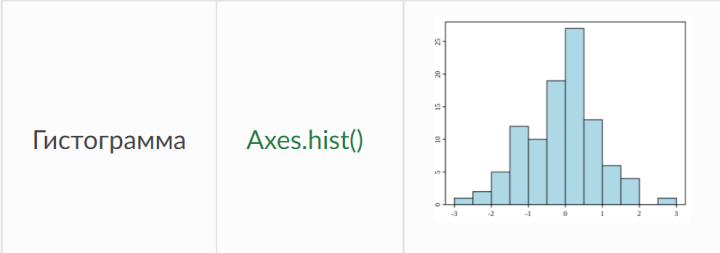
**1\_4-3. Практика 3**

**Основные типы диаграмм в matplotlib**

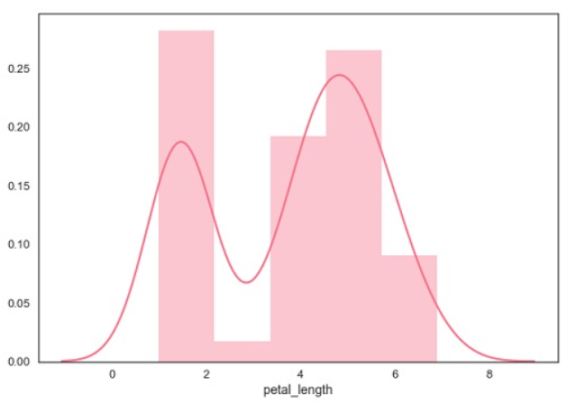
Класс **matplotlib.axes.Axes** содержит множество методов для рисования диаграмм различного типа (Таблица 1).

Таблица 1





Диаграммы можно комбинировать, вызывая последовательно разные методы, например, на одной диаграмме может быть нарисована столбчатая диаграмма и график одновременно.



## 1. Круговые диаграммы

Круговая диаграмма отображает размер элементов некоторого ряда данных относительно суммы всех элементов, а результаты на круговой диаграмме выводятся как проценты от всего круга.

Круговая диаграмма может быть нарисована с использованием метода [Axes.pie()](http://matplotlib.org/api/_as_gen/matplotlib.axes.Axes.pie.html" \l "matplotlib.axes.Axes.pie" \t "_blank).

Функция pie():

**pie(*x*, *explode=None*, *labels=None*, *colors=None*, *autopct=None*, *pctdistance=0.6*, *shadow=False*, *labeldistance=1.1*, *startangle=None*, *radius=None*, *counterclock=True*, *wedgeprops=None*, *textprops=None*, *center=(0, 0)*, *frame=False*)**

рисует круговую диаграмму для значений *х*, где доля круга определяется, как *х*/sum(*x*). По умолчанию доли (сектора) рисуются против часовой стрелки относительно оси OX.

Таблица 2

**Параметры функции** pie()

|  |  |
| --- | --- |
| Параметр | Описание |
| x | набор значений (обязательный параметр) |
| labels | список подписей для секторов |
| explode | список долей, выносимых из диаграммы |
| colors | набор значений цветов секторов (долей) |
| autopct | форматная строка или функция для отображения значения доли на секторах круга |
| pctdistance | расстояние от центра доли до текстовой метки |
| shadow | отображение тени у диаграммы, если True |
| labeldistance | расстояние для текстовых метод долей (по умолчанию 1.1) |
| startangle | начальный угол поворота диаграммы (против часовой стрелки) |
| radius | радиус диаграммы (по умолчанию равен 1) |
| counterclock | порядок размещения долей на диаграмме (по часовой стрелки или против часовой) True/False |
| center | координата центра диаграммы (по умолчанию (0, 0)) |
| frame | отображение рамки вокруг диаграммы (True/False) |
| wedgeprops | словарь дополнительных параметров (см. класс matplotlib.patches.Wedge) |

Для начала работы выполним:

import pandas as pd

import matplotlib as mpl

import matplotlib.pyplot as plt

import numpy as np

%matplotlib inline

**Пример 1.** В компании работают 50 человек из них 40 женщин и 10 мужчин. Отобразим на круговой диаграмме доли мужчин и женщин в компании:

plt.pie([40,10])

plt.show()

**Пример 2.** Количество элементов в массиве определяет количество секторов (клиньев), а величина значений определяет их площадь, например:

plt.pie([5,13,21,27,10,17])

plt.show()

