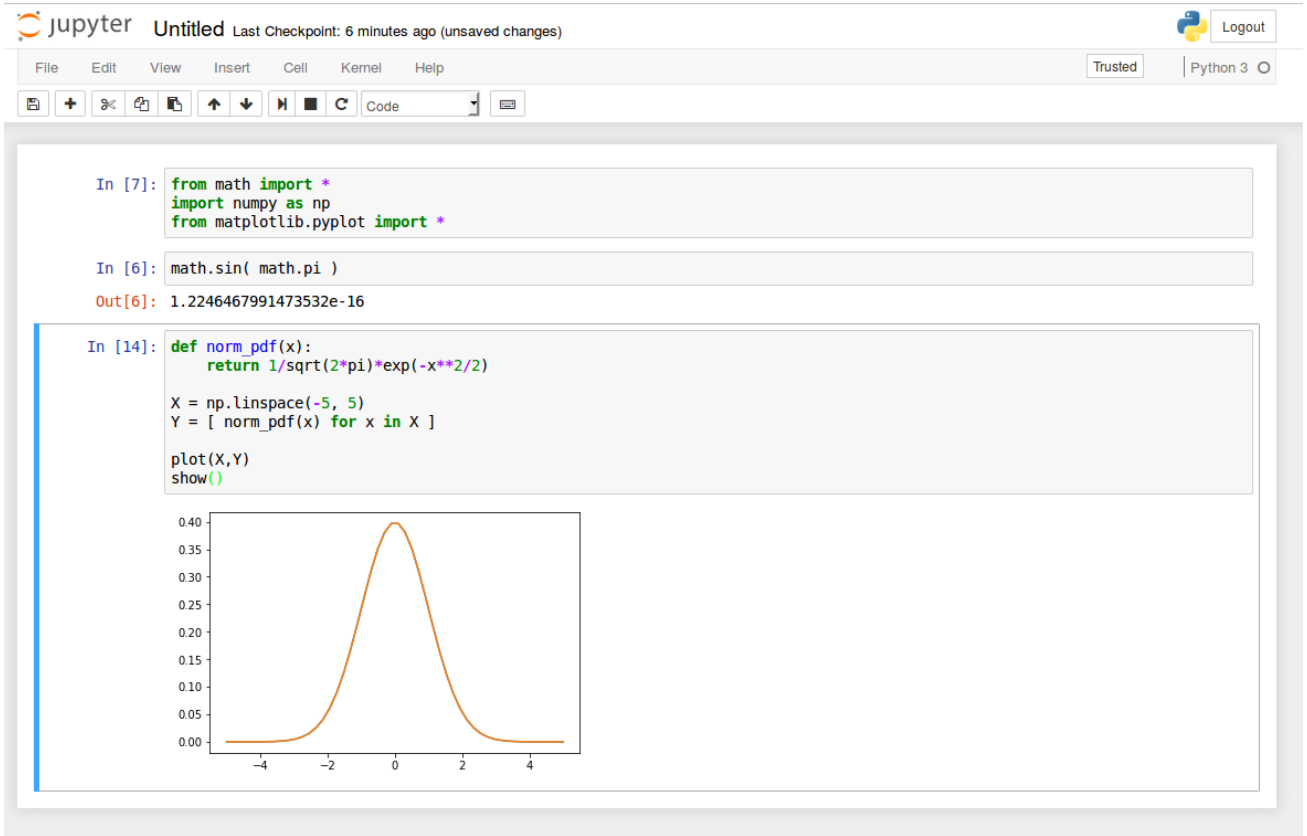
Визуализация математических данных в Python

2018

План

Outline

Jupyter



Для экспериментов с визуализацией (и не только) хорошо подходит Jupyter - интерактивная оболочка для Python, которая позволяет хранить код, результат работы программы, изображения, графики, latex в одном месте.

Jupyter

Использовать Jupyter можно и без установки, на сайте проекта есть[1] онлайн-версия: http://jupyter.org/try

Jupyter входит в дистрибутив **Anaconda**.

Помимо Jupyter туда входят популярные метематические пакеты для Python, не включенные в стандартную библиотеку языка.

Доклад: как использовать Jupyter на 100%
https://www.youtube.com/watch?v=q4d-hKCpTEc

[1] бывает недоступен во время высокой загрузки сервера

Jupyter

Некоторые горячие клавиши Jupyter:

- `Ctrl+Enter` - выполнить текущую ячейку
- `Shift+Enter` - выполнить текущую ячейку и перейти (добавить) следующую
- `Esc` - перейти в режим выделения ячеек
- `Enter` - перейти в режим редактирования выделенной ячейки
- `D,D` - (в режиме выделения) удалить текущую ячейку

