Оглавление

[Введение 4](#_Toc26615640)

[Библиотеки для визуализации данных 5](#_Toc26615641)

[Matplotlib 5](#_Toc26615642)

[Seaborn 6](#_Toc26615643)

[Ggplot 6](#_Toc26615644)

[Bokeh 7](#_Toc26615645)

[Pygal 8](#_Toc26615646)

[Plotly 8](#_Toc26615647)

[Geoplotlib 9](#_Toc26615648)

[Gleam 10](#_Toc26615649)

[Missingno 11](#_Toc26615650)

[Leather 12](#_Toc26615651)

[Примеры реализации 13](#_Toc26615652)

[Установка Python и настройка окружения 13](#_Toc26615653)

[Matplotlib 15](#_Toc26615654)

[Pygal 20](#_Toc26615655)

[Plotly 23](#_Toc26615656)

[Заключение 27](#_Toc26615657)

[Список использованных источников литературы 28](#_Toc26615658)

# Введение

Истоки представления данных в виде таблиц, диаграмм и карт прослеживаются с древнейших времён. Ощутимая потребность в качественном представлении информации стала возникать в эпоху Возрождения, с появлением больших количеств данных и визуальной информации из географии, астрономии, геометрии, статистики и других наук.

В первой половине XIX века наблюдался значительный рост работ, в которых использовалось графическое отображение данных. К середине века были изобретены все основные типы представления данных: столбчатые и круговые диаграммы, гистограммы, линейные графики, графики временных рядов, контурные диаграммы и т. д.

Тенденция роста пошла на спад в начале XX века, уступив место точной математике. Тем не менее, именно в этот период стали появляться учебники и курсы по графическим методам представления данных, а сами графики стали использоваться не только для представления результатов, но и для исследования информации и выдвижения гипотез в астрономии, физике, биологии и других науках. [4]

На данный момент, большая часть работы специалистов в области data science это визуализация данных. На ранних стадиях развития проекта часто необходимо выполнять разведочный анализ данных (РАД, Exploratory data analysis (EDA)), чтобы выявить закономерности, которые обнаруживают данные. Визуализация данных помогает представить большие и сложные наборы данных в простом и наглядном виде. На этапе окончания проекта важно суметь отчитаться о его результатах так, чтобы даже непрофессионалам, не обладающим техническими знаниями, всё стало ясно и понятно. [6, 8]

# Библиотеки для визуализации данных

Если просматривать страницы Python Package Index, можно найти библиотеки практически для любого отображения данных, от GazeParser (для отслеживания движения глаз), до pastalog (для отображения развития нейронной сети в реальном времени). Некоторые из этих библиотек крайне узкоспециализированы, а некоторые могут использоваться почти для любой задачи.

Рассмотрим 10 достаточно универсальных и популярных Python библиотек для отображения данных.

## Matplotlib

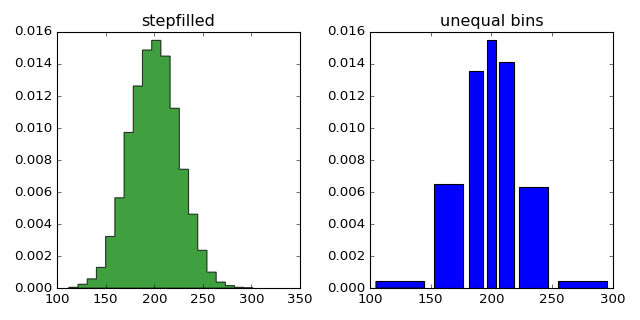
[](https://community.modeanalytics.com/python/libraries/matplotlib/)

Рис. 1. Пример графика из библиотеки Matplotlib

[Matplotlib](https://community.modeanalytics.com/python/libraries/matplotlib/) за более чем 10 лет своего существования уже фактически стала стандартом визуализации на Python. Многие современные библиотеки для построения графиков проектируются для работы совместно с matplotlib. Некоторые библиотеки, например, pandas или Seaborn, представляют из себя обёртки над matplotlib. Однако вместе с широтой возможностей приходит и сложность в устройстве, и, как следствие, работать с библиотекой тоже не всегда легко. [1-3]

## Seaborn

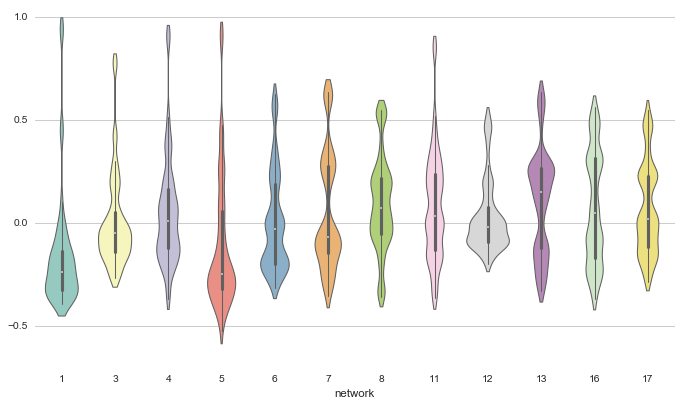
[](https://community.modeanalytics.com/python/libraries/seaborn/)

Рис. 2. Пример графика из библиотеки Seaborn

Seaborn представляет собой обёртку над matplotlib, привносящая в неё улучшения (главным образом в плане эстетики). [1-3]

## Ggplot

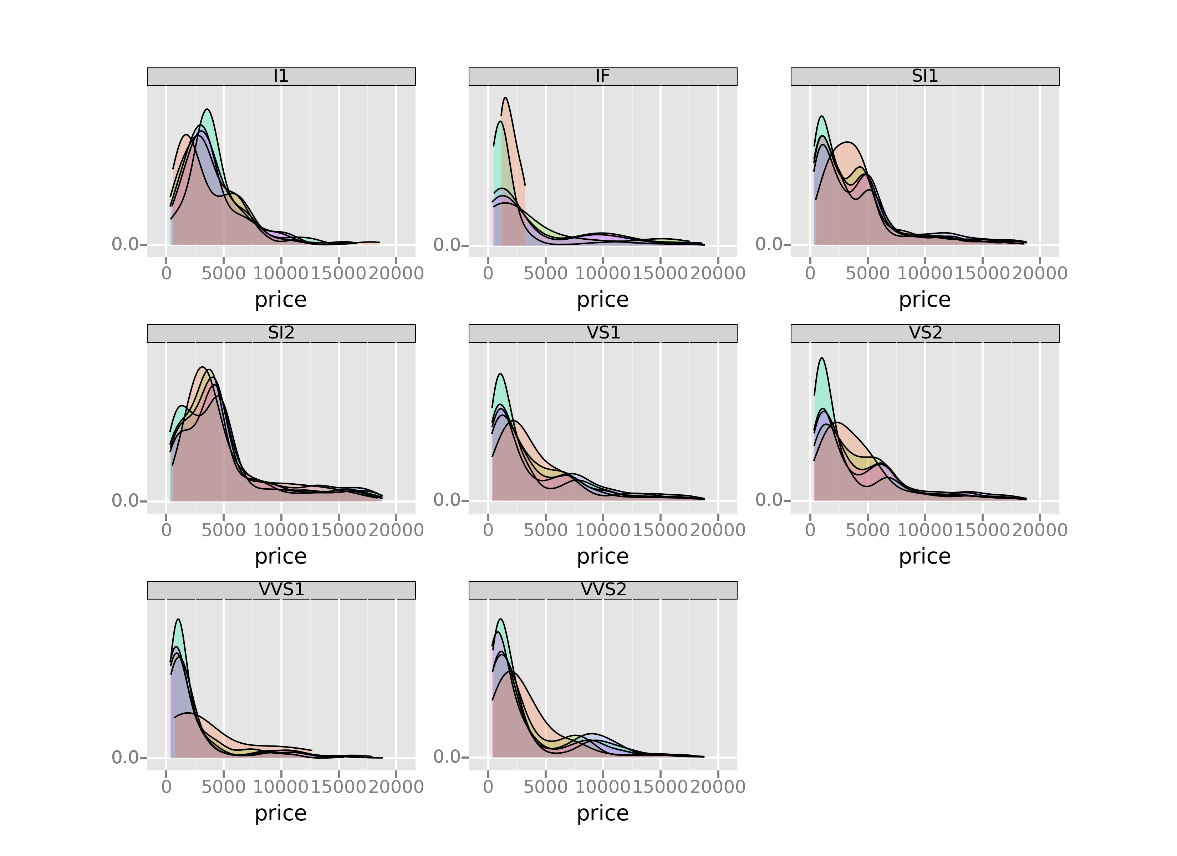
[](http://ggplot.yhathq.com/)

