**Практическая работа**

**Библиотека Matplotlib**

Вероятно, Matplotlib является самым широко распространенным пакетом языка Python для графического отображения данных. Этот пакет можно использовать через процедурный интерфейс *pyplot* в очень быстро выполняемых скриптах для создания простых визуализаций данных. Если уделить немного больше внимания, то можно также получить изображения высокого качества для журнальных статей, книг и прочих публикаций. Функциональность для создания трехмерных графиков несколько ограничена, потому что эта библиотека предназначена главным образом для формирования двумерных изображений.

**Линейные графики и точечные диаграммы**

Matplotlib – крупный пакет, организованный в форме иерархической структуры: на самом высоком уровне находится модуль *matplotlib.pyplot*. Этот модуль предоставляет «среду конечного автомата» с интерфейсом, похожим на интерфейс MATLAB, и позволяет пользователю добавлять элементы графика (точки данных, линии, аннотации и т. п.) с помощью простых вызовов функций. На более низком уровне, который обеспечивает более продвинутое и гибко настраиваемое использование, *Matplotlib* предлагает объектно-ориентированный интерфейс, позволяющий создавать объект рисунка (figure) с прикрепленным нему одним или несколькими объектами осей (axes). В дальнейшем большинство изображений, аннотаций и прочих специализированных объектов существуют на этих осях. Именно такой подход применяется и описывается в практической работе.

Для использования библиотеки Matplotlib подобным способом рекомендуется выполнить следующие команды импорта:

*import matplotlib.pyplot as plt*

*import numpy as np*

**Построение графика на одном объекте оси**

Объект самого верхнего уровня, содержащий все элементы графика, называется Figure. Для создания объекта рисунка Figure вызывается метод plt.figure. Какие-либо аргументы не требуются, но дополнительные параметры специальной настройки можно задать, устанавливая значения, описанные в таблице 1. Например:

*# По умолчанию рисунок с заголовком "Figure 1".*

*fig = plt.figure()*

*# Небольшой рисунок (4.5" x 2") с красным фоном.*

*fig = plt.figure('Population density', figsize=(4.5, 2.),facecolor='red')*

|  |  |
| --- | --- |
| Аргумент | Описание |
| num | Идентификатор рисунка – если значение не задано, то используется целое число, начиная с 1, и увеличивается на 1 для каждого следующего создаваемого рисунка. Кроме  того, использование строки установит заголовок окна этого рисунка при выводе его  с помощью метода plt.show() |
| figsize | Кортеж размеров рисунка (width, height) – к сожалению, только в дюймах |
| dpi | Разрешение рисунка в точках на дюйм |
| facecolor | Цвет фона рисунка |
| edgecolor | Цвет границ рисунка |

Чтобы действительно изобразить данные графически, необходимо создать объект оси Axes – область рисунка, содержащую оси, графические метки, надписи, линии и маркеры графика и т. п. Самый простой рисунок, состоящий из одного объекта Axis, создается и возвращается следующим методом:

*ax = fig.add\_subplot()*

объекте Axis ax можно действительно графически изображать данные помощью метода ax.plot. Но здесь следует отметить, что метод plot в действительности возвращает список (list) объектов, представляющих графически изображаемые линии. При простейшем варианте использования изображается одна линия, поэтому список состоит из одного объекта типа Line2D, который можно присвоить переменной, если это необходимо. Рассмотрим более полный пример – приведенное ниже сравнение цепной линии *y = cosh(x)* и ее параболическую аппроксимацию *y = 1 + x2/2*.

*import matplotlib.pyplot as plt*

*import numpy as np*

*fig = plt.figure()*

*ax = fig.add\_subplot()*

*x = np.linspace(-2, 2, 1000)*

*line\_cosh = ax.plot(x,np.cosh(x))*

*line\_quad, = ax.plot(x, 1 + x\*\*2 / 2)*

*#plt.show()*

*plt.draw()*

*fig.savefig('plot.png', dpi=500)*

Две изображенные этим кодом линии показаны на рис. 1.