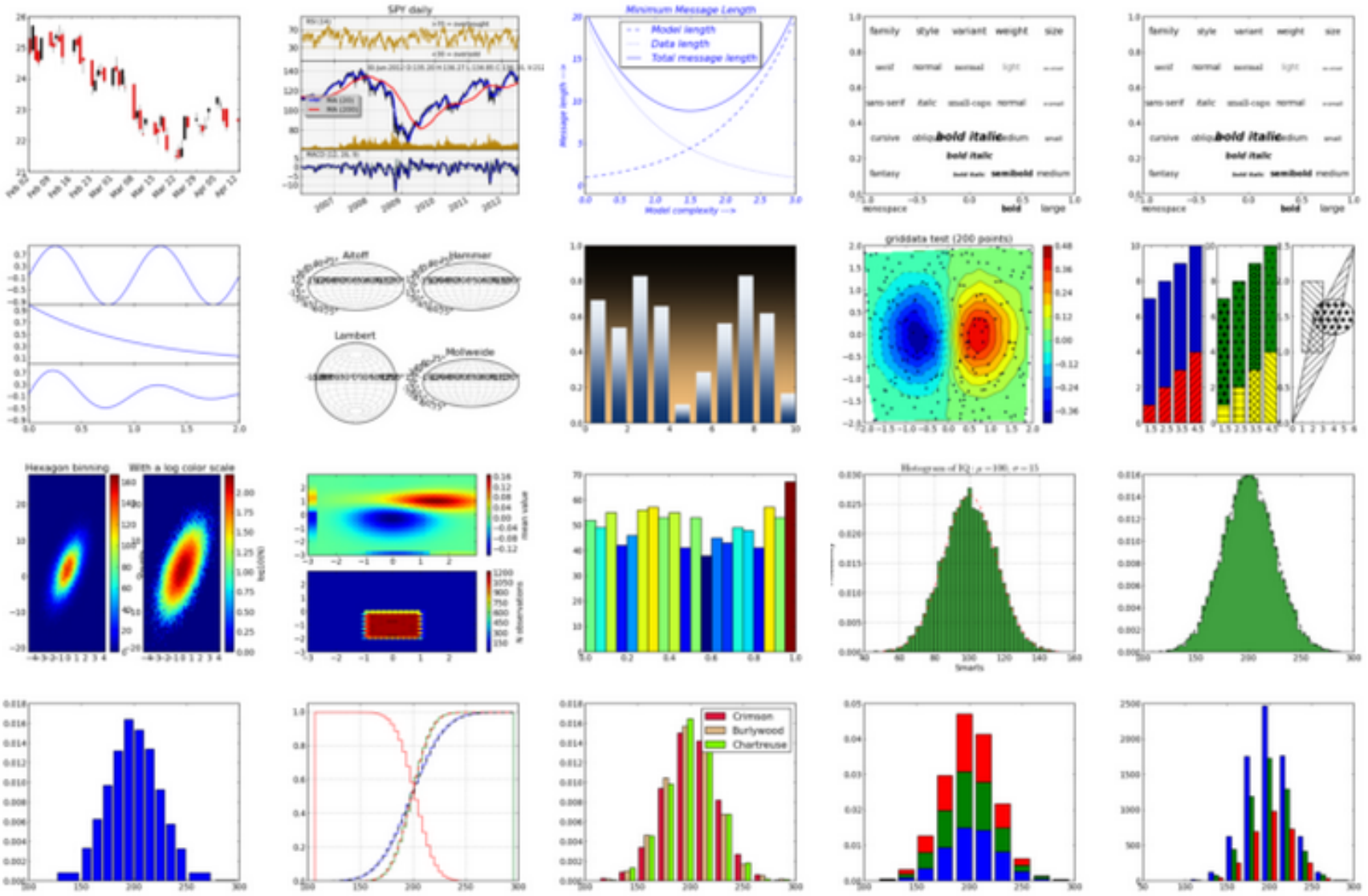
При работе могут быть полезны следующие команды (jupyter magic)
`%ls` - показать содержимое текущего каталога
`%load имя`

Некоторые популярные пакеты для визуализации данных

- **matplotlib** - построение графиков и диаграмм;
- **seaborn** - визуализация статистических данных, эстетичнее чем matplotlib;
- **mpld3** - использование D3.js для построения интерактивных matplotlib графиков в окне браузера;
- **pandas** - анализ данных: статистики, регрессия, визуализация и т.п.

Outline

Пакет matplotlib



Пакет matplotlib

Пакет **matplotlib** содержит модули предназначенные для построения диаграмм и графиков.

Модуль **pyplot** из этого пакета предназначен непосредственно для построения графиков. Этого модуля будет достаточно для построения относительно простых графиков.

Остальные модули пакета matplotlib содержат преимущественно функции для гибкой настройки вида графиков, осей, подписей к осям, компоновки нескольких графиков на одном листе и т.п.

Пакет matplotlib

```
from matplotlib.pyplot import plot
```

Дополнительно можно подключить модуль seaborn. Тогда графики станут эстетичнее

```
import seabor
```

По умолчанию графики будут показаны непосредственно в блокноте Jupyter. Если это не так, то используется команда jupyter:

```
%matplotlib inline
```

Создание графика

plot - функция с переменным числом параметров. Если некоторые параметры не указаны, то им задаётся значение по-умолчанию.

- **plot(list_of_y)** - создаёт график в памяти программы list_of_y - список ординат (значений y).

  Здесь в качестве абсцисс (значений x) используются числа 0, 1, 2 и так далее до последнего индекса y_list

Пакет matplotlib

- **plot(x-list, y-list)**

  x-list - список абсцисс (значений x).
  y-list - список ординат (знаений y).

  Длины списков x-list и y-list должны быть одинаковы.

Пакет matplotlib

Дополнительные параметры

- **plot(x-list, y-list, style)**

  style - стиль графика. Определяет цвет, вид кривой и точек

  - \- прямая линия
  - \- - пунктирная линия
  - **.** только точки
  - **v** треугольники вместо точек

  Цвета

  - **'b'** blue
  - **'g'** green
  - **'r'** red
  - **'k'** black

Дополнительную информацию о стилях см. в документации: help(plot)

Пакет matplotlib

Дополнительные параметры

- **plot(x-list, y-list, style, label)**
  label - подпись к графику. По-умолчанию не показывается.
- **plot(x-list, y-list, style, label, linewidth)**
  linewidth - толщина линии. По-умолчанию - 1.