Рис. 3. Пример графика из библиотеки Ggplot

Ggplot базируется на ggplot2 (система построения графиков на языке R) и использует принципы Grammar of Graphics. Как следствие, работа с ней сильно отличается от работы с matplotlib. Исходя из описания, которое предоставил автор библиотеки, библиотека не предназначена для создания сложных персонализированных графиков, а ориентирована больше на простоту. [1-3]

## Bokeh

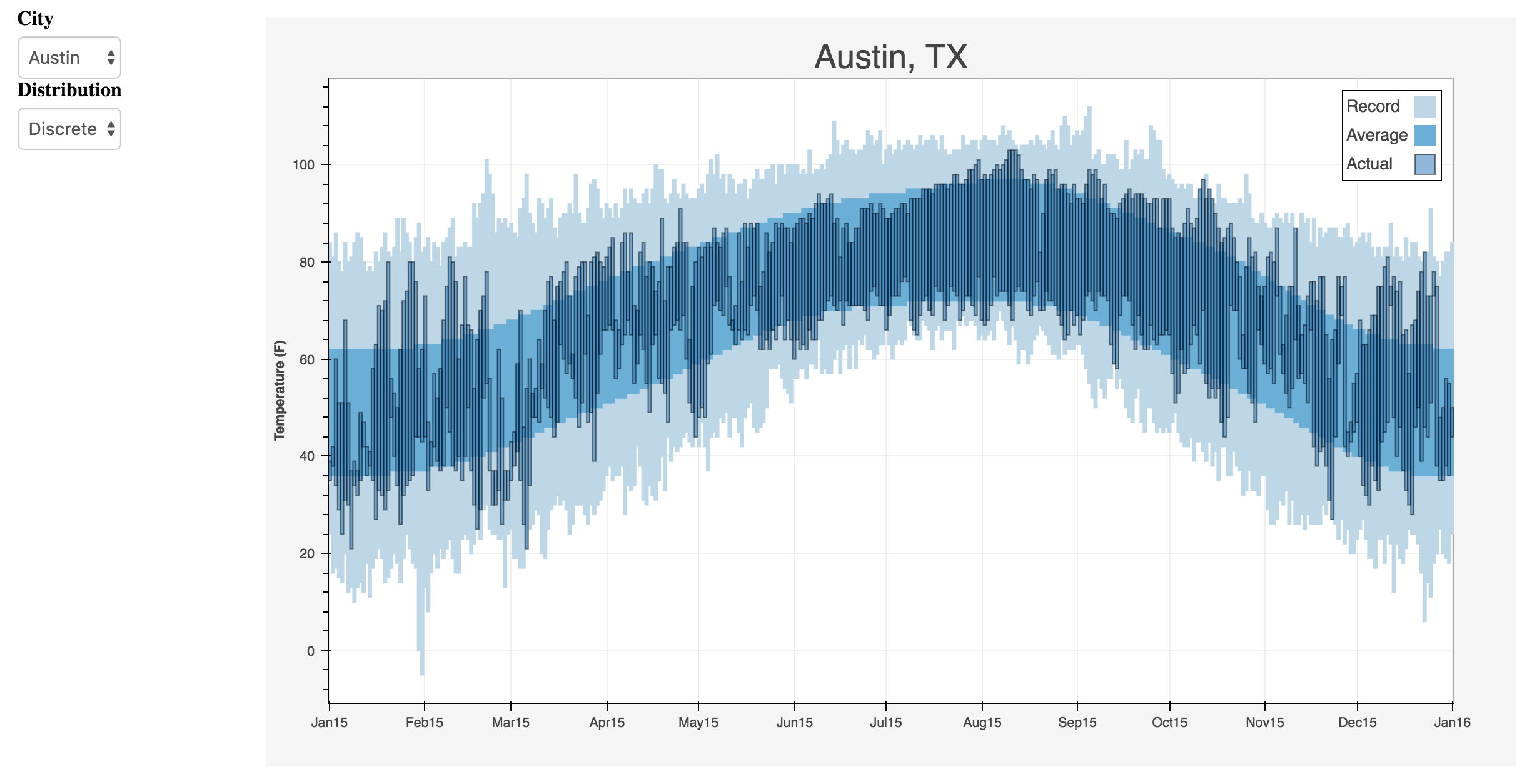
[](https://bokeh.pydata.org/en/latest/)

Рис. 4. Пример графика из библиотеки Bokeh

Bokeh также использует Grammar of Graphics, однако, в отличии от ggplot, он не портирован с R, а написан на самом Python. Библиотека поддерживает выгрузку в виде объектов JSON, в HTML-документы или интерактивные веб-приложения, а также поддерживает передачу данных в реальном времени и в виде потоков. [1-3]

## Pygal

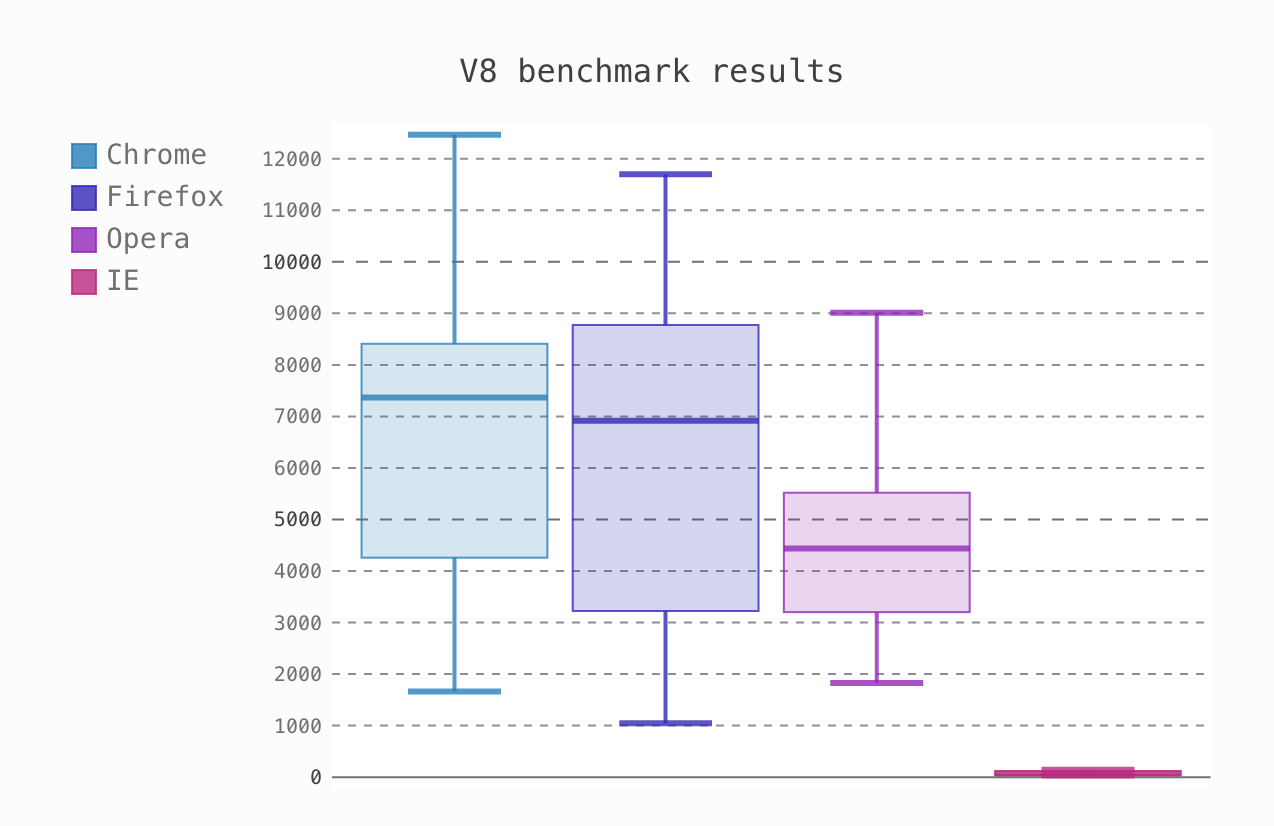
[](http://www.pygal.org/en/latest/index.html)

