**Работа с библиотекой Matplotlib**

**Теоретическая часть**

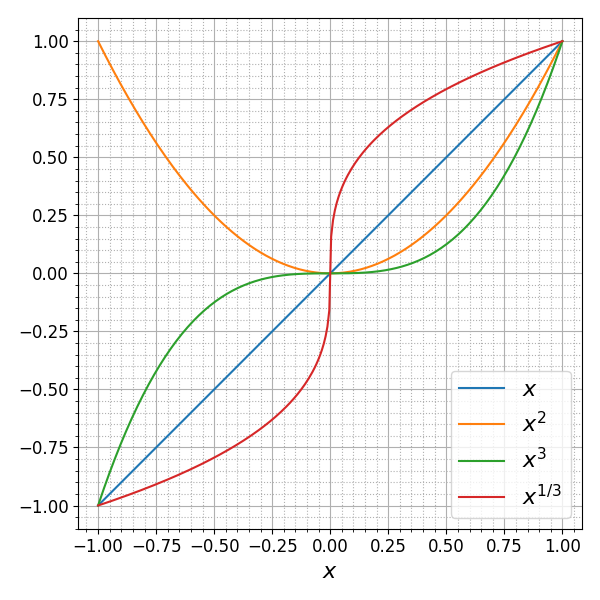
**Размер изображения**

До сих пор мы строили графики в одном окне, размер которого был задан по умолчанию. За кадром matplotlib создавал объект типа Figure, который определяет размер окна и содержит все остальные элементы. Кроме того, автоматически создавался объект типа Axis. Подробнее работа с этими объектами будет рассмотрена ниже. Сейчас же мы рассмотрим функцию pyplot.figure, которая позволяет создавать новые объекты типа Figure и переключаться между уже созданными объектами.

Функция pyplot.figure может принимать множество аргументов. Вот основные:

* num: int или str — уникальный идентификатор объекта типа. Если задан новый идентификатор, то создается новый объект и он становится активным. В случае, если передан идентификатор уже существующего объекта, то этот объект возвращается и становится активным
* /media//media/figsize: (float, float) — размер изображения в дюймах
* dpi: float — разрешение в количестве точек на дюйм

Используем функцию pyplot.figure и функцию pyplot.axis чтобы улучшить наш пример с построением степенных функций:



Мы добавили две строки по сравнению с прошлой версией:

fig = plt.figure(/media//media/figsize=(6, 6))

*# ...*

plt.axis('equal')

Функция pyplot.axis позволяет задавать некоторые свойства осей. Ее вызов с параметром 'equal' делает одинаковыми масштабы вертикальной и горизонтальной осей, что кажется хорошей идеей в этом примере. Функция pyplot.axis возвращает кортеж из четырех значений xmin, xmax, ymin, ymax, соответствующих границам диапазонов значений осей.

Некоторые другие способы использования функции pyplot.axis:

* Кортеж из четырех float задаст новые границы диапазонов значений осей
* Строка 'off' выключит отображение линий и меток осей

**Гистограммы**

Функция pyplot.hist строит гистограмму по набору значений:

import numpy as np

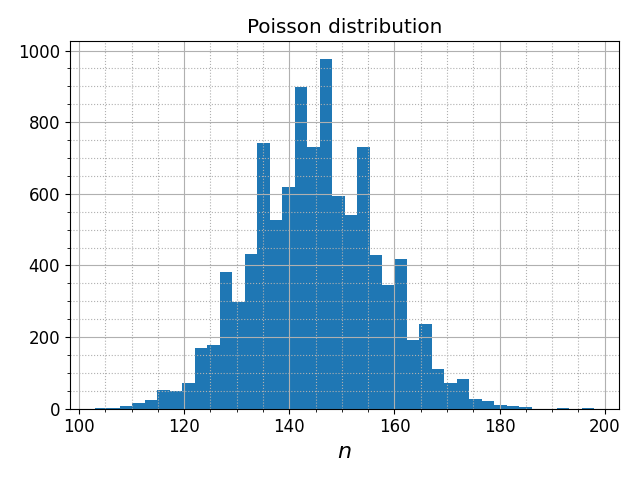
import matplotlib.pyplot as plt

rg = np.random.Generator(np.random.PCG64(5))

data = rg.poisson(145, 10000)

plt.hist(data, bins=40)