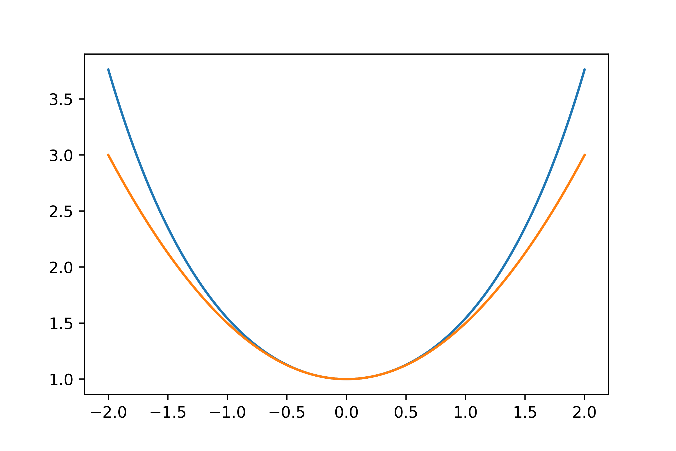


Рис.1.Простой график из двух линий в одном объекте Axes

Если в методе *fig.add\_subplot()* не требуется передача аргументов, то объекты *fig* и *ax* можно создать в одной строке кода с помощью удобной функции *plt.subplots()*:

*fig, ax = plt.subplots()*

**Границы графика**

По умолчанию *Matplotlib* отображает все данные, переданные в метод *plot*, устанавливает границы оси соответствующим образом. Чтобы установить для границы оси другое значение, используются методы *ax.set\_xlim* и *ax.set\_ ylim*. Кроме того, можно установить обе границы или значения границ по отдельности с помощью аргументов *left*, *right* (или *xmin*, *xmax*) и *bottom*, *top* (или *ymin*, *ymax*). Незаданные значения границ остаются неизменными. Например:

*x = np.linspace(-3, 3, 1000)*

*y = x\*\*3 + 2 \* x\*\*2 - x - 1*

*fig = plt.figure()*

*ax = fig.add\_subplot()*

*ax.plot(x, y)*

*ax.set\_xlim(-1, 2)*

*ax.set\_ylim(bottom=0)*

*# Установлены границы оси x: от -1 до 2.*

*# ymin=0: график будет "отсечен" по нижней границе bottom.*

Если значение *bottom* больше значения *top* или значение *right* меньше значения *left*, то направление соответствующей оси будет изменено на противоположное, т. е. значения по этой оси будут уменьшаться слева направо (или снизу вверх). Если необходимо изменить направление оси на противоположное без изменения значений границ, то вызываются методы *ax.invert\_xaxis()* и *ax.invert\_yaxis()*, позволяющие сделать это.

**Стили линий, маркеры и цвета**

Как мы уже видели ранее, стиль графика можно определить, передавая дополнительные аргументы в метод *plot()*. По умолчанию для линии назначается сплошной стиль, толщина 1.5 пт и цвет, определяемый порядком, в котором эта линия добавляется в график. Другой стиль линии можно выбрать из предварительно определенных вариантов с помощью аргумента *linestyle* (или просто ls). Допустимые строковые значения для передачи в этом аргументе (включая пустую строку для отсутствия отображения линии) показаны в табл. 2.

Таблица 2. Стили линий Matplotlib

|  |  |
| --- | --- |
| - | Сплошная |
| -- | Штриховая |
| : | Пунктирная |
| -. | Штрихпунктирная |

Можно еще более подробно определить стиль линии, передавая в аргументе *dashes* последовательность значений, описывающих повторяющийся шаблон штрихов в пунктах (пт). Например, *dashes=[2, 4, 8, 4, 2, 4]]* представляет шаблон из точки (2 пт), пробела (4 пт), штриха (8 пт), пробела (4 пт), точки (2 пт), пробела (4 пт), который должен повторяться, формируя стиль линии. Для получения того же результата можно вызвать метод изображаемой линии set\_dashes, как показано в следующем фрагменте кода:

*x = np.linspace(-np.pi, np.pi,1000)*

*line, = plt.plot(x, np.sin(x))*

*line.set\_dashes([2, 4, 8, 4, 2, 4]) # Шаблон точка-штрих-точка.*

Толщина линии определяется значением аргумента lineweight (или просто lw) в пунктах (пт). Цвет линии передается в аргументе color (или просто c), используемом одним из следующих способов:

* строка: буква или имя – одно из значений, приведенных в табл. 3. Обозначенные одной буквой цвета более светлые, а заданные по умолчанию «живые» цвета более приятны для глаз;
* строка: HTML-строка из шести шестнадцатеричных цифр с префиксом ‘#’, например ‘#ffff00’ обозначает желтый цвет;
* строка: строковое представление числа типа float со значением от 0. до 1. (например, '0.4') определяет шкалу оттенков серого цвета от черного (0.) до белого (1.); кортеж чисел типа float в диапазоне от 0. до 1.: компоненты RGB, на-пример (0.5, 0., 0.) – это темно-красный цвет.