Рис. 5. Пример графика из библиотеки Pygal

Основным преимуществом данной библиотеки является возможность выгрузки данных в SVG-файлы (при больших объёмах данных SVG, могут возникнуть проблемы с производительностью) и возможность встраивать результат работы в веб-приложения. [1-3]

## Plotly

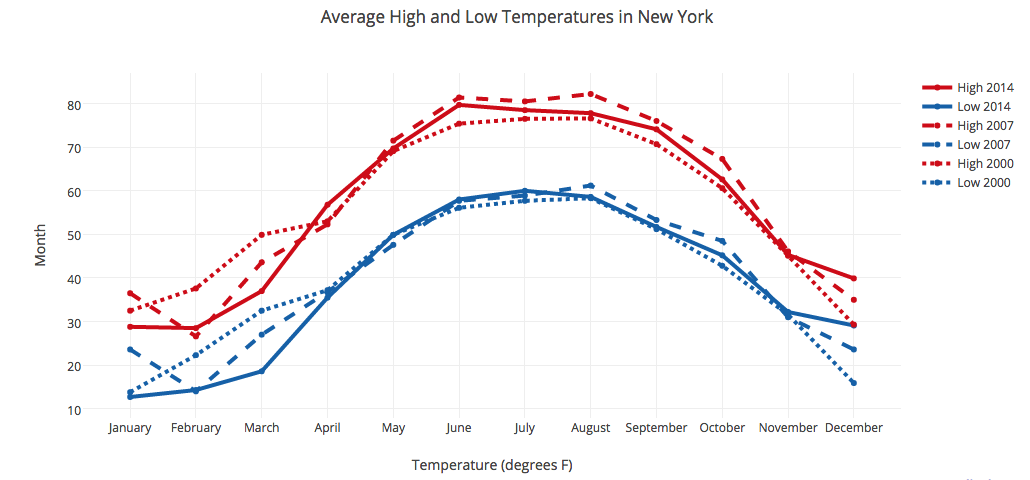
[](https://plot.ly/python/)

Рис. 6. Пример графика из библиотеки Plotly

Так же, как Pygal и Bokeh, Plotly адаптирован для работы в интерактивных веб-приложениях. Его уникальные возможности — контурные графики, дендограммы и 3D чертежи. [1-3]

## Geoplotlib

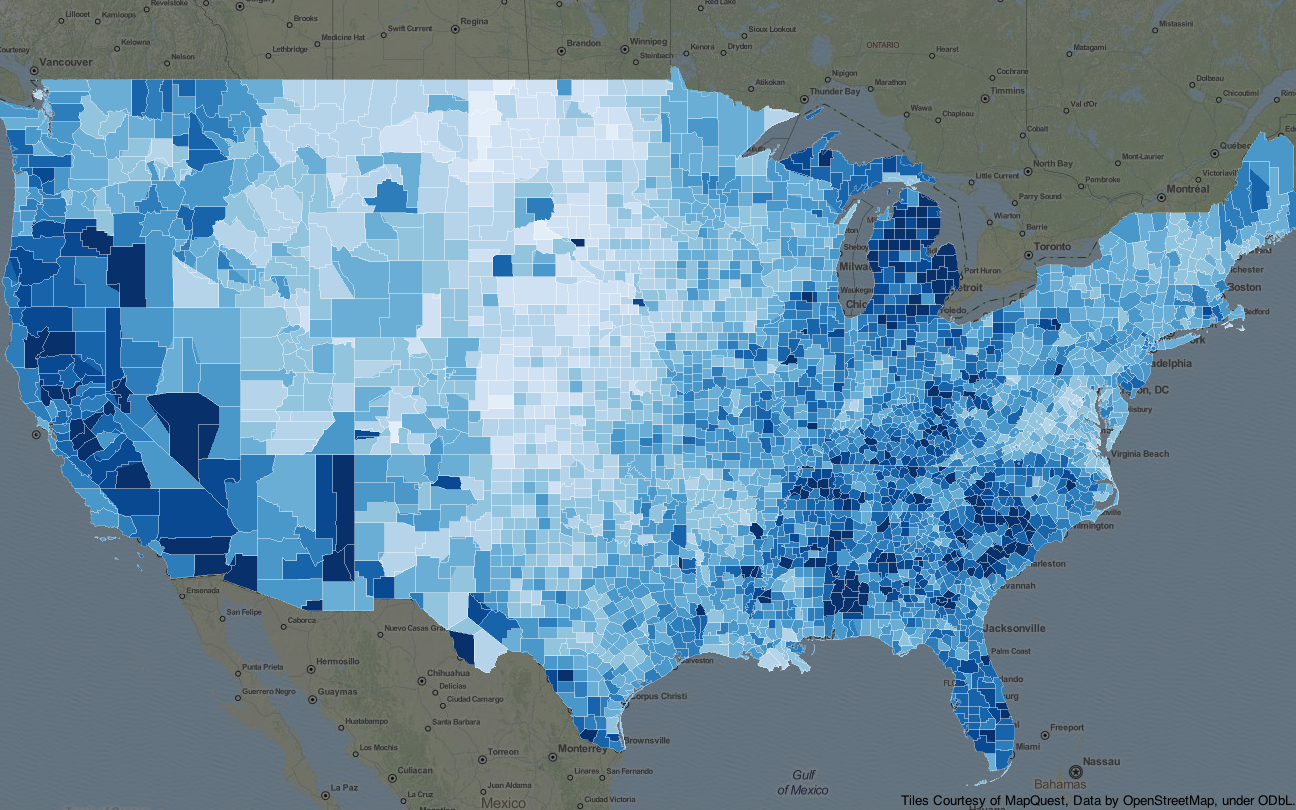
[](https://github.com/andrea-cuttone/geoplotlib)

Рис. 7. Пример графика из библиотеки Geoplotlib

Geoplotlib заточена под работу с картами и предлагает API для работы с ними. Также имеется возможность масштабировать и панорамировать карту. Geoplotlib автоматически обрабатывает загрузку данных, проекцию карты, загрузку листов карты и визуализацию графики с помощью OpenGL. Для её работы необходим Pyglet (объектно-ориентированный интерфейс). [1-3]

## Gleam

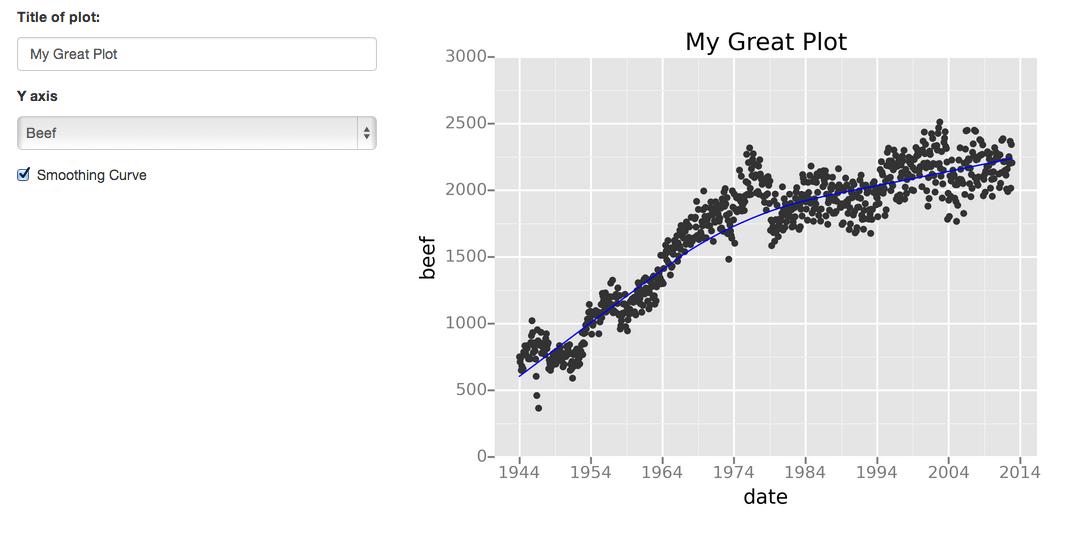
[](https://github.com/dgrtwo/gleam)

