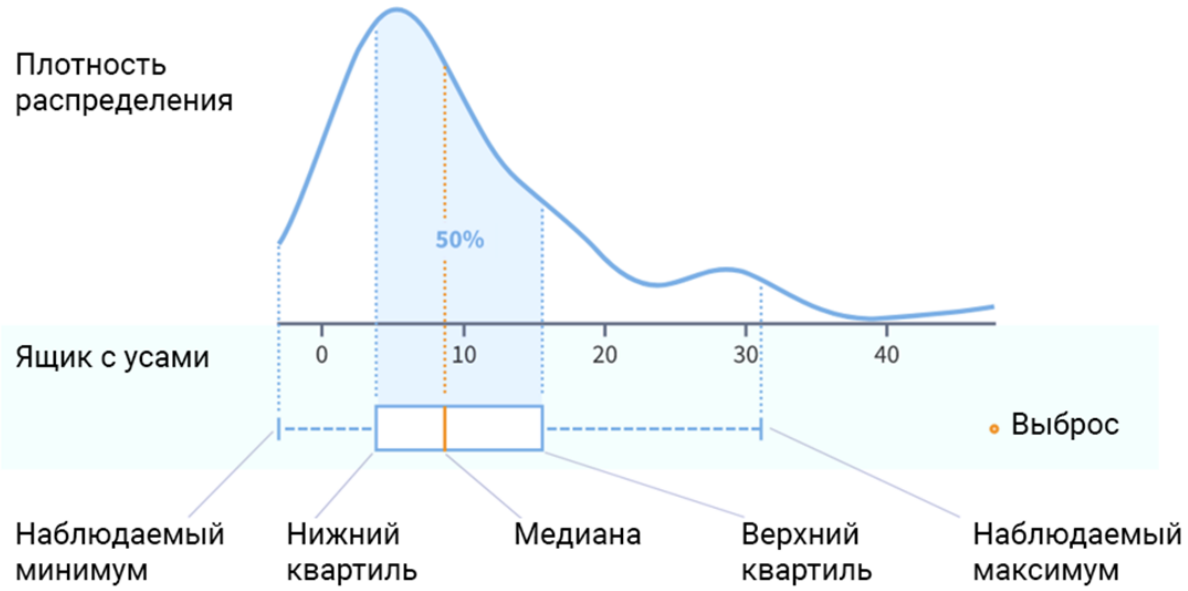
$$P = (1 - w)x_{(j)} + wx_{(j+1)}.$$

Медиана — это то же самое, что и 50-й процентиль.

**Межквартильный размах** (interquartile range, IQR) — разница между 75-м и 25-м процентилями.

Все описанные свойства могут быть представлены в виде графических диаграмм.

**Плотность распределения** — сглаженная версия гистограммы:



Анализ данных во многих исследованиях предусматривает изучение зависимости между числовыми переменными — корреляцию.

**Коэффициент корреляции** (correlation coefficient) — метрика, измеряющая степень, с которой числовые переменные $X$ и $Y$ связаны друг с другом (в диапазоне от -1 до 1):