Пакет matplotlib

```
from matplotlib.pyplot import grid, xlabel, ylabel, legend
```

- **grid(True)** - "включает"координатную сетку.
  Шаг сетки выбирается автоматически.
- **xlabel**("подпись") - добавляет название для оси x

- **ylabel**("подпись") - добавляет название для оси y
- **legend(loc)** - добавляет к графикам пояснения (легенду)
  loc - положение блока с пояснениями. По-умолчанию - справа сверху.
  Для автоматического выбора положения следует задать параметр loc = 'best'

Пакет matplotlib

```
from matplotlib.pyplot import show, savefig
```

- **show()** - создаёт и показывает окно содержащие построенные функциями plot() графики.
  Оси координат строятся строятся автоматически, масштаб выбирается в зависимости от ширины и высоты графика.
- **savefig(filename [, dpi])** - сохраняет изображение графика в файл. Графический формат определяется по расширению файла.
  filename - имя файла
  dpi - количество точек на дюйм (DPI)

Пакет matplotlib

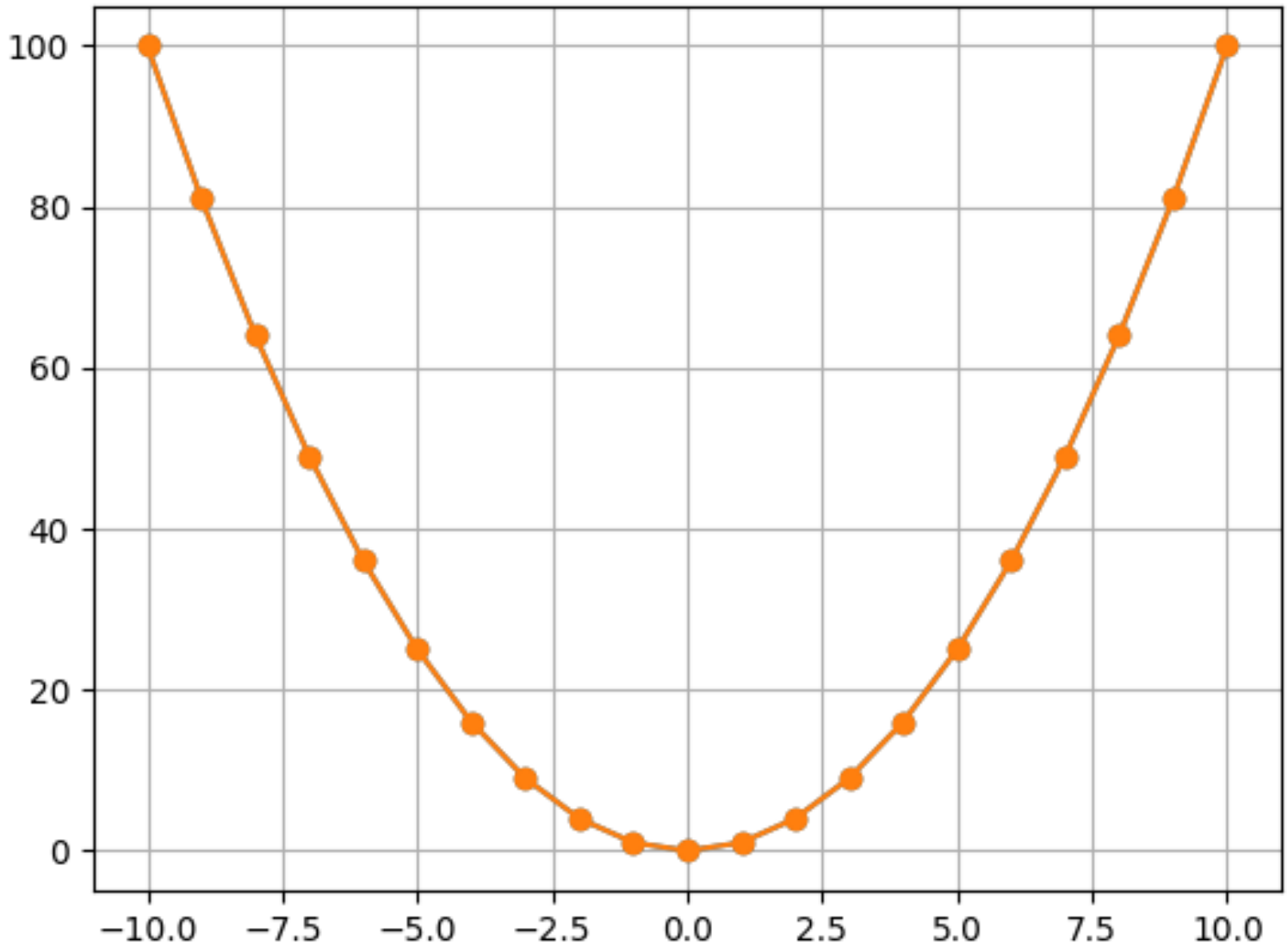
Типичный алгоритм построения графика

1. Создать графики
2. Настроить оформление графика
3. Показать или сохранить график

Примитивный пример

```
X = list(range(-10,11))
Y = [x**2 for x in X]
plot(X,Y,'-o')
grid(True)
show()
```