**Пример 3.** Круговая диаграмма со списком подписей секторов

vals = [10, 40, 23, 30, 7]

labels = ['Toyota', 'BMW', 'Lexus', 'Audi', 'Lada']

plt.pie(vals, labels = labels)

plt.show()

**Пример 4.** Нарисуем круговую диаграмму, используя параметры autopct – функция для отображения значения доли на секторах круга и explode – список долей, выносимых из диаграммы (доли выносятся в exp против часовой стрелки):

vals = [10, 40, 23, 30, 7]

labels = ['Toyota', 'BMW', 'Lexus', 'Audi', 'Lada']

exp = (0.1, 0.2, 0, 0, 0)

plt.pie(vals, labels = labels, autopct = '%.2f', explode = exp, shadow = True)

plt.show()

**Пример 5.** Используя словарь wedgeprops (клинья), можно сформировать диаграмму с пустотой внутри:

ax.pie(vals, labels = labels, shadow = True, wedgeprops = dict(width = 0.5))

Программный код:

vals = [10, 40, 23, 30, 7]

labels = ['Toyota', 'BMW', 'Lexus', 'Audi', 'Lada']

plt.pie(vals, labels = labels, shadow = True, wedgeprops = dict(width = 0.5))

plt.show()

**Пример 6.** Рассмотрим еще один пример создания круговой диаграммы

fig, ax = plt.subplots()

fig.canvas.set\_window\_title("Фактическое исполнение бюджета Москвы в 2015 году")

*# Настройки диаграммы и осей*

ax.set\_title("Фактическое исполнение бюджета Москвы в 2015 году")

ax.set\_xlabel("https://data.mos.ru/opendata/"

"7710152113-struktura-dohodov-byudjeta/row/37352740")

*# https://data.mos.ru/opendata/7710152113-struktura-dohodov-byudjeta/row/37352740*

data= [

["Налог на прибыль", 484.7],

["Налог на доходы физических лиц", 691.6],

["Прочие налоговые доходы", 221.2],

["Арендные платежи", 59.4],

["Доходы от продажи имущества", 26.7],

["Безвозмездные поступления", 75.5]

]

values = [x[1] **for** x **in** data]

labels = [x[0] **for** x **in** data]

ax.pie(values, labels = labels, autopct = "**%.1f%%**", radius = 1.2)

ax.set\_aspect("equal")

plt.show()

## 2. Столбчатые диаграммы

Столбчатые диаграммы используются для сравнения отдельных элементов данных. В диаграммах этого типа категории обычно располагаются на вертикальной оси, а величины – на горизонтальной.

Столбчатая диаграмма может быть нарисована с использованием методов:

* [Axes.bar()](http://matplotlib.org/api/_as_gen/matplotlib.axes.Axes.bar.html#matplotlib.axes.Axes.bar) – вертикальная;
* [Axes.barh()](http://matplotlib.org/api/_as_gen/matplotlib.axes.Axes.barh.html#matplotlib.axes.Axes.barh) – горизонтальная.

**Class** matplotlib.lines.Axes

Функции bar() и barh() содержат ряд полезных параметров:

Таблица 2

|  |  |
| --- | --- |
| Параметр | Описание |
| left | координата *x* левого края каждого прямоугольника (столбца) |
| height | высота каждого прямоугольника |
| width | ширина столбцов (число или список) |
| bottom | начальное значение столбцов (по умолчанию 0) –координата *y* нижнего края каждого прямоугольника |
| align | выравнивание столбцов относительно риски: {'center',  'edge'} (по умолчанию 'center') |
| alpha | степень прозрачности (число от 0 до 1) |
| color | цвет столбцов |
| edgecolor | цвет границы |
| linewidth | толщина линии (вокруг столбца) |
| xerr, yerr | отображение величины погрешности (ошибки) для столбцов по горизонтали и по вертикали (число или список) |
| ecolor | цвет рисок линий погрешностей |
| log | True/False (включение/выключение логарифмического масштаба) |
| orientation | ориентация столбцов: {'vertical', 'horizontal'} |
| tick\_label | подпись по оси OX для каждого прямоугольника |

В самом простом варианте функции bar(x, y)нужно передать список значений для столбцов *x* и значения высот каждого столбца *y*.