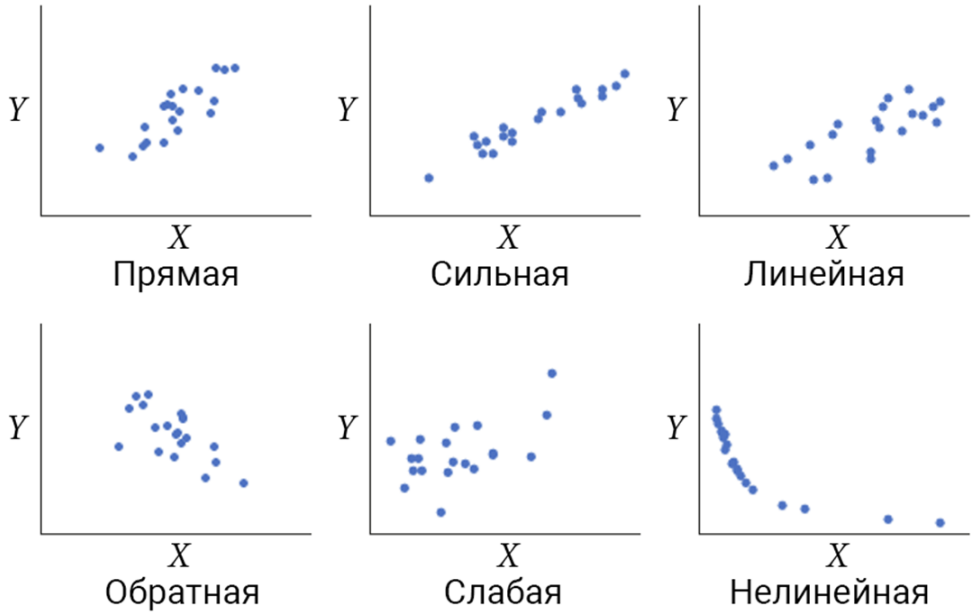
$$r = \frac{\sum_{i=1}^{n}\left(x_i-\bar{x}\right)\left(y_i-\bar{y}\right)}{(n-1)s_x s_y}$$

Принято говорить, что переменные $X$ и $Y$ коррелируют **положительно** или **прямо**, если большие значения $X$ сопровождаются высокими значениями $Y$, а низкие значения $X$ сопровождаются низкими значениями $Y$.

Если высокие значения $X$ сопровождаются низкими значениями $Y$, и наоборот, то переменные коррелируют **отрицательно** или **обратно**.

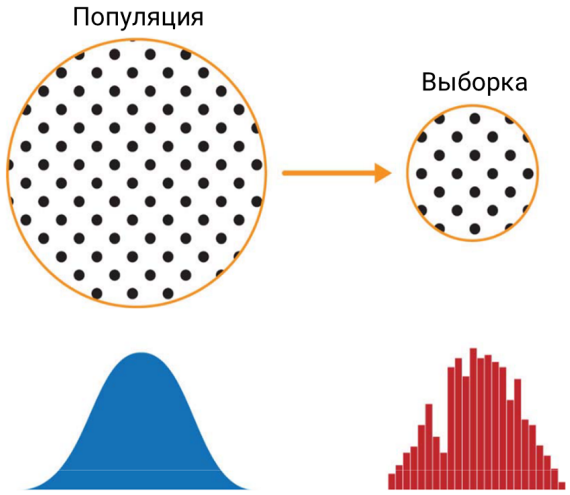
Чем ближе данные расположены друг к другу и соответствуют прямой $y = x$, тем сильнее степень связи.

Также зависимость может быть линейной или нелинейной:



### Выборка данных

Еще одно важное понятие математической статистики — выборка. Популярное мнение ошибочно утверждает, что эра больших данных сводит на нет потребность в выборке. Но на самом деле быстрое распространение данных переменного качества и релевантности только укрепляет потребность в выборке, как в инструменте эффективной работы с разнообразными данными и минимизацией смещения. Представленная схема подкрепляет этот факт:




Слева представлена популяция, которая в статистике предположительно подчиняется базовому, но при этом неизвестному закону. Единственное, что имеется, это выборка данных, и ее эмпирическое распределение (справа). Чтобы попасть из левой стороны в правую, используется процедура отбора (оранжевая стрелка).

Традиционная статистика главным образом сосредоточена на левой стороне, используя теорию, основанную на серьезных допущениях о популяции. Современная статистика переместилась в правую сторону, где такие допущения не нужны.

Популяция часто именуется **генеральной совокупностью**. Выборка — подмножество данных из этой генеральной совокупности.