Таблица 3. Буквенные коды цветов Matplotlib

|  |  |
| --- | --- |
| Коды основных цветов | Живые цвета |
| b = синий | tab:blue |
| g = зеленый | tab:orange |
| r = красный | tab:green |
| c = бирюзовый | tab:red |
| m = фиолетовый | tab:purple |
| y = желтый | tab:brown |
| k = черный | tab:pink |
| w = белый | tab:gray |
|  | tab:olive |
|  | tab:cyan |

По умолчанию объект *Line2D*, создаваемый при вызове метода *plot* в объекте Axes, не включает маркеры: символы, выводимые в каждой точке данных графика. Для их добавления необходимо определить один из односимвольных кодов маркеров и передать его в аргументе *marker*:

*ax.plot(x, y, marker='v') # Маркер: треугольник вершиной вниз.*

Другие свойства маркеров можно определить с помощью аргументов, перечисленных в табл. 4.

Таблица 4. Свойства маркеров Matplotlib

|  |  |  |
| --- | --- | --- |
| **Аргумент** | **Сокращенный вариант** | **Описание** |
| markersize | ms | Размер маркера в пунктах (пт) |
| markevery |  | Если явно установлено положительное целочисленное значение N, то маркер выводится в каждой N-й точке. По умолчанию установлено значение None – маркер выводится в каждой точке |
| markerfacecolor | mfc | Цвет заливки маркера |
| markeredgecolor | mec | Цвет границы маркера |
| markeredgewidth | mew | Толщина границы маркера в пунктах (пт) |

Свойства маркеров Matplotlib можно настраивать более детально, подробности см. в официальной документации (https://matplotlib.org/api/markers\_api.html).

**Точечные диаграммы**

Обычная двумерная точечная диаграмма изображает данные как точки в декартовой системе координат (осей). Иногда не имеет смысла или не приносит никакой пользы упорядочивание данных, следовательно, нет необходимости соединять точки данных линиями. Метод *pyplot.scatter* создает точечную диаграмму (график рассеяния). В дополнение к одномерным последовательностям x- и y-данных, как для *pyplot.plot*, цвета и размеры маркеров точек данных можно устанавливать отдельно, передавая последовательность соответствующих значений той же длины, что и сама последовательность данных, в аргументах c и s соответственно. Размер маркера указывается в пт2 (в квадратных пунктах), так что площадь маркеров пропорциональна значениям, передаваемым аргументе s. Работа с размером маркеров является общепринятым способом обозначения третьего измерения данных, как показано в следующем примере.

**Пример 1.** Для исследования зависимости между уровнем рождаемости, средней продолжительностью жизни и доходом на душу населения можно воспользоваться точечной диаграммой. Размеры маркеров устанавливаются пропорционально объему ВВП на душу населения, но их необходимо немного масштабировать, чтобы они не становились слишком большими (см. рис. 2).