Рис. 8. Пример графика из библиотеки Gleam

Данная библиотека позволяет превращать результаты анализа данных в интерактивные веб-приложения, используя только Python скрипты, то есть не нужно знать ни HTML, ни CSS, ни JavaScript. Gleam может работать совместно с любой библиотекой визуализации Python. Создав график, вы можете подключить к нему поля для фильтров, чтобы пользователи могли сортировать и отбирать данные, которые им необходимы. [1-3]

## Missingno

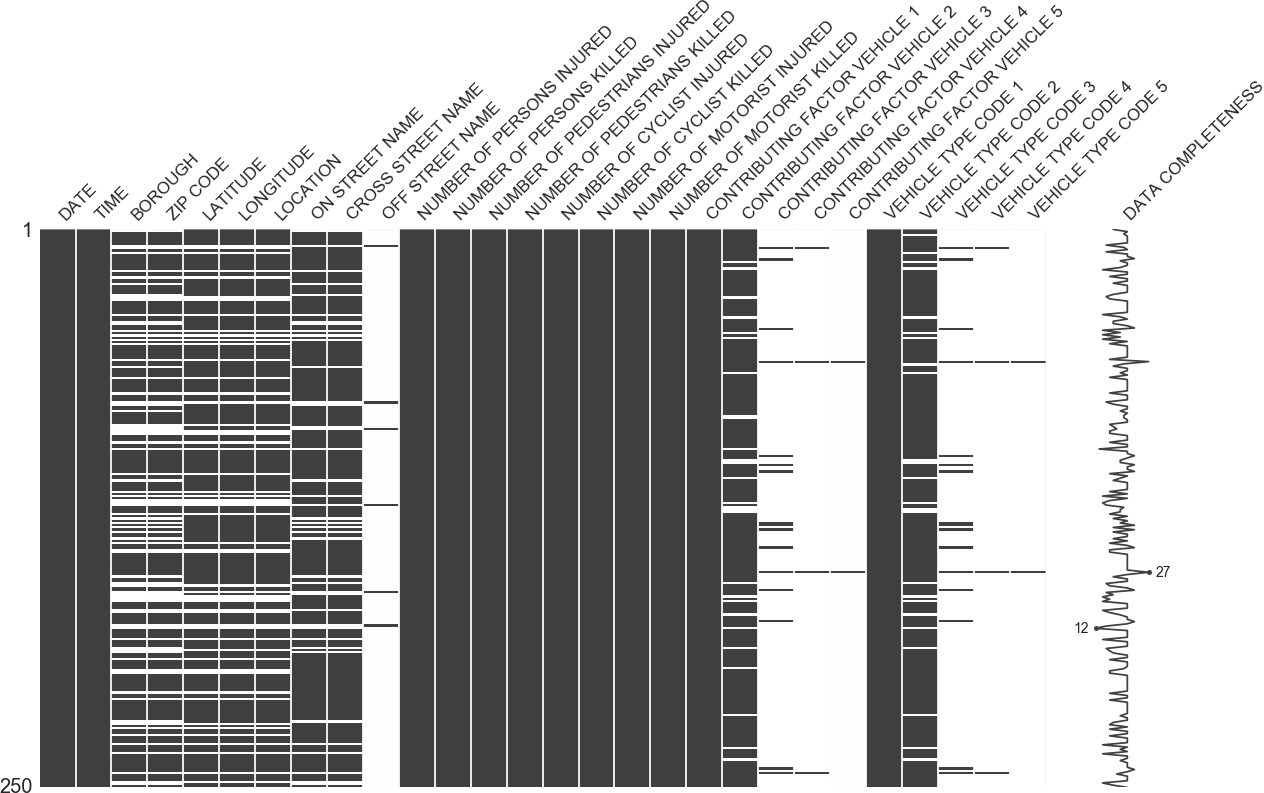
[](https://github.com/ResidentMario/missingno)

Рис. 9. Пример графика из библиотеки Missingno

Если библиотеки для очистки входных данных от пропущенных полей (путём удаления таких записей целиком, или подстановки средних/медианных значений) не подходят, то можно легко визуализировать полноту данных с помощью Missingno. Библиотека может не только визуализировать данные, но и сортировать их или отбирать в зависимости от корреляций. [1-3]

## Leather

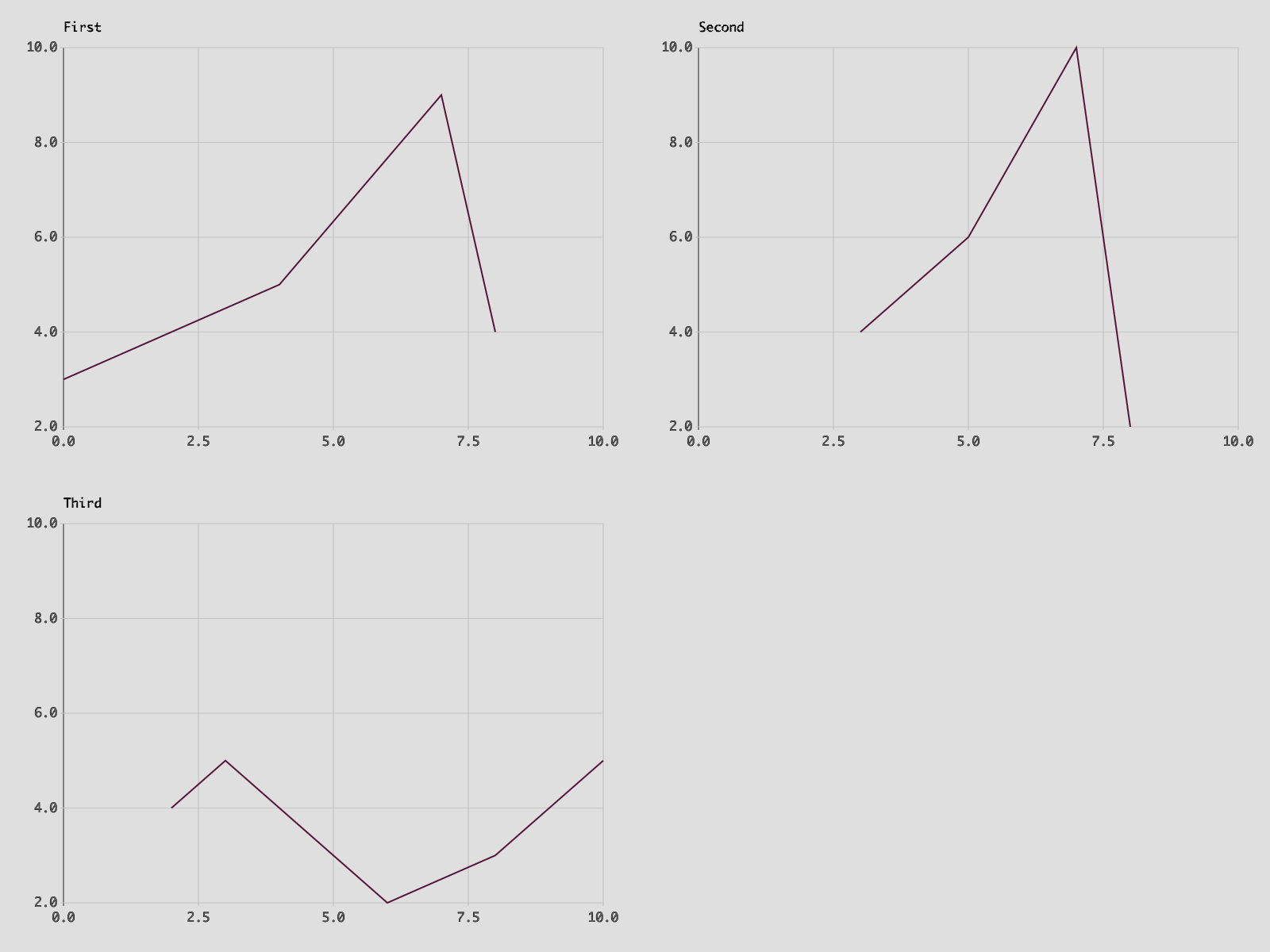
[](https://leather.readthedocs.io/en/latest/index.html)