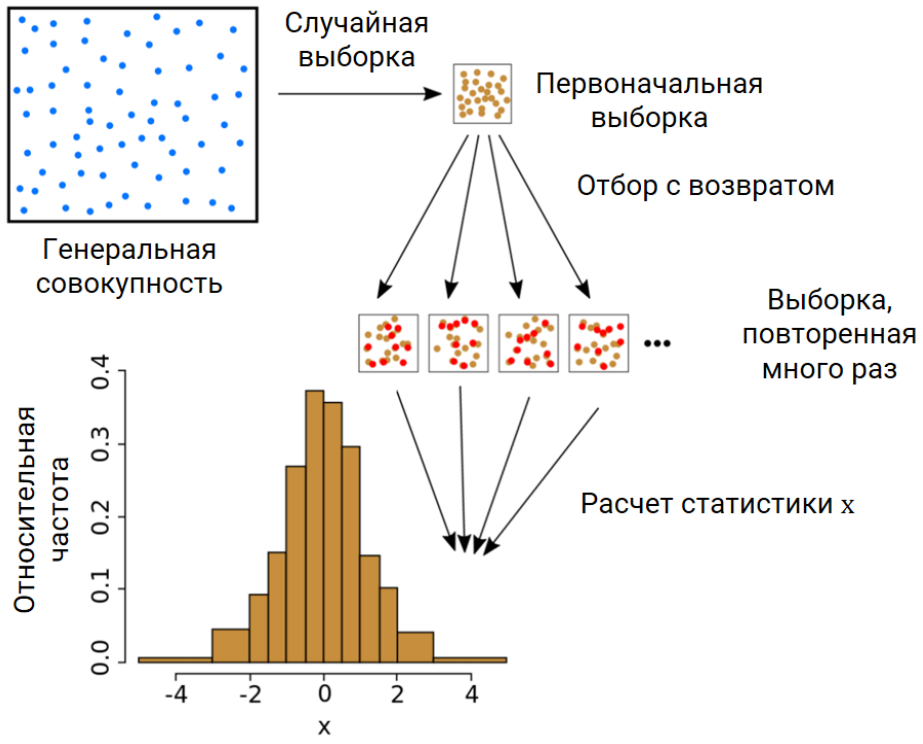
Главная задача процедуры отбора данных в выборку заключается в том, чтобы исследуемые выборки могли дать нам как можно более точное представление о генеральной совокупности.

Выборка считается **репрезентативной**, если у каждого элемента генеральной совокупности равные шансы попасть в выборку.

Качество данных часто имеет большее значение, чем их количество, когда выполняется оценка, либо создается модель на основе выборки.

### Бутстрап (bootstrapping)

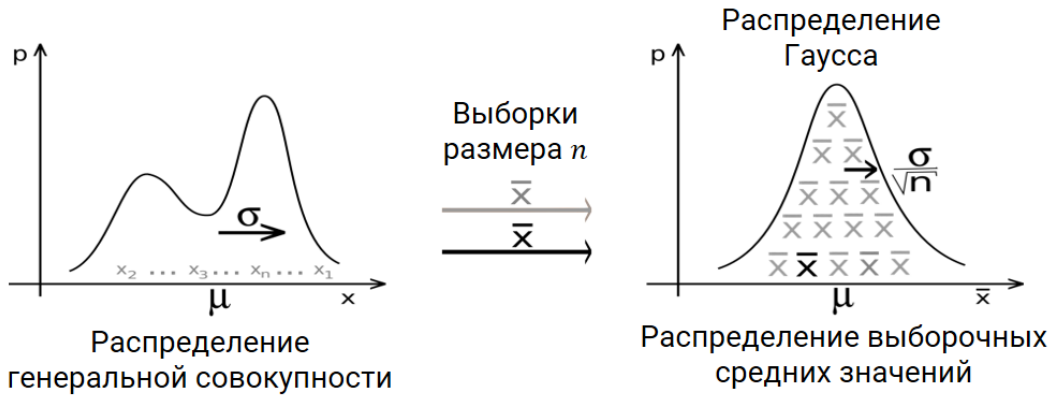
Один из простых и эффективных способов оценивания выборочного распределения статистической величины или модельных параметров — извлекать дополнительные выборки с возвратом из самой выборки и перевычислять статистику или модель для каждой повторной выборки. Процедура называется **бутстрапом** и она не сопряжена с какими-либо допущениями о том, что данные или выборочная статистика нормально распределены.