Примитивный пример



numpy linspace и arange

Для создания набора значений лучше использовать функции **linspace** и **arange** пакета numpy.

```
# разделить отрезок [-10, 10] на 100 точек
# x будет иметь тип numpy array
x1 = np.linspace(-10, 10, 100) -> numpy array


# разделить отрезок [-10, 10] на части с шагом 0.5
# правая граница не включается
x2 = np.arange(-10, 10.5, 0.5) -> numpy array

# numpy array даёт возможность применять функцию
# ко всем элементам массива (
y = sin(x1)  # -> numpy array
# теперь y массив синусов икса
```

Пример получше

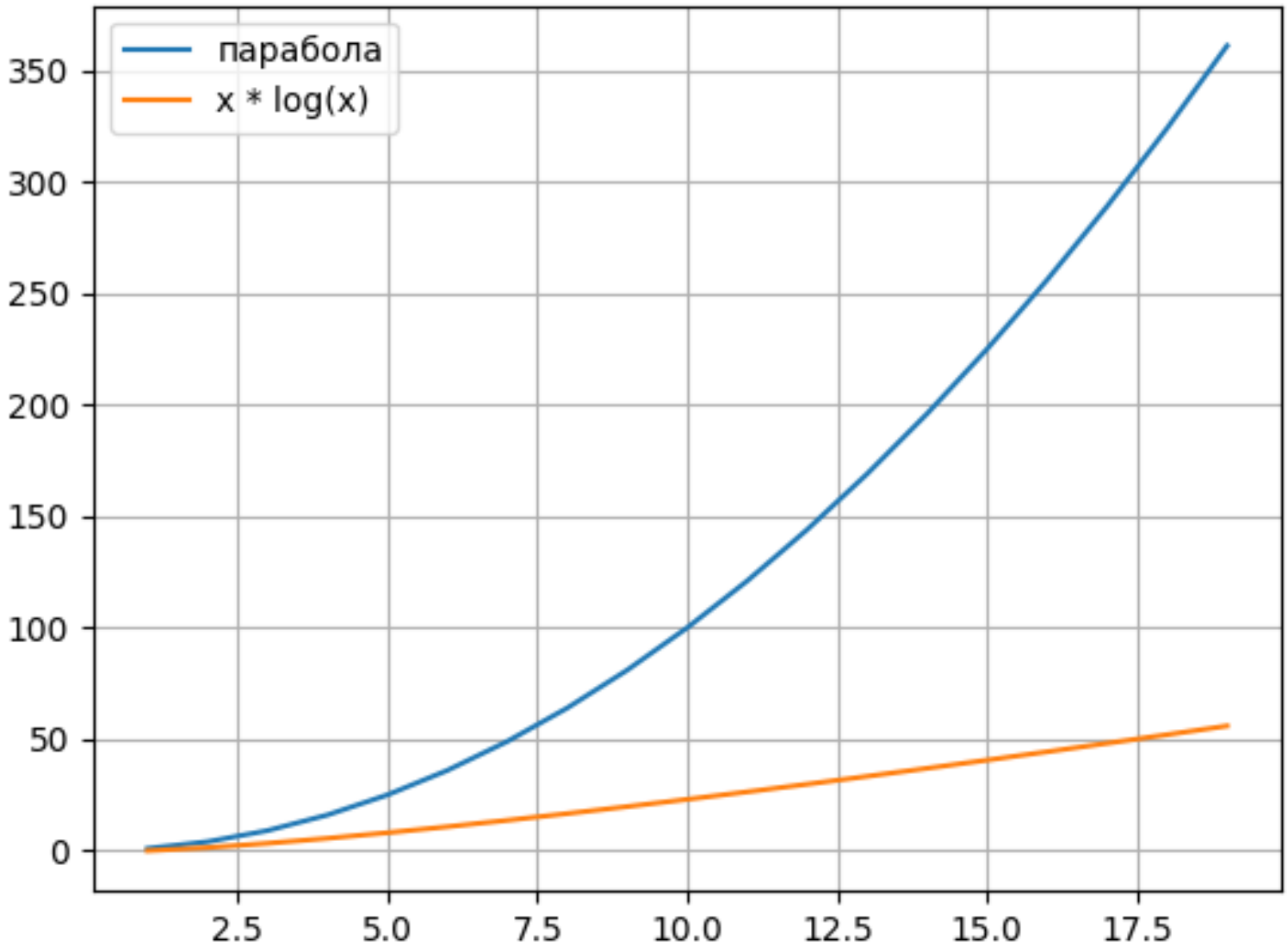
```
...
import numpy as np

# разделить отрезок [-10, 10] на 100 точек
# x будет иметь тип numpy array
x = np.linspace(-10, 10, 100)

# над целым numpy array можно производить операции
# как если бы это было отдельное значение
# в результате y будет массивом квадратов
y = x**2
plot(X,Y)
grid(True)
show()
```

Пакет matplotlib

На одном поле можно построить несколько графиков.
Чтобы их различать стоит указать для них названия.
Для этого будем явно задавать имя параметра: **label**

```
X = np.linspace(1,20, 100)
Y1 = X**2
Y2 = X * np.log(X)  #
plot(X,Y1, label = 'парабола')
plot(X,Y2, label = 'x * log(x)')
legend()
grid(True)
show()
```

Пакет matplotlib



Добавление дополнительных элементов на график

- Заголовок

  ```
  title('1a TITLE')
  ```

- Текст

  ```
  text(x,y, 'текст')
  ```

Пакет matplotlib

Matplotlib и Jupyter
По-умолчанию графики построенные в Jupyter будут показаны в ячейке вывода.