Рис. 10. Пример графика из библиотеки Leather

Данная чертёжная библиотека для Python спроектирована для работы со всеми типами данных и выводит данные в SVG, благодаря чему вы можете масштабировать графики без потери качества. Библиотека достаточно новая, поэтому к ней всё ещё отсутствует часть документации. Чертежи с её помощью можно сделать достаточно примитивные. [1-3]

# Примеры реализации

## Установка Python и настройка окружения

Для установки Python достаточно будет перейти на сайт python.org, загрузить установочный файл и установить его. На Linux системах уже предустановлен Python 3.6:

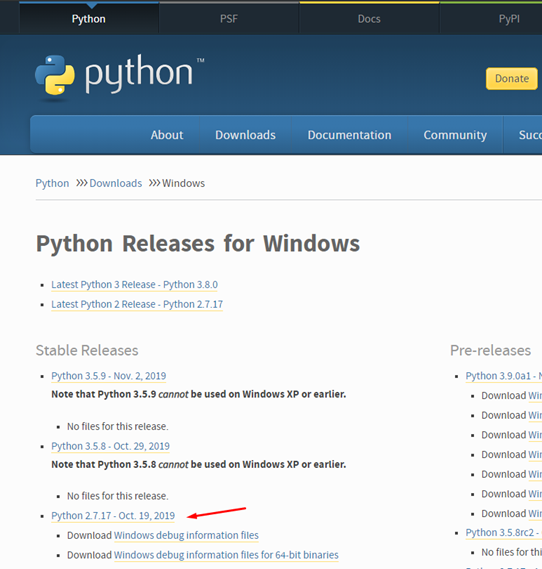


Рис. 11. Загрузка Python

Так как будет использоваться редактор кода Visual Studio Code, необходимо установить дополнительные расширения для работы с Python: Python и Code Runner:

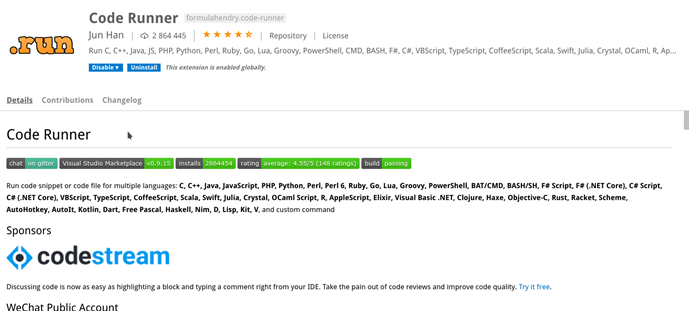
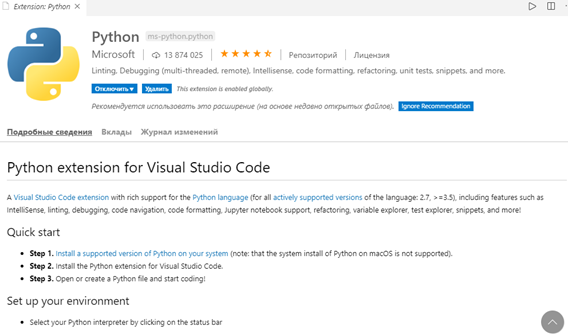


Рис. 12. Расширения для работы с Python

Для проверки работоспособности, достаточно создать файл «main.py» и ввести в нем код, выводящий в консоль «test» (для запуска кода выполнить комбинацию клавиш Сtrl + Аlt + N):

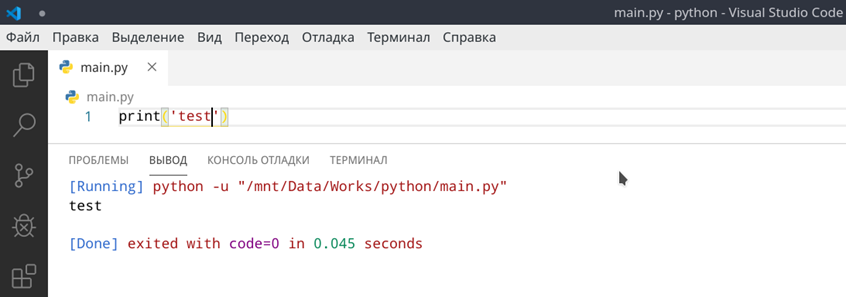


Рис. 13. Проверка работоспособности Python

Для демонстрации визуализации данных были отобраны следующие библиотеки: Matplotlib, Plotly и Pygal. С каждой библиотеки будет реализовано по несколько графиков.

## Matplotlib

Сперва, визуализируем данные с наиболее популярной библиотекой - Matplotlib. Для установки, потребуется ввести следующую команду, после чего начнётся её установка:



Рис. 14. Установка Matplotlib

Для начала, построим простой график sin(x), код которого приведен ниже, сохранив результат построения графика в файл. [11, 12, 14, 15]

Код:

# -\*- coding: utf-8 -\*-

import matplotlib.pyplot as plt

import numpy as np

# Сгенерируем значения X в пределах от -10 до 10 с шагом 0.01

x = np.arange(-10, 10, 0.01)

# Расчет значений для функции Y

y = 6 \* np.sin(x + 1) + 5

# Рисуем график

plt.plot(x, y)

# Устанавливаем описания для осей

plt.xlabel('X')

plt.ylabel('Y')

# Устанавливаем заголовок графика

plt.title('6 \* sin(x + 1) + 5')

# Включаем отображение сетки

plt.grid(True)

# Сохраняем результат в файл

plt.savefig('matplotlib/demo1.png')

# Отображаем график

plt.show()

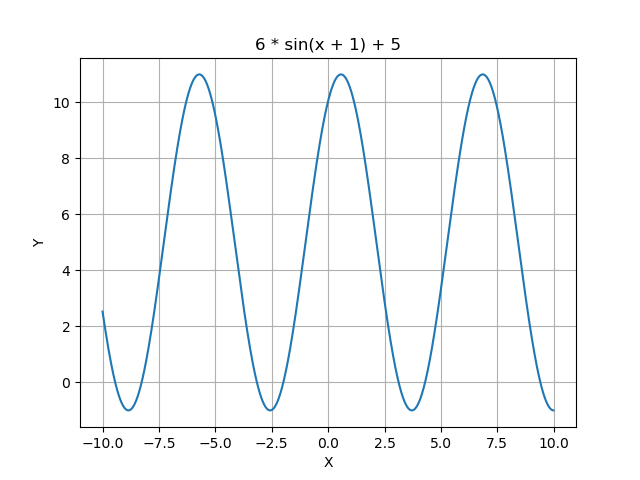


Рис. 15. Линейный график

Для построения гистограммы, при помощи данной библиотеки, не потребуется писать много кода (в основном большая часть кода описывает текстовую информацию графика). [9, 12, 15]

Код:

# -\*- coding: utf-8 -\*-

import matplotlib.pyplot as plt

import numpy as np

# Метки по оси Х

labels = ['G1', 'G2', 'G3', 'G4', 'G5']

# Значение по оси Y

men\_means = [20, 34, 30, 35, 27]

women\_means = [25, 32, 34, 20, 25]

x = np.arange(len(labels))  # Расположение меток

width = 0.4  # Ширина бара

# Создаем фигуру и набор подзаголовков

fig, ax = plt.subplots()

rects1 = ax.bar(x - width/2, men\_means, width, label='Мужской'.decode('utf-8'))

rects2 = ax.bar(x + width/2, women\_means, width, label='Женский'.decode('utf-8'))

# Добавляем текст к меткам и заголовку

ax.set\_ylabel('Счет'.decode('utf-8'))

ax.set\_title('Счет по группам и полу'.decode('utf-8'))

ax.set\_xticks(x)

ax.set\_xticklabels(labels)

ax.legend()

