**Визуализация данных в Python**

***Для наглядности настоятельно рекомендуется рассмотреть подробные примеры в \*.ipynb-файле к занятию. Если будут проблемы, я предупреждал :)***

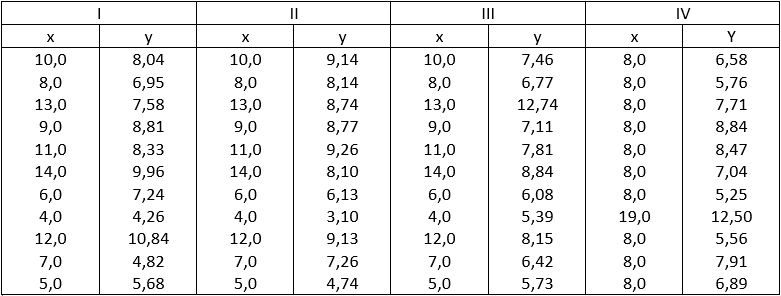
Темы на сегодня:

* Зачем нужна визуализация?
* Библиотеки Matplotlib и Seaborn
* Библиотека Plotly

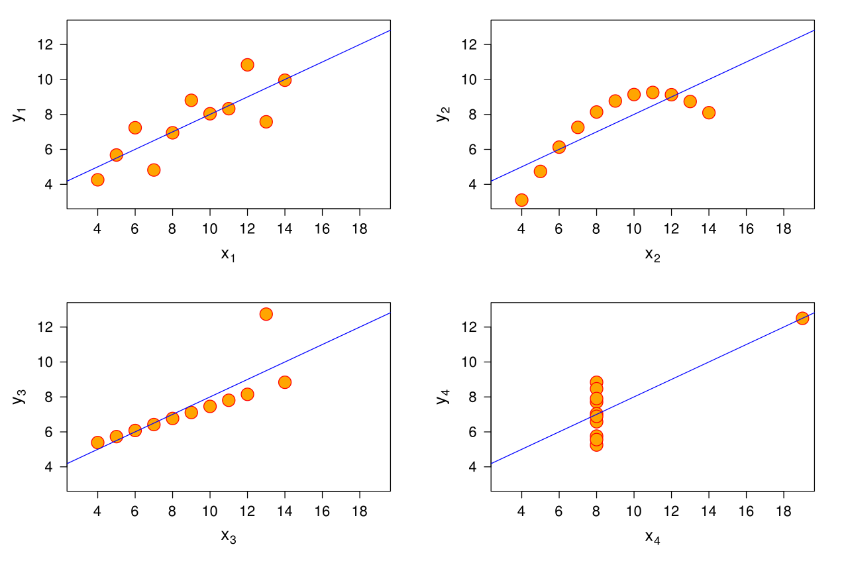
ЗАЧЕМ НУЖНА ВИЗУАЛИЗАЦИЯ?

Уже было изучено достаточно инструментов работы с данными, методов обработки, преобразования данных. Однако, главная задача – сделать выводы по-прежнему всецело зависит от аналитика, который должен увидеть закономерности. Табличные данные – хорошо, но не наглядно. Именно поэтому данные визуализируют: представляют в графическом виде, что обеспечивает наиболее эффективную работу по их изучению.

А ещё потому, что данные могут быть обманчивы. Ярким примером является т. н. «квартет Энскомба» – 4 набора данных, у которых имеются одинаковые статистические показатели: среднее значение, дисперсия, корреляция и т.д.



При этом у них абсолютно разные графики:



Нужно понимать:

* Методов визуализации существует очень много. Все их знать не требуется, но полезно знать, какие и когда следует применять.
* Визуализация должна быть простой: не нужно перегружать её эффектами (тени, 3D и т.д.): они делают её менее наглядной.
* Хорошая визуализация не требует дополнительных подписей / цифр
* Гистограмма ≠ столбчатая диаграмма. Гистограмма используется для отображения распределений

Применительно к Python существует много библиотек:

* matplotlib
* seaborn рассмотрим далее
* plotly
* ggplot – порт одноимённой библиотеки из языка R
* bokeh для интерактивной визуализации,
* pygal сложны для изучения с нуля
* и т.д.

БИБЛИОТЕКИ MATPLOTLIB И SEABORN

Библиотека Matplotlib – первая и самая популярная библиотека для визуализации на Python. По умолчанию входит в пакет Anaconda. При помощи этой библиотеки возможно создать любые представления, а также очень гибко их настроить. Недостатком является то, что для этого понадобится большое количество кода. Для пользователя самым необходимым модулем библиотеки является модуль **pyplot**, который специализируется на готовых решениях и базовых параметрах.

**Пример вызова:**

*import matplotlib.pyplot as plt*

Методы построения типичных визуализаций:

* .plot(набор\_значений\_1, набор\_значений\_2) – линейный график
* .scatter(набор\_значений\_1, набор\_значений\_2) – точечная диаграмма
* .bar(набор\_значений\_1, набор\_значений\_2) – столбчатая диаграмма
* .barh(набор\_значений\_1, набор\_значений\_2) – столбчатая диаграмма в горизонтальном виде
* .hist(набор\_значений\_1, bins=количество\_групп\_распределения) – гистограмма
* .pie(набор\_значений\_1) – круговая диаграмма
* .boxplot(набор\_значений\_1) – диаграмма размаха («ящик с усами»)

Для отображения графика используется метод **.show()**. В Jupyter Notebook вызывать этот метод необязательно.