Чтобы показать их в отдельном окне, с возможностью масштабирования и перемещения следует перед построением графиков выполнить в Jupyter команду:

```
%matplotlib
```

Чтобы вернуть построение графиков в блокнот нужно выполнить команду

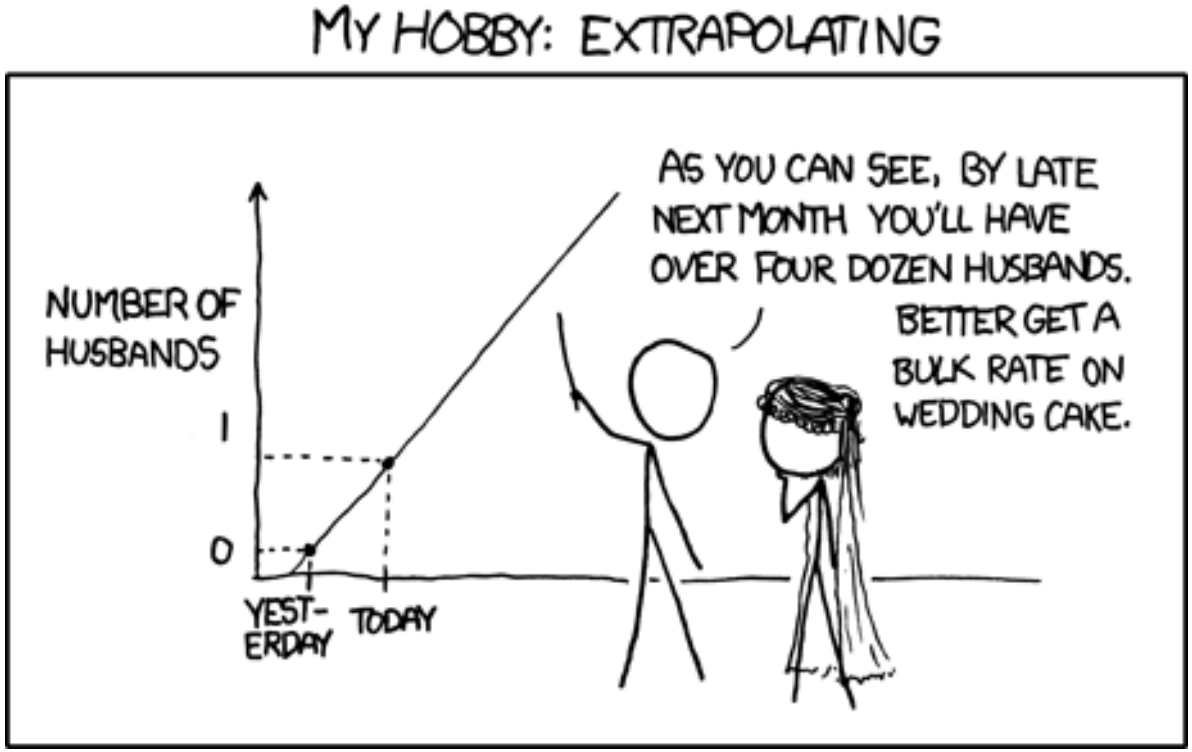
```
%matplotlib inline
```

Пакет matplotlib

Ещё примеры и документация
matplotlib.org/tutorials/introductory/pyplot.html

Бонус

Вызов функции `xkcd()` делает стиль графиков похожим на стиль веб-комикса xkcd



Outline

Векторизация функций

Для построение трёхмерных графиков не удобно пользоваться скалярными функциями, вручную вычисляя значения для каждого значения аргумента.

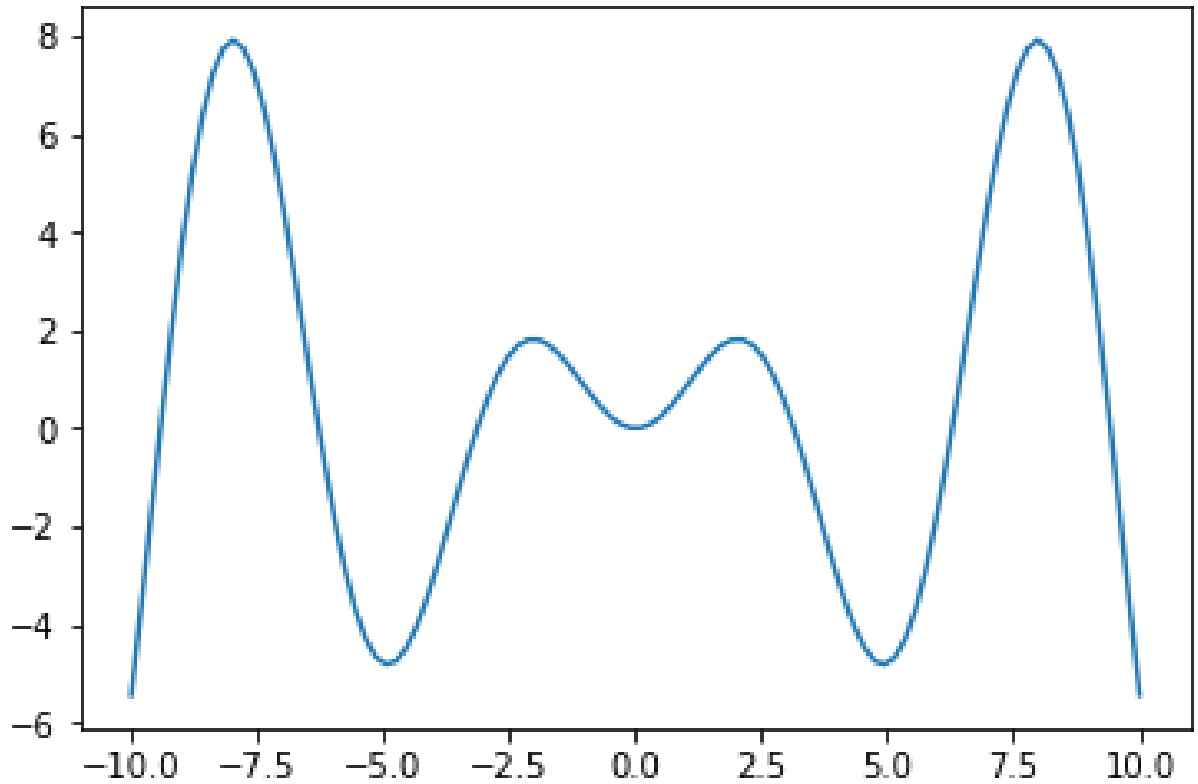
Класс **vectorize** пакета numpy может быть использован для "векторизации"функции. Такая функция может принимать в качестве параметра array и возвращать array.

```
from math import *
import numpy as np

def foo(x):
    return x*sin(x)

foo = np.vectorize(foo)
```