# Функция, формирующая метки над значениями

def autolabel(rects):

    for rect in rects:

        height = rect.get\_height()

        ax.annotate('{}'.format(height),

                    xy=(rect.get\_x() + rect.get\_width() / 2, height),

                    xytext=(0, 5),  # Вертикальный отступ на 10 пикселей

                    textcoords="offset points",

                    ha='center', va='bottom')

autolabel(rects1)

autolabel(rects2)

fig.tight\_layout()

# Сохраняем результат в файл

plt.savefig('matplotlib/demo2.png')

# Отображаем график

plt.show()

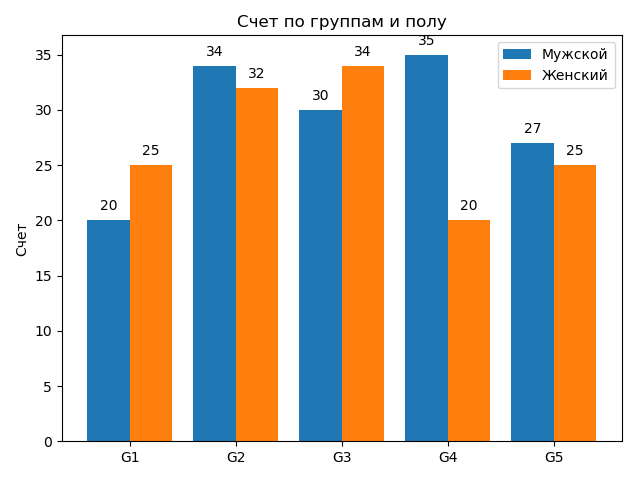


Рис. 16. Гистограмма

Также данная библиотека включает в себя возможность построения круговой диаграммы. [5, 12, 15]

Код:

# -\*- coding: utf-8 -\*-

import matplotlib.pyplot as plt

# Метки и их значения для диаграммы типа Pie

labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'

sizes = [15, 30, 45, 10]

explode = (0, 0.1, 0, 0)  # Акцентируем внимание на 2 ломтике - 'Hogs'

# Создаем фигуру и набор подзаголовков

fig1, ax = plt.subplots()

# Передаем информацию меток и значений

ax.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',

        shadow=True, startangle=90)

# Включаем отображение сетки

plt.grid(True)

# Сохраняем результат в файл

plt.savefig('matplotlib/demo3.png')

# Отображаем график

plt.show()

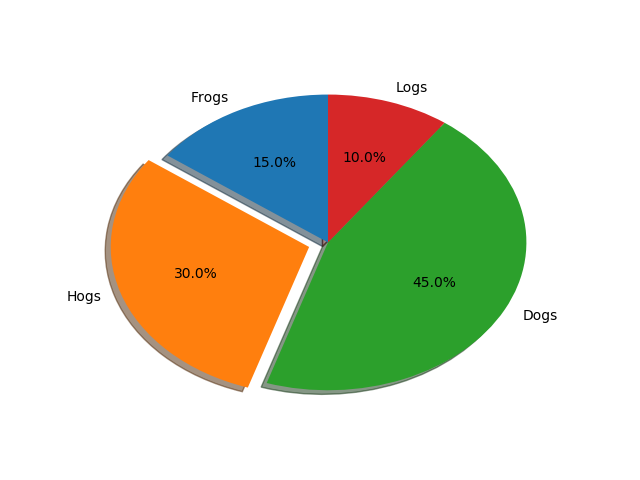


Рис. 17. Круговая диаграмма

Код для построения гистограммы распределения IQ при µ = 100 и σ = 15, приведен ниже. [9, 12, 15]

Код:

# -\*- coding: utf-8 -\*-

import numpy as np

import matplotlib.pyplot as plt

# Формируем 10к рандомных значений Х

mu, sigma = 100, 15

x = mu + sigma \* np.random.randn(10000)

# Передаем в метод данные X

plt.hist(x, 50, density=True, facecolor='g', alpha=0.75)

# Устанавливаем метки по осям

plt.xlabel('IQ')

plt.ylabel('Вероятность'.decode('utf-8'))

plt.title('Гистограмма распределения IQ'.decode('utf-8'))

plt.text(60, .025, r'$\mu=100,\ \sigma=15$')

plt.xlim(40, 160)

plt.ylim(0, 0.03)

plt.grid(True)

plt.savefig('matplotlib/demo4.png')

plt.show()

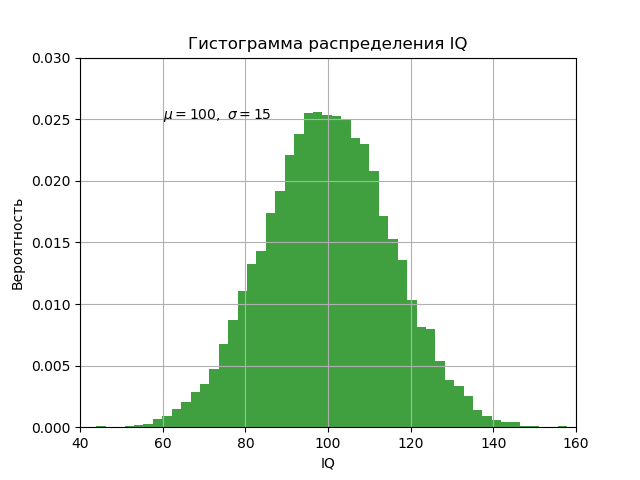


Рис. 18. Гистограмма распределения

В Matplotlib также имеется возможность для построения 3D фигур. [10, 12, 15]

Код:

# -\*- coding: utf-8 -\*-

import numpy as np

import matplotlib.pyplot as plt

from matplotlib import cm

from mpl\_toolkits.mplot3d import Axes3D

# Генерируем значения X и Y от -5 до 5 с шагом 0.25

X = np.arange(-5, 5, 0.25)

Y = np.arange(-5, 5, 0.25)

# Формируем координаты в виде матрицы

X, Y = np.meshgrid(X, Y)

# Находим расстояние

R = np.sqrt(X\*\*2 + Y\*\*2)

# Вычисляем значение Z

Z = np.sin(R)

# Создаем фигуру

fig = plt.figure()

ax = Axes3D(fig)

# Создаем полотно

ax.plot\_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.viridis)

plt.savefig('matplotlib/demo5.png')

plt.show()

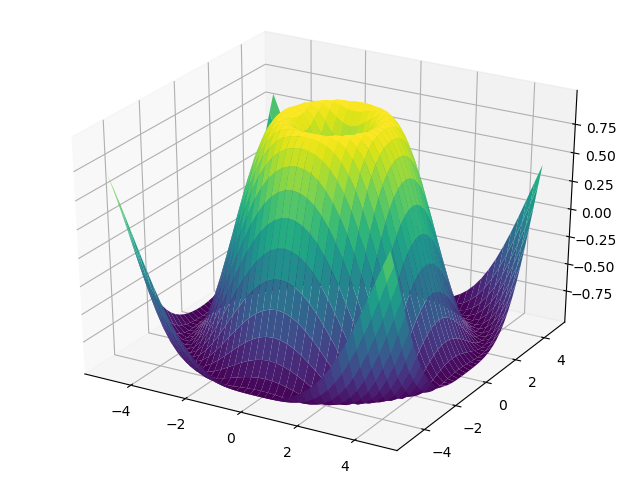


Рис. 19. 3D график

## Pygal

Следующей библиотекой будет Paygal, которая позволяет строить SVG графики и сохранять их в файлы под разными расширениями. Для установки необходимо ввести команду, приведенную ниже, после чего начнётся её установка:

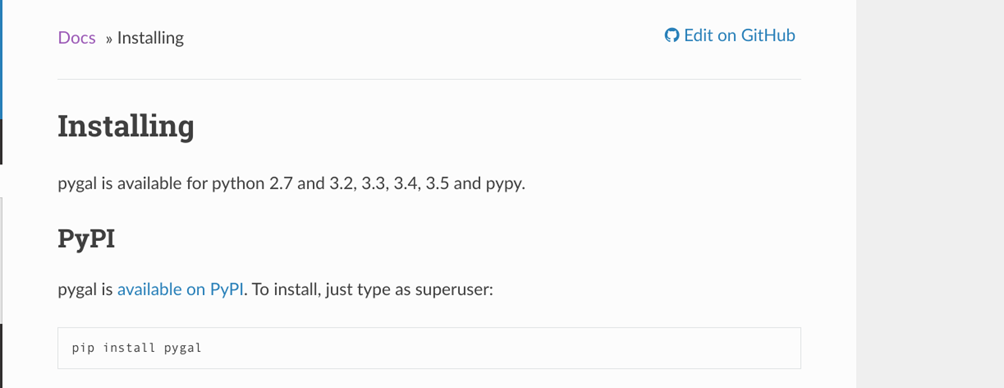


Рис. 20. Установка Pygal

Первым графиком данной библиотеки будет горизонтальная гистограмма, после выполнения кода ниже, станет доступен SVG файл, который можно будет открыть в браузере.