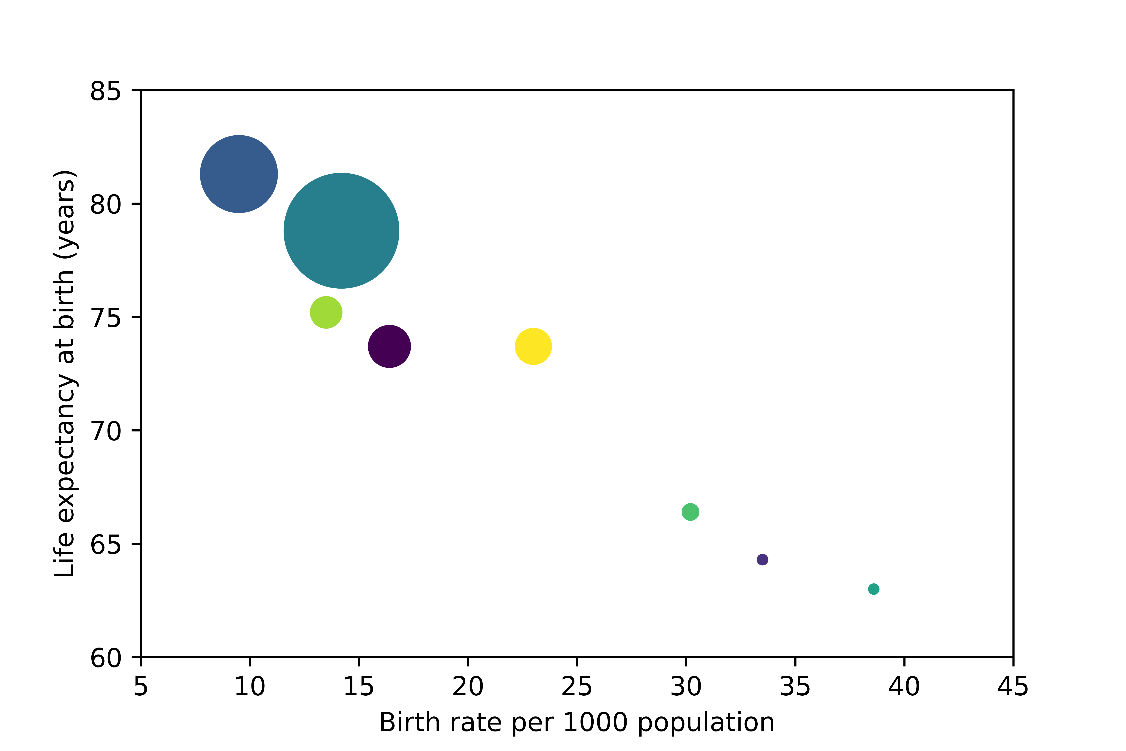


Рис. 2. Точечная диаграмма с различными размерами маркеров, обозначающими объем ВВП в каждой стране

Листинг 1. Точечная диаграмма демографических данных по восьми странам

*import numpy as np*

*import matplotlib.pyplot as plt*

*countries = ['Brazil', 'Madagascar', 'S. Korea', 'United States', 'Ethiopia', 'Pakistan', 'China', 'Belize']*

*#Уровень рождаемости на 1000 населения.*

*birth\_rate = [16.4, 33.5, 9.5, 14.2, 38.6, 30.2, 13.5, 23.0]*

*#Ожидаемая средняя продолжительность жизни при рождении, в г.*

*life\_expectancy = [73.7, 64.3, 81.3, 78.8, 63.0, 66.4, 75.2, 73.7]*

*#Доход на душу населения, определенный в 2000 г. в долл. США.*

*GDP = np.array([4800, 240, 16700, 37700, 230, 670, 2640, 3490])*

*fig, ax = plt.subplots()*

*#Некоторые произвольно выбираемые цвета:*

*colors = range(len(countries))*

*ax.scatter(birth\_rate, life\_expectancy, c=colors , s=GDP/20)*

*ax.set\_xlim(5, 45)*

*ax.set\_ylim(60, 85)*

*ax.set\_xlabel('Birth rate per 1000 population')*

*ax.set\_ylabel('Life expectancy at birth (years)')*

*plt.show()*

*plt.draw()*

*fig.savefig("image.png",dpi=500)*

**Задание 1**

Чтобы пометить пузырьки в примере 1, добавьте следующий код перед отображением графика:

*offset = 1*

*for x, y, s, country in zip(birth\_rate, life\_expectancy, GDP, countries):*

*ax.text(x+offset, y, country, va='center')*

**Задание 2**

Кружковая («пузырьковая») диаграмма представляет собой тип точечной диаграммы, который может отображать три измерения данных с использованием позиции (точки данных как координаты x и y) и размера маркера. Метод plt.scatter может создавать кружковые диаграммы, принимая размер маркера в атрибуте s  (в кв. пт. – так что площадь маркера пропорциональна величине в третьем измерении – см. пример П7.1).

Файлы gdp.tsv, bmi\_men.tsv и population\_total.tsv, доступные для скачивания в ресурсах, содержат следующие данные, начиная с 2007 г. для каждой страны: ВВП на душу населения в международных долл., зафиксированных в ценах 2005 г., индекс массы тела (ИМТ) мужчин (в  кг/м2 ) и  общая численность населения. Создать кружковую диаграмму отношения ИМТ и ВВП, на которой численность населения соответствует размеру круговых маркеров. Предупреждение: для некоторых стран отдельные точки данных отсутствуют. Дополнительное задание: обозначить цветом кружки по континентам, используя список из файла continents.tsv. Результат представлен на рисунке 3.

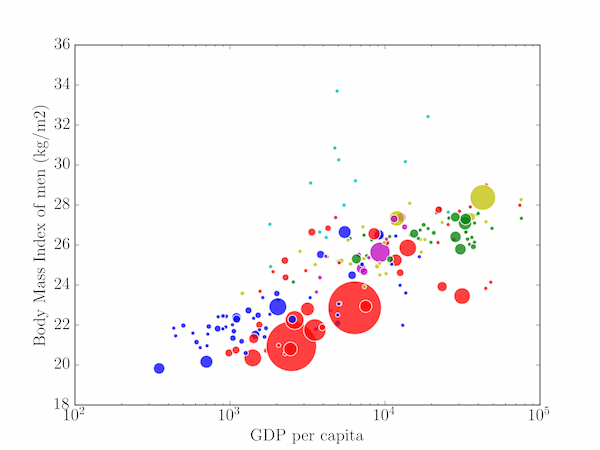


Рисунок 3. Точечная диаграмма