Векторизация функций

```
from math import *
from matplotlib.pyplot import *
import numpy as np

def foo(x):
    return x*sin(x)

foo = np.vectorize(foo)

# создание array из 100 точек отрезка [-10, 10]
x = np.linspace(-10, 10, 100)
y = foo(x)

plot(x,y)
show()
```

Векторизация функций



Векторизация функций

Вместо обычных функций можно использовать лямбда-функции

```
from math import *
import numpy as np

foo = np.vectorize(lambda x: x*sin(x))
```

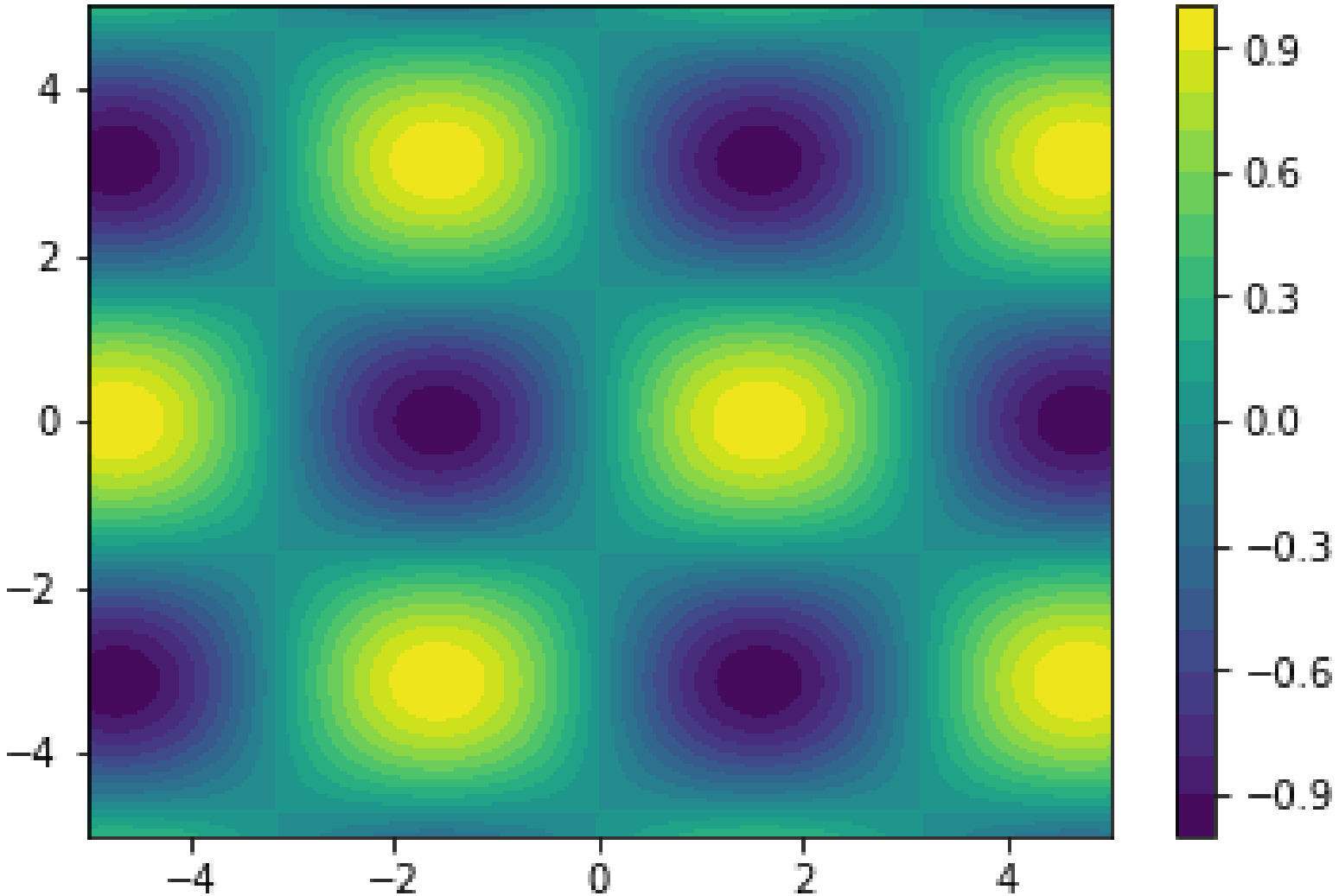
Outline

Пример

Тепловая карта с линиями уровня

```
from matplotlib.pyplot import *
import numpy as np
from math import *

X = np.linspace(-5, 5, 100)
Y = np.linspace(-5, 5, 100)

# создание сетки (декартового произведения X и Y)
xx,yy = np.meshgrid( X, Y)

foo = np.vectorize(lambda x,y: sin(x)*cos(y))
zz = foo(xx,yy)

contourf(xx, yy, zz, 20)
colorbar()
show()
```