Код:

# -\*- coding: utf-8 -\*-

import pygal

# Инициализируем горизонтальный бар

line\_chart = pygal.HorizontalBar()

# Добавляем заголовок

line\_chart.title = 'Процент поддерживаемых браузер на февраль 2012'.decode('utf-8')

# Добавляем серию

line\_chart.add('IE', 19.5)

line\_chart.add('Firefox', 36.6)

line\_chart.add('Chrome', 36.3)

line\_chart.add('Safari', 4.5)

line\_chart.add('Opera', 2.3)

# Сохраняем график в файл SVG

line\_chart.render\_to\_file('./pygal/demo1.svg')

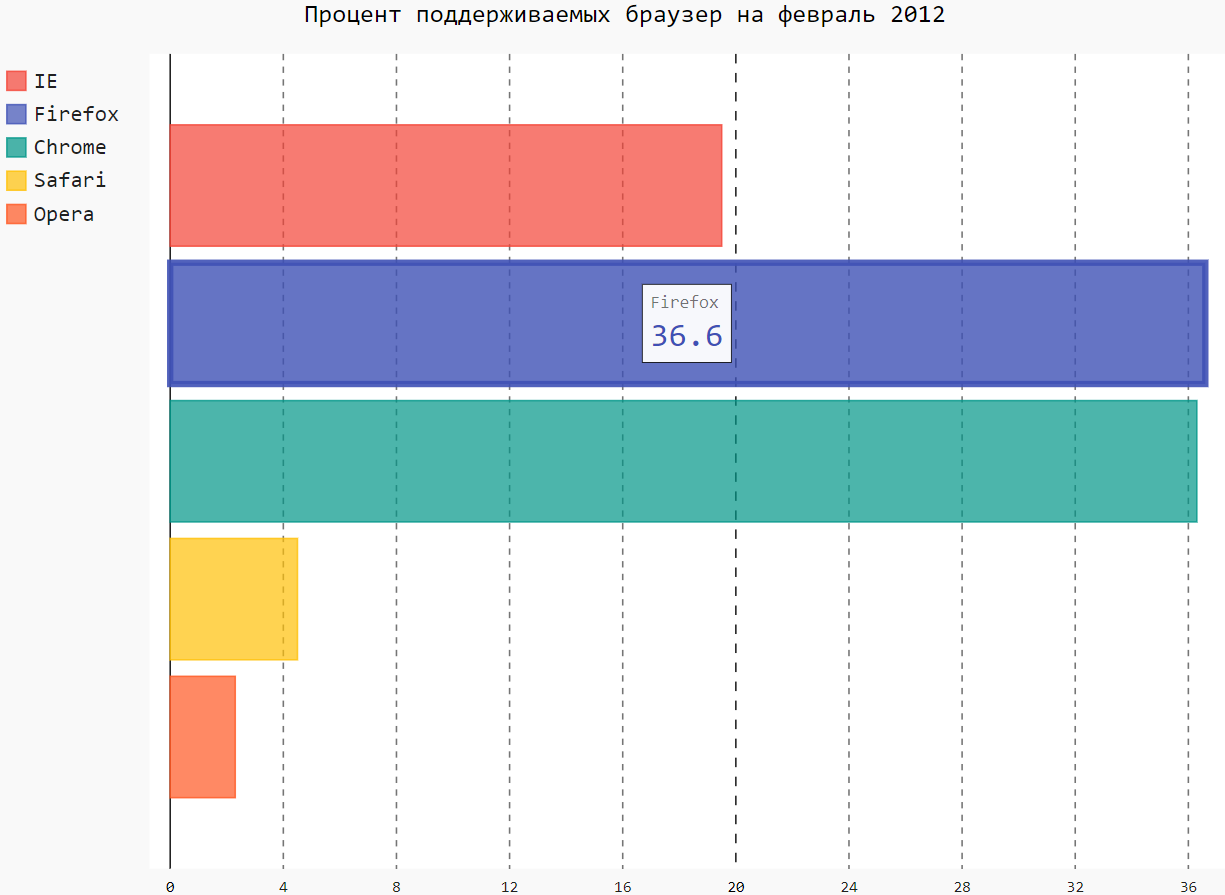


Рис. 21. Горизонтальная диаграмма

В данной библиотеке также доступна возможность построения датчиков, первая серия которой, будет выводиться в долларовом формате, а остальные в процентном. [7]

Код:

# -\*- coding: utf-8 -\*-

import pygal

# Создаем датчики с радиусом 0.70

gauge = pygal.SolidGauge(inner\_radius=0.70)

# Определяем функции для форматирования чисел

def percent\_formatter(x): return '{:.10g}%'.format(x)

def dollar\_formatter(x): return '{:.10g}$'.format(x)

# Указываем какой формат использовать

gauge.value\_formatter = percent\_formatter

# Также можно изменить тип форматирования числа на долларовый

gauge.add('Series 1', [{'value': 225000, 'max\_value': 1275000}], formatter=dollar\_formatter)

gauge.add('Series 2', [{'value': 110, 'max\_value': 100}])

gauge.add('Series 3', [{'value': 3}])

gauge.add(

    'Series 4', [

        {'value': 51, 'max\_value': 100},

        {'value': 12, 'max\_value': 100}])

gauge.add('Series 5', [{'value': 79, 'max\_value': 100}])

gauge.add('Series 6', 99)

gauge.add('Series 7', [{'value': 100, 'max\_value': 100}])

# Сохраняем график в файл SVG

gauge.render\_to\_file('./pygal/demo2.svg')