**Пример 7.** Отобразим выпадение осадков за 3 месяца: в июне выпало 10 мм, в июле – 15 мм, в августе – 21 мм.

plt.bar([6, 7, 8],[10, 15, 21])

plt.show()

Первый массив содержит номера месяцев, а второй массив – значения показателей.

**Пример 8.** Прямоугольники в примере 7 построены вертикально, но их можно отображать и в горизонтальном виде.

plt.barh([6, 7, 8],[10, 15, 21])

plt.show()

**Пример 9.** Столбчатые диаграммы могут отображать несколько наборов данных, что очень удобно для их сравнения.

plt.bar([6, 7, 8],[10, 15, 21])

plt.bar([6, 7, 8],[6, 12, 21])

plt.show()

Такой график мог бы отображать летние осадки за два года. При этом, прямоугольники строятся поверх друг друга.

**Пример 10.** Укажем небольшое смещение по оси x:

plt.bar([6, 7, 8],[10, 15, 21])

plt.bar([6.5, 7.5, 8.5],[6, 12, 21])

plt.show()

**Пример 11.** Сузим прямоугольники и расположим их без наложения друг на друга: для этого изменим немного элементы первых массивов:

plt.bar([5.9, 6.9, 7.9],[10, 15, 21], width = 0.2)

plt.bar([6.1, 7.1, 8.1],[6, 12, 21], width = 0.2)

plt.show()

**Пример 12.** Зададим: 1) подписи к столбцам в цикле; 2) набор *y* – высоту столбиков – определим случайным числом. По умолчанию функции bar() и barh() разбивают весь интервал на равных 10 диапазонов.

x = [f'H{i+1}' for i in range(10)]

y = np.random.randint(1, 5, len(x))

f, ax = plt.subplots()

ax.bar(x, y)

plt.grid()

#plt.show()

#x

#y

**Пример 13.** Отобразим распределение нормальной случайной величины с помощью столбчатой диаграммы. Сначала сформируем сами величины и разобьем весь интервал на 10 равных диапазонов:

y = np.random.normal(0, 2, 500)

x = np.linspace(np.min(y), np.max(y), 10)

Подсчитаем, сколько величин попало в соответствующий диапазон и выведем список bars с помощью функции bar():

bars= [len(y[np.bitwise\_and(y >= x[i], y < x[i+1])]) **for** i **in** range(len(x)-1)]

ax.bar(range(len(x)-1), bars)

***После исполнения программного кода ничего не выводится –отсутствует следующая инструкция:***

fig, ax = plt.subplots()

***Не заданы области*** fig ***и*** ax.

**Пример 14.** Если нам нужно отображать столбики относительно оси ординат, то для этого существует функция barh(), которая работает аналогично функции bar():

f, ax = plt.subplots()

ax.barh(range(len(x)-1), bars)

y = np.random.normal(0, 2, 500)

x = np.linspace(np.min(y), np.max(y), 10)

bars = [len(y[np.bitwise\_and(y >= x[i], y < x[i+1])]) for i in range(len(x)-1)]

ax.barh(range(len(x)-1), bars)

**Пример 15.** Для примера, воспользуемся некоторыми параметрами из таблицы 2:

width – ширина столбцов;

linewidth – толщина линии вокруг столбца;

edgecolor – цвет границы;

yerr – отображение величины погрешности (ошибки) для столбцов по вертикали;

bottom – начальное значение столбцов (по умолчанию 0) – координата *y* нижнего края каждого прямоугольника.

x = [f'H{i+1}' **for** i **in** range(10)]

y = np.random.randint(-20, 20, len(x))

f, ax = plt.subplots()

ax.bar(x, y, width = 0.5, linewidth = 2, edgecolor = 'r', yerr = 2, bottom = 10)

ax.grid()

**Пример 16.** Сформируем столбчатые диаграммы с помощью параметра width следующим образом:

x = np.arange(10)

y1 = np.random.randint(3, 20, len(x))

y2 = np.random.randint(3, 20, len(x))

w = 0.3

f, ax = plt.subplots()

ax.bar(x - w/2, y1, width=w)

ax.bar(x + w/2, y2, width=w)