Пример

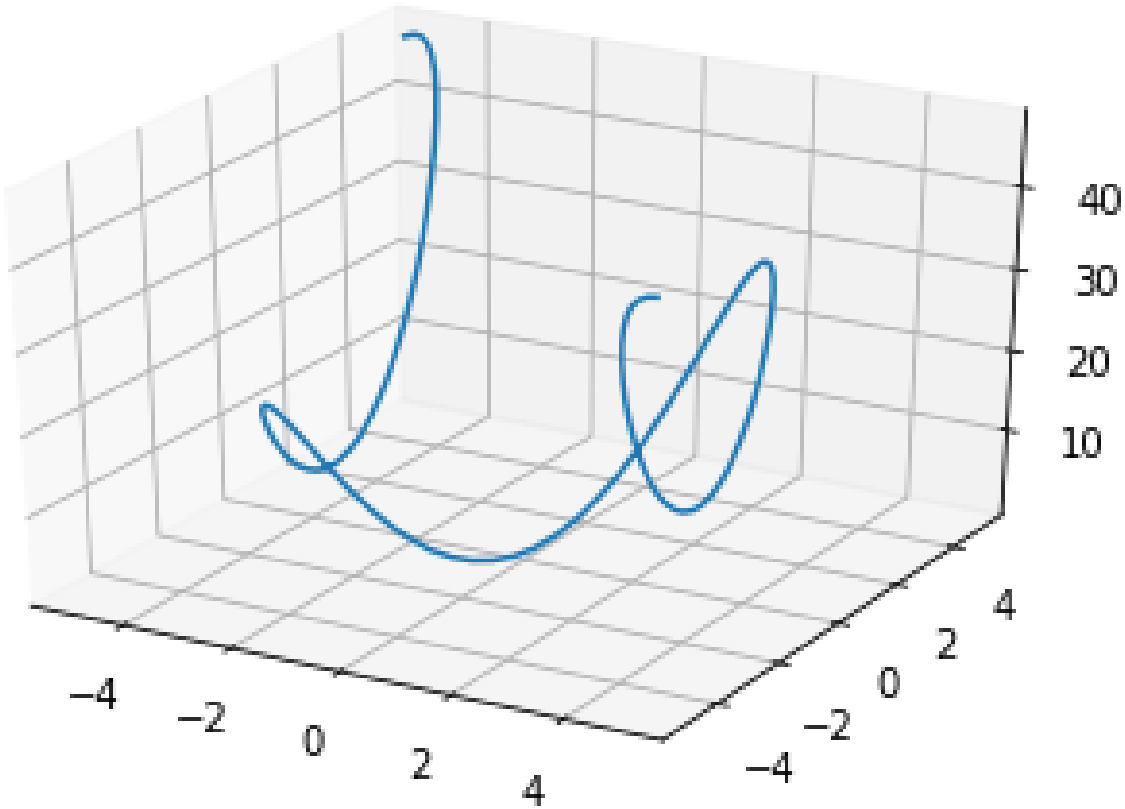


Ещё примеры:
jakevdp.github.io/PythonDataScienceHandbook/04.04-density-and-contour-plots.html

Outline

Графики в изометрии

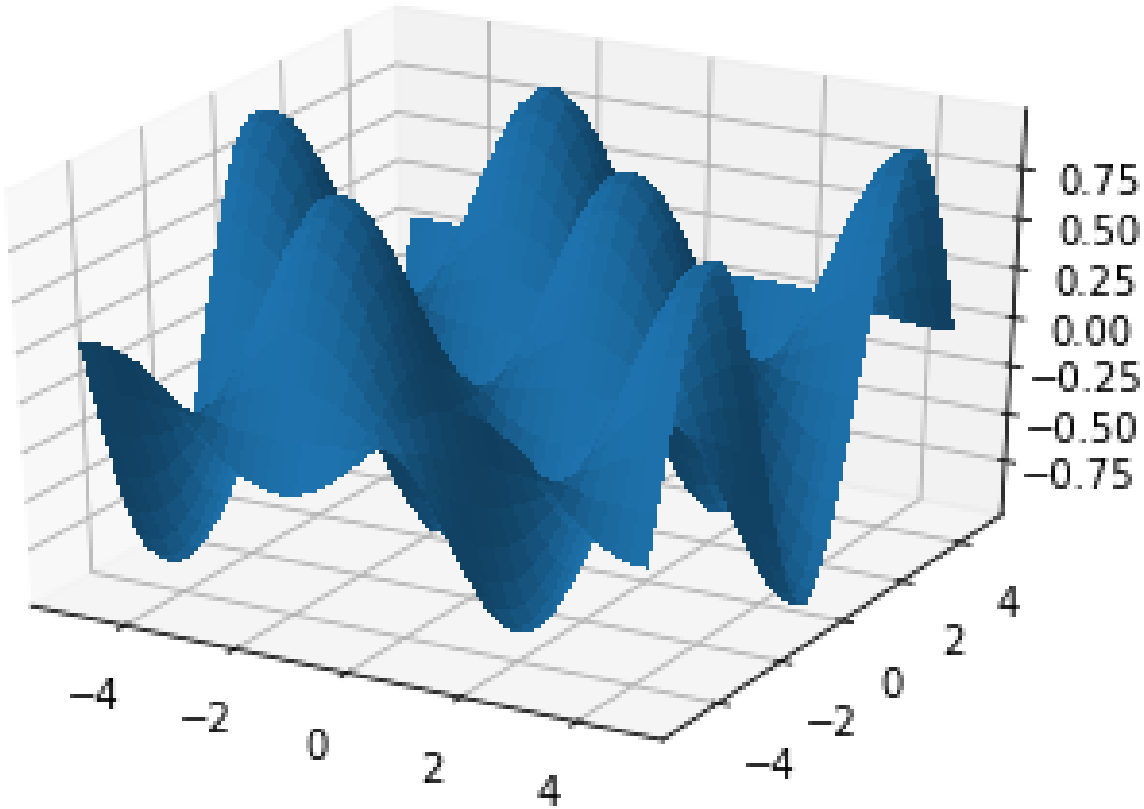
```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

fig = plt.figure()
ax = fig.gca(projection='3d')

x = np.linspace(-5, 5, 100)
y = np.sin(x)*5
z = x**2 + y**2

ax.plot(x, y, z, antialiased=True)

plt.show()
```