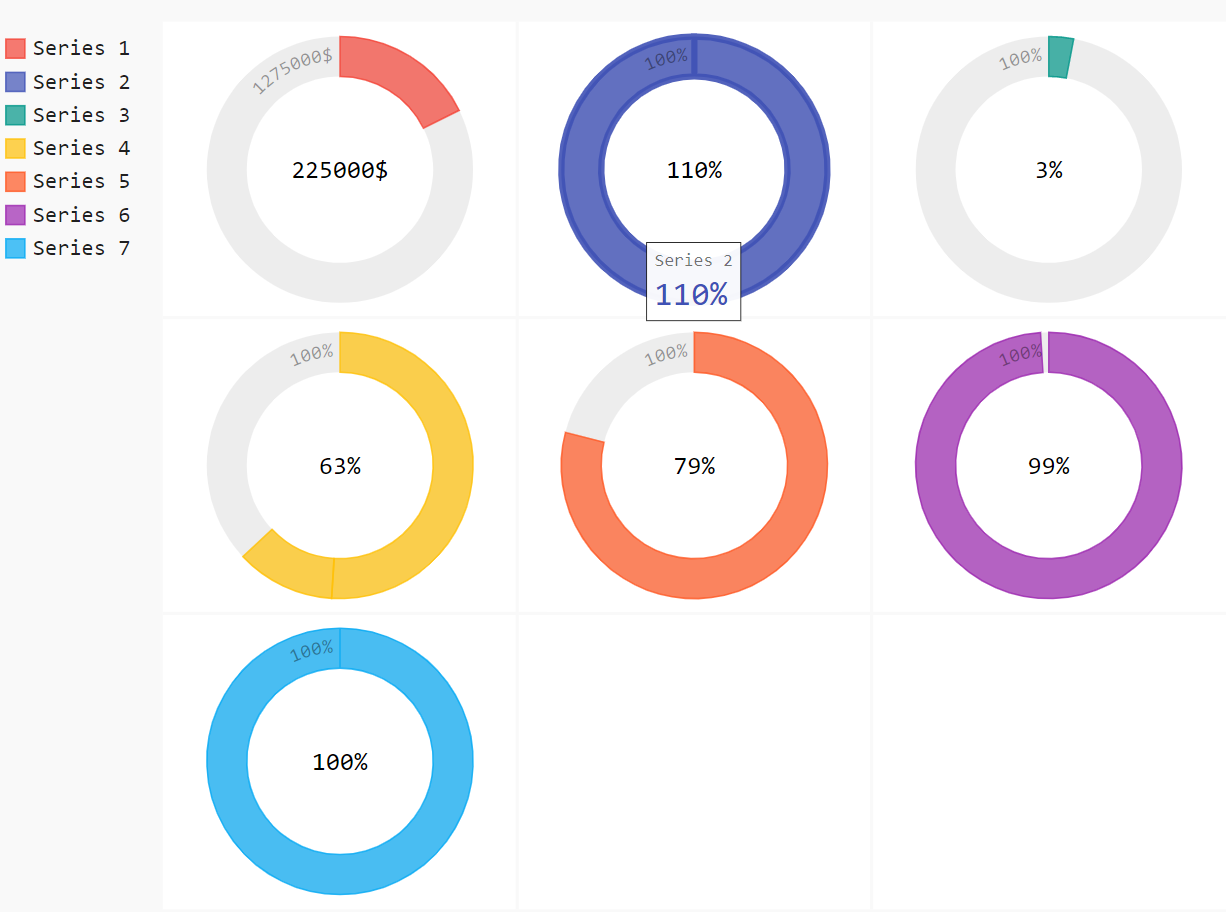


Рис. 22. Датчики

## Plotly

Данная библиотека имеет большой список возможных графиков для построения, но не более чем в Matplotlib. Для установки необходимо ввести команду, приведенную ниже, после чего начнётся её установка:



Рис. 23. Установка Plotly

Для первого графика потребуется дополнительно установить пакет Pandas, для парсинга файла с расширением \*.csv. После парсинга полученных данных можно приступать к построению графика японских свеч. Все графики будут выводиться в браузере. [13]

Код:

# -\*- coding: utf-8 -\*-

import plotly.graph\_objects as go

import pandas as pd

from datetime import datetime

# Считываем CSV данные с удаленного сервера

df = pd.read\_csv('https://raw.githubusercontent.com/plotly/datasets/master/finance-charts-apple.csv')

# Создаем экземпляр фигуры

fig = go.Figure(data=[go.Candlestick(x=df['Date'],

                open=df['AAPL.Open'],

                high=df['AAPL.High'],

                low=df['AAPL.Low'],

                close=df['AAPL.Close'])])

# Показываем график

fig.show()



Рис. 24. Японские свечи

Данная библиотека также имеет поддержку для построения 3D графиков.

Код:

# -\*- coding: utf-8 -\*-

import plotly.graph\_objects as go

import numpy as np

N = 70

# Создаем экземпляр фигуры с сгенерированными значениями X, Y, Z

fig = go.Figure(data=[go.Mesh3d(x=(70\*np.random.randn(N)),

                                y=(55\*np.random.randn(N)),

                                z=(40\*np.random.randn(N)),

                                opacity=0.5,

                                color='rgba(244,22,100,0.6)'

                                )])

# Показываем график

fig.show()

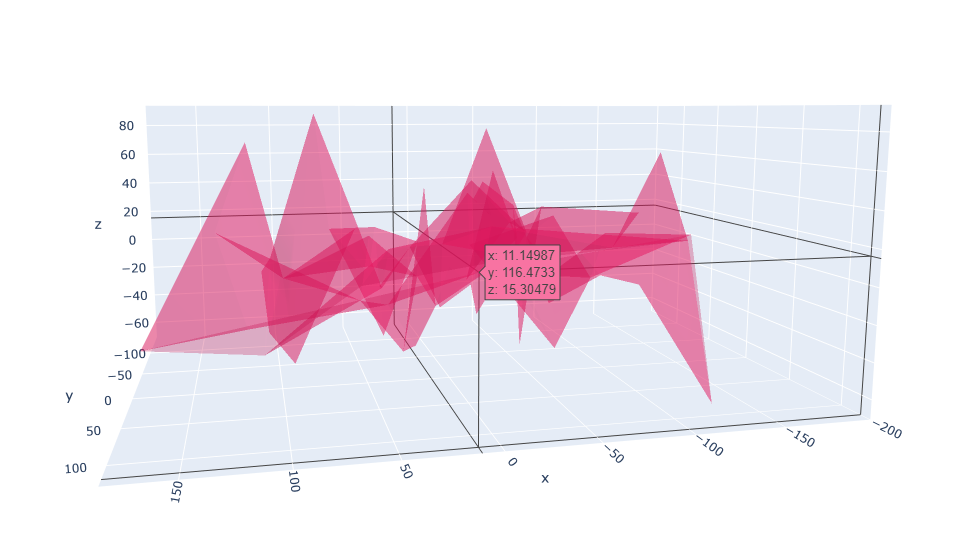


Рис. 25. 3D график

# Заключение

Ознакомившись с тремя библиотеками: Plotly, Pygal и Matplotlib, были продемонстрированы примеры реализации нескольких графиков с каждой библиотеки на языке Python.

Matplotlib — это популярная библиотека для визуализации данных, написанная на языке Python. Хоть пользоваться ей очень просто, настройка данных, параметров, графиков и отрисовки для каждого нового проекта — занятие нудное и утомительное.

Основным преимуществом библиотеки Pygal является возможность выгрузки данных в SVG-файлы (при больших объёмах данных SVG, могут возникнуть проблемы с производительностью) и возможность встраивать результат работы в веб-приложения/

Plotly адаптирован для работы в интерактивных веб-приложениях. Его уникальные возможности — контурные графики, дендограммы и 3D чертежи.

# Список использованных источников литературы

1. 10 простых хаков, которые ускорят анализ данных Python. URL: https://webstudio-uwk.ru/10-prostyh-hakov-kotorye-uskorjat-analiz-dannyh/ (дата обращения: 15.10.2019).
2. 18 полезных библиотек для анализа данных в Python. URL: https://tproger.ru/digest/python-data-library/ (дата обращения: 15.10.2019).
3. Анализируй данные с помощью одной строки на Python. URL: https://webstudio-uwk.ru/analiziruj-dannye-s-pomoshhju-odnoj-stroki-na/ (дата обращения: 15.10.2019).
4. Визуализация данных. URL: https://ru.wikipedia.org/wiki/Визуализация\_данных (дата обращения: 15.10.2019).
5. Построение графиков в Python. URL: http://acm.mipt.ru/twiki/bin/view/Cintro/PythonGraphs (дата обращения: 15.10.2019).
6. 5 простых способов визуализации данных на Python. С кодом. URL: https://medium.com/nuances-of-programming/5-простых-способов-визуализации-данных-на-python-с-кодом-e0053808c83d (дата обращения: 15.10.2019).
7. SolidGauge charts. URL: http://www.pygal.org/en/stable/documentation/types/solidgauge.html (дата обращения: 15.10.2019).
8. PyWay – гуру Питона. URL: https://tgstat.ru/channel/@pyway/183 (дата обращения: 15.10.2019).
9. Pyplot Text. URL: https://matplotlib.org/3.1.1/gallery/pyplots/pyplot\_text.html (дата обращения: 15.10.2019).
10. Как вывести на поверхность график / 3D-график из данных? URL: https://overcoder.net/q/573419/как-вывести-на-поверхность-график-3d-график-из-данных (дата обращения: 15.10.2019).
11. Настройка графиков. Текстовые элементы графикаю URL: https://devpractice.ru/matplotlib-lesson-3-3-text-elements/ (дата обращения: 15.10.2019).
12. Файловый вывод. URL: https://pyprog.pro/mpl/mpl\_file\_output.html (дата обращения: 15.10.2019).
13. Встречайте: версия 4.0 графической библиотеки plotly для Python. URL: https://proglib.io/p/plotly-4 (дата обращения: 15.10.2019).
14. Немного про кино или как делать интерактивные визуализации в python. URL: https://habr.com/ru/post/308162 (дата обращения: 15.10.2019).
15. 50 оттенков matplotlib — The Master. URL: https://habr.com/ru/post/468295 (дата обращения: 15.10.2019).