*# для краткости мы опускаем код для настройки осей, сетки и т.д.*

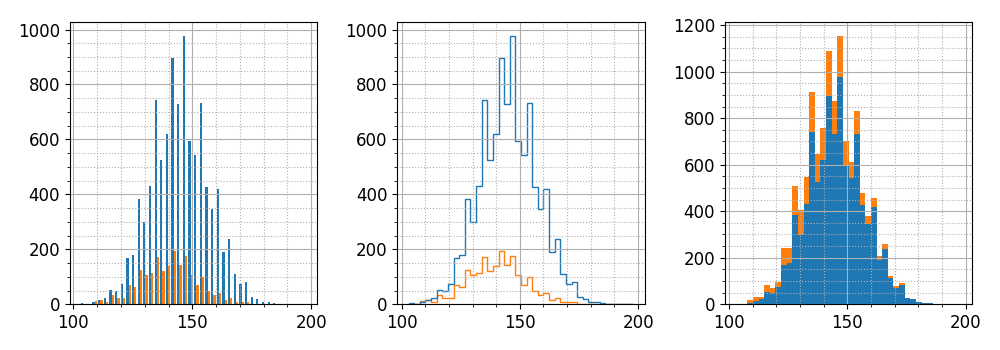


Аргумент bins задает количество бинов гистограммы. По умолчанию используется значение 10. Если вместо целого числа в аргумент bins передать кортеж значений, то они будут использованы для задания границ бинов. Таким образом можно построить гистограмму с произвольным разбиением.

Некоторые другие аргументы функции pyplot.hist:

* range: (float, float) — диапазон значений, в котором строится гистограмма. Значения за пределами заданного диапазона игнорируются.
* density: bool. При значении True будет построена гистограмма, соответствующая плотности вероятности, так что площадь гистограммы будет равна единице.
* weights: список float значений того же размера, что и набор данных. Определяет вес каждого значения при построении гистограммы.
* histtype: str. может принимать значения {'bar', 'barstacked', 'step', 'stepfilled'}. Определяет тип отрисовки гистограммы.

В качестве первого аргумента можно передать кортеж наборов значений. Для каждого из них будет построена гистограмма. Аргумент stacked со значением True позволяет строить сумму гистограмм для кортежа наборов. Покажем несколько примеров:



rg = np.random.Generator(np.random.PCG64(5))

data1 = rg.poisson(145, 10000)

data2 = rg.poisson(140, 2000)

*# левая гистограмма*

plt.hist([data1, data2], bins=40)

*# центральная гистограмма*

plt.hist([data1, data2], bins=40, histtype='step')

*# правая гистограмма*

plt.hist([data1, data2], bins=40, stacked=**True**)

В физике гистограммы часто представляют в виде набора значений с ошибками, предполагая при этом, что количество событий в каждом бине является случайной величиной, подчиняющейся биномиальному распределению. В пределе больших значений флуктуации количества событий в бине могут быть описаны распределением Пуассона, так что характерная величина флуктуации определяется корнем из числа событий. Библиотека matplotlib не имеет инструмента для такого представления данных, однако его легко получить с помощью комбинации numpy.histogram и pyplot.errorbar:

**def** poisson\_hist(data, bins=60, lims=**None**):

*""" Гистограмма в виде набора значений с ошибками """*

hist, bins = np.histogram(data, bins=bins, range=lims)

bins = 0.5 \* (bins[1:] + bins[:-1])

**return** (bins, hist, np.sqrt(hist))

rg = np.random.Generator(np.random.PCG64(5))

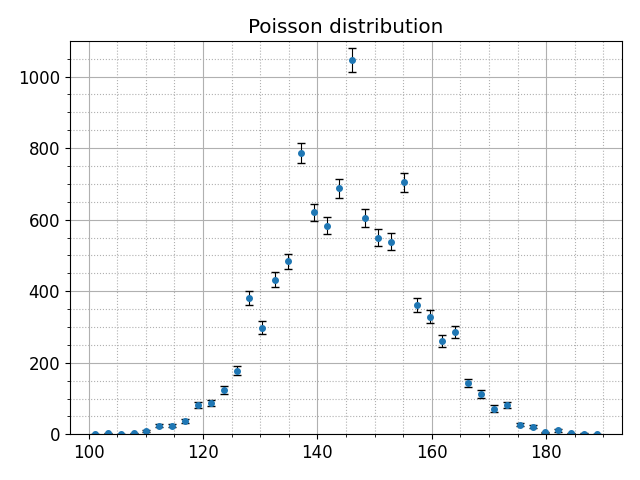
data = rg.poisson(145, 10000)

x, y, yerr = poisson\_hist(data, bins=40, lims=(100, 190))

plt.errorbar(x, y, yerr=yerr, marker='o', markersize=4,

linestyle='none', ecolor='k', elinewidth=0.8,

capsize=3, capthick=1)