Базовую настройку графиков можно произвести с помощью следующих методов:

* .xlabel(Название\_оси) – указать название оси абсцисс
* .ylabel(Название\_оси) – указать название оси ординат
* .title(Название) – указать название графика
* .grid() – добавить сетку графика
* .text(абсцисса, ордината, текст) – добавить текст в нужном месте
* .figure(figsize=(ширина, высота)) – изменить размер графика (в дюймах)

Кроме того, можно сохранить график в файл с помощью метода **.savefig(путь\_к\_файлу)**. Сохранять можно в любых популярных форматах изображений: \*.jpg, \*.png и т.д.

**Важно:** библиотека Pandas предоставляет для датафреймов собственный метод **.plot()**. Его параметры:

* kind – тип графика
* rot – уровень наклона подписи к оси абсцисс
* stacked – если True, что график строится с накоплением

Этот метод также позволяет строить графики. Но на самом деле он является своеобразной прослойкой между Pandas и Matplotlib, поскольку именно последний занимается отрисовкой графиков.

Библиотека Seaborn – библиотека визуализации, основанная на Matplotlib. Создана для наиболее простого построения визуализаций.

**Пример вызова:**

*import seaborn as sns*

Позволяет строить красивые графики при помощи небольшого количества кода. В том числе и сложные. Некоторые методы:

* .pairplot(наборы\_данных) – набор графиков с отображением взаимных связей
* .distplot(набор\_данных) – гистограмма + [ядерная оценка плотности](https://ru.wikipedia.org/wiki/%D0%AF%D0%B4%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BE%D1%86%D0%B5%D0%BD%D0%BA%D0%B0_%D0%BF%D0%BB%D0%BE%D1%82%D0%BD%D0%BE%D1%81%D1%82%D0%B8)
* .jointplot() – гибрид точечной диаграммы и гистограммы
* .heatmap() – тепловая карта – распределение численного признака по 2 категориональным.

Однако, если стандартное отображение по каким-то причинам не удовлетворяет требованиям, то для настройки необходимо обращаться в тонкости Matplotlib.

*Документация по Matplotlib:* [*https://matplotlib.org/tutorials/index.html*](https://matplotlib.org/tutorials/index.html)

*Документация по Seaborn:* [*https://seaborn.pydata.org/tutorial.html*](https://seaborn.pydata.org/tutorial.html)

БИБЛИОТЕКА PLOTLY

Библиотека Plotly представляет собой иной подход к визуализации на Python, отличный от Matplotlib. Изначально это был онлайн-сервис для создания и публикации графиков. Имеет офлайн-режим: без регистрации на сервере Plotly, а также без отправки графиков на сервер. Переписана на Python, изначально реализована на языке JavaScript (plotly.js), которая основана на библиотеке визуализации D3.js. По умолчанию не входит в Anaconda, устанавливается из Python:

*pip install plotly==4.4.1* *# Актуальную версию необходимо уточнять*

Для вызова требуется следующая последовательность действий:

**Пример вызова:**

*from plotly.offline import init\_notebook\_mode, iplot*

*import plotly*

*import plotly.graph\_objs as go*

*init\_notebook\_mode(connected=True)*

В Plotly все строится на объекте Figure, который состоит из данных (массив линий, которые в библиотеке называются Traces) и оформления/стиля, за который отвечает объект Layout. В простых случаях можно вызвать функцию iplot() от массива Traces.

**Пример:**

*trace0 = go.Scatter( # Выбирается нужный тип графика*

*x=years\_df.index, # Задаётся массив значений по абсциссе*

*y=years\_df.Global\_Sales, # Аналогично для ординат*

*name='Global Sales' # Имя линии для легенды*

*)*

*trace1 = go.Scatter(*

*x=years\_df.index,*

*y=years\_df.Number\_of\_Games,*

*name='Number of games released'*

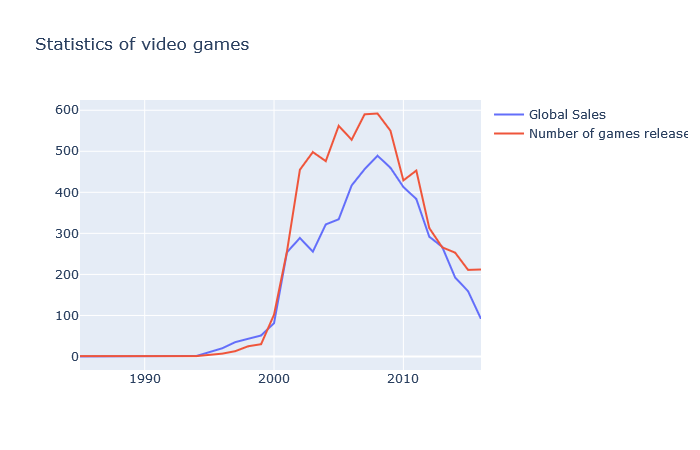
*)*

*traces = [trace0, trace1] # Массив линий*

*layout = {'title': 'Statistics of video games'} # Тут наводим красоту*

*fig = go.Figure(data=traces, layout=layout) # Формируем график*

*iplot(fig) # Отрисовываем*

**

Библиотека работает медленнее, чем Matplotlib, но пользуется спросом ввиду красоты отображения, а также интерактивности. Для новичка поначалу может показаться сложной.

*Документация по Plotly:* [*https://plot.ly/python/*](https://plot.ly/python/)