**Пример 17.** Зададим цвет каждого отдельного прямоугольника, указав в параметре color массив цветов:

x =np.arange(1, 8)

y =np.random.randint(1, 10, size = 7)

fig, ax =plt.subplots()

color\_rectangle = np.random.rand(7, 3) # RGB

ax.bar(x, y, color = color\_rectangle)

fig.set\_figwidth(12) # ширинаи

fig.set\_figheight(6) # высота "Figure"

fig.set\_facecolor('floralwhite')

ax.set\_facecolor('seashell')

plt.show()

**Пример 18.**

Matplotlib поддерживает разные цветовые модели и форматы. Задавая цвет в модели RGBA вы можете контролировать прозрачность прямоугольников, что позволяет размещать несколько наборов данных без смещения:

x =np.arange(1, 8)

y1 =np.random.randint(1, 10, size = 7)

y2 =np.random.randint(1, 10, size = 7)

fig, ax =plt.subplots()

color\_rectangle = np.random.rand(7, 3) # RGB

ax.bar(x, y1, color = color\_rectangle, width = 0.5)

color\_rectangle = np.random.rand(7, 4) # RGBA

color\_rectangle[:,3] = 0.5

ax.bar(x, y2, color = color\_rectangle)

fig.set\_figwidth(12) # ширина и

fig.set\_figheight(6) # высота "Figure"

fig.set\_facecolor('floralwhite')

ax.set\_facecolor('seashell')

plt.show()

**Пример 19.** Выделим границу прямоугольников, задав ее цвет (edgecolor) и толщину (linewidth):

x = np.arange(1, 8)

y = np.random.randint(1, 10, size = 7)

fig, ax = plt.subplots()

ax.bar(x, y,

color ='chartreuse', edgecolor='darkblue',

linewidth =5)

fig.set\_figwidth(12) # ширина и

fig.set\_figheight(6) # высота "Figure"

ax.set\_facecolor('blue')

plt.show()

**Пример 20.** В листинге и на рисунке приведен пример создания столбчатой диаграммы.

|  |  |
| --- | --- |
|  | title = "Валовой внутренний продукт РФ: в ценах 2011г. (2011-2016 гг.)"  fig,(ax1, ax2)=plt.subplots(ncols=2)  fig.canvas.set\_window\_title(title)  fig.suptitle(title+  "**\n**http://www.gks.ru/free\_doc/new\_site/vvp/vvp-god/tab2a.xls")  *# Настройки диаграммы и осей*  ax1.set\_xlabel("Год")  ax1.set\_ylabel("В ценах 2011 г., млрд.руб.")  ax2.set\_xlabel("В ценах 2011 г., млрд.руб.")  ax2.set\_ylabel("Год")  *# http://www.gks.ru/free\_doc/new\_site/vvp/vvp-god/tab2a.xls*  data=[  [2011, 59698.1],  [2012, 61798.3],  [2013, 62588.9],  [2014, 63038.4],  [2015, 61249.4],  [2016, 61097.5]  ]  size = [x[1] **for** x **in** data]  nums = [x+1 **for** x **in** range(len(size))]  tick\_label=[x[0] **for** x **in** data]  ax1.bar(nums, size, tick\_label = tick\_label,width = 0.5, color = "#a500ff")  ax2.barh(nums, size, tick\_label = tick\_label, height = 0.7, color = "#ffa500")  plt.show() |

### [3. Гистограмма](https://www.yuripetrov.ru/edu/python/ch_12_01.html#id94)

Гистограммы графически отображают распределение значений непрерывных переменных, разделяя диапазон значений на заданное количество отрезков по оси OX и отображая частоту значений на каждом отрезке по оси OY.

В самом простом случае, гистограмма – это множество прямоугольников, площадь которых (или высота) пропорциональна какой-нибудь величине.

Гистограмма может быть нарисована с использованием метода [Axes.hist()](http://matplotlib.org/api/_as_gen/matplotlib.axes.Axes.hist.html" \l "matplotlib.axes.Axes.hist" \t "_blank).

Class matplotlib.lines.Axes

matplotlib.pyplot.hist(x, bins=None, range=None, density=None, weights=None, cumulative=False, bottom=None, histtype='bar', align='mid', orientation='vertical', rwidth=None, log=False, color=None, label=None, stacked=False, normed=None, \*, data=None, \*\*kwargs)[source]

рисует гистограмму по данным х.