Такой подход позволяет получить некоторую гипотетическую популяцию, из которой затем можно извлекать выборки в целях оценивания выборочного распределения.

### Центральная предельная теорема

Важно отметить, что если имеются некоторые достаточно крупные выборки, распределение которых не сильно отклоняется от нормального, то средние значения, извлеченные из этих выборок, будут напоминать знакомую колоколообразную нормальную кривую, даже если исходная популяция не является нормально распределенной. Это явление называется **центральной предельной теоремой**.




### Доверительный интервал

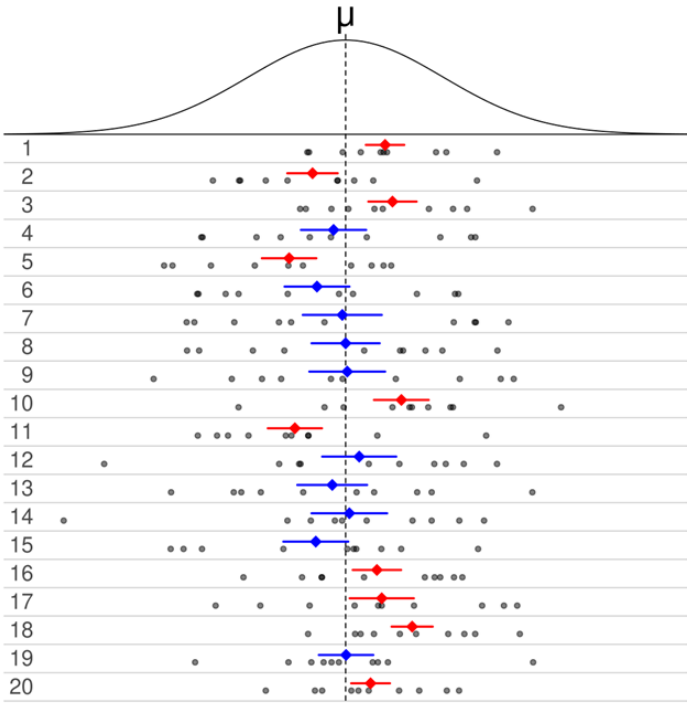
Сама центральная предельная теорема не занимает какое-то особое место в практике науки о данных, однако ее значение полезно для понимания различных статистик. Например, доверительный интервал.

Человеку естественным образом свойственно избегать неопределенности. Ученые крайне редко говорят «я не знаю». Аналитики, признавая неопределенности, вынуждены опираться на точечные оценки. Поэтому они зачастую представляют оценку не единственным числом, а диапазоном.

Доверительные интервалы делают это способом, который берет свое начало в статистических принципах отбора. Доверительные интервалы всегда сопровождаются уровнем покрытия, выраженным высоким процентом, например, 90% или 95%.

90%-ый доверительный интервал можно представить следующим образом. Это интервал, в который попадают 90% значений выборочных статистик, например, полученных с помощью бутстрапа.

Доверительный интервал для среднего значения может быть рассчитан следующим образом:

Несмотря на то, что статистические свойства точечные оценки, благодаря существованию доверительных интервалов, мы можем противодействовать этой тенденции.

## Дополнительные материалы для самостоятельного изучения

1. `Pandas` documentation — `Pandas` 1.5.3 documentation
2. Comparison with other tools — `Pandas` 2.0.2 documentation (pydata.org)
3. Comprehensive Guide to Grouping and Aggregating with `Pandas` — Practical Business Python (pbpython.com)
4. Наглядная шпаргалка по операциям с DataFrame в `Pandas` для data wrangling и не только (tproger.ru)