**Диаграммы рассеяния**

Распределение событий по двум измерениям удобно визуализировать с помощью диаграммы рассеяния:

rg = np.random.Generator(np.random.PCG64(5))

means = (0.5, 0.9)

covar = [

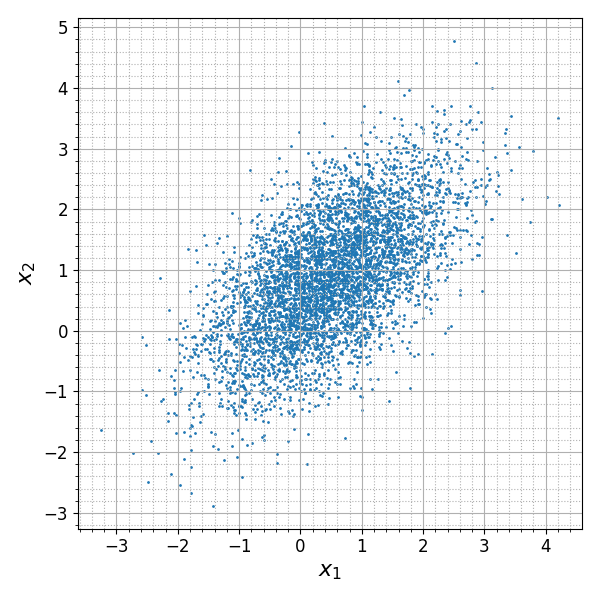
[1., 0.6],

[0.6, 1.]

]

data = rg.multivariate\_normal(means, covar, 5000)

plt.scatter(data[:,0], data[:,1], marker='o', s=1)



Каждой паре значений в наборе данных соответствует одна точка на диаграмме. Несмотря на свою простоту, диаграмма рассеяния позволяет во многих случаях наглядно представлять двумерные данные. Функция pyplot.scatter позволяет визуализировать и данные более высокой размерности: размер и цвет маркера могут быть заданы для каждой точки отдельно:

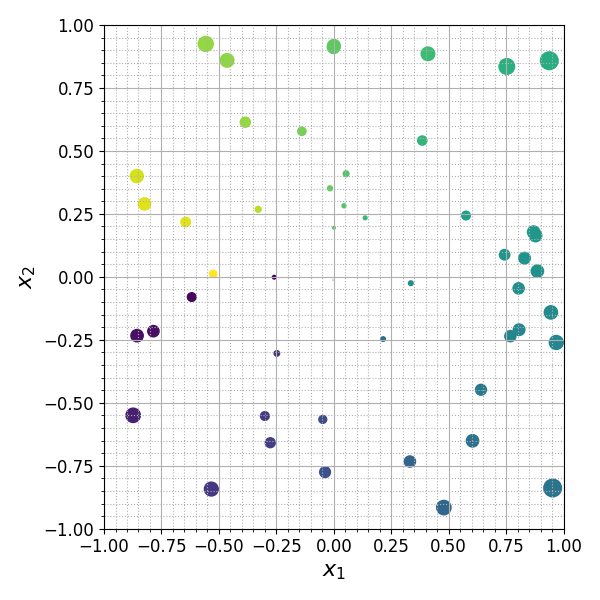
rg = np.random.Generator(np.random.PCG64(4))

data = rg.uniform(-1, 1, (50, 2))

col = np.arctan2(data[:, 1], data[:, 0])

size = 100\*np.sum(data\*\*2, axis=1)

plt.scatter(data[:,0], data[:,1], marker='o', s=size, c=col)



Цветовую палитру можно задать с помощью аргумента cmap. Подробности и описание других аргументов функции pyplot.scatter можно найти в [документации](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.scatter.html).

**Задание (с использованием библиотеки pyplot)**

1. Импортируйте данные из файла CSV «Experiment - obval2.txt».
   1. Постройте графики изменения значений во времени (четные варианты – столбцы device2,4,6,8; нечетные - столбцы device1,3,5,7). Данные рисунки должны быть трех различных физических размеров. Второй больше первого на 20%, четвертый больше второго на 20% и т.д. Ограничьте ось Y значениями ±3.
2. Импортируйте данные из файла CSV «city\_russia.csv».
   1. Постройте гистограмму по столбцу «population» (четные варианты) или foundation\_year (нечетные варианты).
   2. По этим же данным построить гистогграмму плотности вероятности.
3. Постройте гистограмму, похожую на гистограмму на рис. ниже.



1. По данным из файлов oscar\_age\_male.csv и oscar\_age\_female.csv (столбец Age) построить одновременно две гистограммы на одном рисунке. Выбрать цвет для гистограмм: для женщин – зеленый, для мужчин – желтый.
2. Сгенерировать набор данных следующего вида:

|  |  |
| --- | --- |
| Температура | Погрешность |
| 0 |  |
| 5 |  |
| 6 |  |
| 12 |  |
| 18 |  |
| 16 |  |
| 6 |  |
| 10 |  |
| 11 |  |

Значения в столбце «погрешность» = ±n, где n – последняя цифра в логине студента.

* 1. Построить по данным на которой будут отмечены для каждого бина возможные флуктуации измерений (погрешность).

1. Для набора данных mlb\_players.csv
   1. построить диаграмму рассеяния по значениям: Height(inches) и Age (четные варианты) или "Weight(lbs)" и "Age" (нечетные варианты).
   2. Построить диаграмму рассеяния для столбцов Height(inches) и Weight(lbs), где Age будет задаваться цветом (четные вариантs) или размером (нечетные варианты).
2. Построить диаграмму рассеяния, которая генерировалась бы из 1000 точек. В итоге диаграмма должна иметь форму окружности.