Параметры:

* **x** (набор числовых значений (одномерный или двумерный)) – входные данные;
* **bins** (int или набор данных) – количество интервалов разбиения;
* **align** ([str](https://www.yuripetrov.ru/edu/python/ch_03_01.html" \l "id41" \o "str)) – выравнивание прямоугольника гистограммы:
  + **"left"** (по левому краю);
  + **"mid"** (по центру – по умолчанию);
  + **"right"** (по правому краю).
* **orientation** ([str](https://www.yuripetrov.ru/edu/python/ch_03_01.html" \l "id41" \o "str)) – ориентация гистограммы:
  + **"horizontal"** (горизонтально);
  + **"mid"** (вертикально – по умолчанию).
* **color** – набор значений цветов данных;
* **label** – набор подписей для данных.

**Результат** – кортеж:

* **n**: массив длин прямоугольников или список таких массивов (если было передано несколько наборов данных);
* **bins**: массив – границы прямоугольников;
* **patches**: набор вложенных массивов – индивидуальных параметров прямоугольников в случае построения нескольких наборов данных.

**Пример 21.** Пример создания гистограммы.

|  |  |
| --- | --- |
|  | title = "Распределение сумм кредитов"  fig,ax = plt.subplots()  fig.canvas.set\_window\_title(title)  *# Настройки диаграммы и осей*  ax.set\_title(title)  ax.set\_xlabel("Сумма")  ax.set\_ylabel("Количество")  *# Суммы кредитов, взятых в банке*  data = [40000,120000,90000,160000,290000,250000,90000,260000,170000,11000,500000,250000]  bins\_to\_be = 3 *# Количество интервалов разбиения*  n, bins, patches = ax.hist(data, bins = bins\_to\_be, color = "brown", edgecolor = "black")  *# Вывод итоговых данных в легенду*  res = ""  **for** i **in** range(bins\_to\_be):  res += "Группа №**{}**, **{:.0f}** чел., от **{:.2f}** руб.**\n**".\  format(i+1, n[i], bins[i])  ax.legend([res.strip()])  plt.show() |

**Пример 22.** На гистограммах можно указывать погрешность измерения величины, как по горизонтали (xerr) так и вертикали (yerr):

# Горизонтальная ошибка

x = np.arange(1, 8)

y =np.random.randint(1, 10, size =7)

fig, ax =plt.subplots()

y\_error=np.random.randint(5, 20, size = (2, 7))/15

ax.bar(x, y,

yerr = y\_error, # границы погрешностей

ecolor = 'darkred', # цвет линии погрешности

capsize = 10, # горизонтальная черточка

edgecolor = 'red', # цвет края прямоугольника

linewidth = 2, # ширина крайней линии

color = 'seashell', # цвет прямоугольника

linestyle = '--') # начертание линии

fig.set\_figwidth(12) # ширина и

fig.set\_figheight(6) # высота "Figure"

fig.set\_facecolor('floralwhite')

ax.set\_facecolor('seashell')

plt.show()

**Пример 23. Выполнить самостоятельно:**

Сгенерируем N случайных чисел с нормальным (гауссовым) распределением (среднее – 0, среднеквадратичное отклонение – 1), и раскидаем их по 20 бинам от −3 до 3 (точки за пределами этого интервала отбрасываются). Для сравнения, вместе с гистограммой нарисуем Гауссову кривую в том же масштабе и напишем формулу Гаусса.

**1) Вставить необходимые инструкции.**

**2) Исправить ошибки в программном коде.**

N=10000

r=normal(size=N)

n,bins,patches=hist(r,range=(-3,3),bins=20)

x=linspace(-3,3,100)

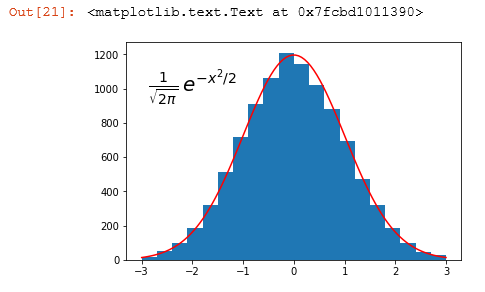
plot(x,N/sqrt(2\*pi)\*0.3\*exp(-0.5\*x\*\*2),'r')

text(-2,1000,r'$\frac{1}{\sqrt{2\pi}}\,e^{-x^2/2}$',

fontsize=20,horizontalalignment='center',

verticalalignment='center')

**3) Прикрепить выполненные задания в соответствующем разделе.**



**Образец рисунка, который должен получиться.**