40
30
20
10
−4
−2
0
2
4
−4
−2
0
2
4

Графики в изометрии

График кривой в пространстве

```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import numpy as np
from math import *

fig = plt.figure()
ax = fig.gca(projection='3d')

X = np.arange(-5, 5, 0.25)
Y = np.arange(-5, 5, 0.25)
X, Y = np.meshgrid(X, Y)

foo = np.vectorize(lambda x, y: sin(x) * cos(y))
Z = foo(X,Y)

ax.plot_surface(X, Y, Z, antialiased=False)
plt.show()
```


0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−4
−2
0
2
4
−4
−2
0
2
4

Графики в изометрии

Дополнительно можно раскрасить поверхность как тепловую карту, изобразить сетку вместо поверхности и настроить друге параметры отображения.

Документации matplotlib с примерами:
matplotlib.org/mpl_toolkits/mplot3d/tutorial.html

Outline

plotly

plot.ly

plotly

plotly - модуль для создание интерактивных графиков.

Существуют версии для Python, R, Matlab

Графики представляют собой html фрейм с JavaScript (фреимворк D3.js).

Существуют два режима режима с графиками:

- оффлайн (график сохраняется как отдельный html файл)
- графики публикуются на сервере plot.ly

plotly

Для создания оффлаин-графиков нужно использовать отдельный модуль offline пакета plotly.

```
from plotly.offline import download_plotlyjs, \
init_notebook_mode, plot, iplot
from plotly.graph_objs import Scatter, Figure, Layout
import numpy as np

x = np.linspace(-10, 10, 100)
y = np.vectorize(lambda x: np.sin(x) * x )(x)
plot([Scatter(x=x, y=y)])
```

Будет создан отдельный html файл, который автоматически откроется в браузере

plotly

Чтобы графики отображались в окне Jupyter следует явно это указать

```
init_notebook_mode(connected=True)
```

и использовать функцию рисовани **iplot** вместо plot.

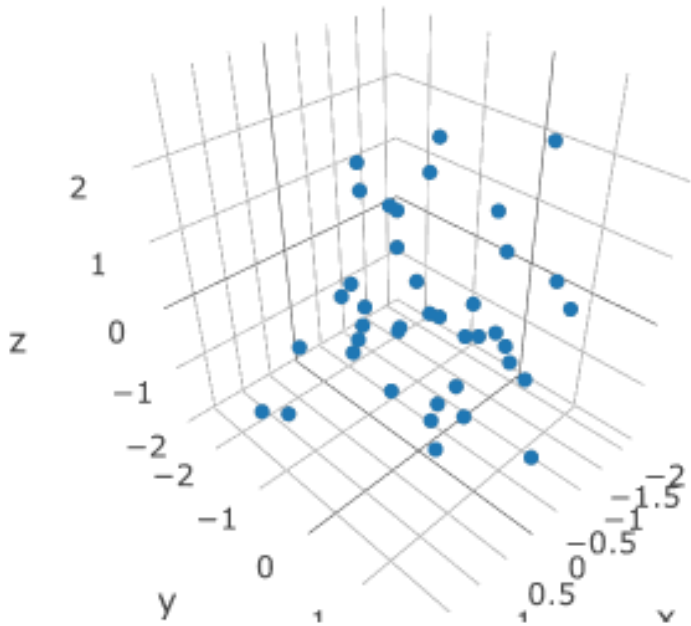
```
from plotly.offline import download_plotlyjs, \
            init_notebook_mode, plot, iplot
from plotly.graph_objs import Scatter, Figure, Layout
import numpy as np

init_notebook_mode(connected=True)

x = np.linspace(-10, 10, 100)
y = np.vectorize(lambda x: np.sin(x) * x )(x)
iplot([Scatter(x=x, y=y)])
```

plotly

Диаграмма рассеивания трёхмерной случайной величины

```
from plotly.offline import download_plotlyjs, \
        init_notebook_mode, iplot
from plotly.graph_objs import Scatter3d, Figure
import numpy as np

init_notebook_mode(connected=True)
# создание 3-х массивов заполненных 10 случайными значениями
x,y,z = np.random.uniform( size=(3, 10))

trace1 = Scatter3d(x=x, y=y, z=z, mode='markers', \
                   marker=dict(size=2))

data = [trace1]
fig = Figure(data=data)
iplot(fig)
```

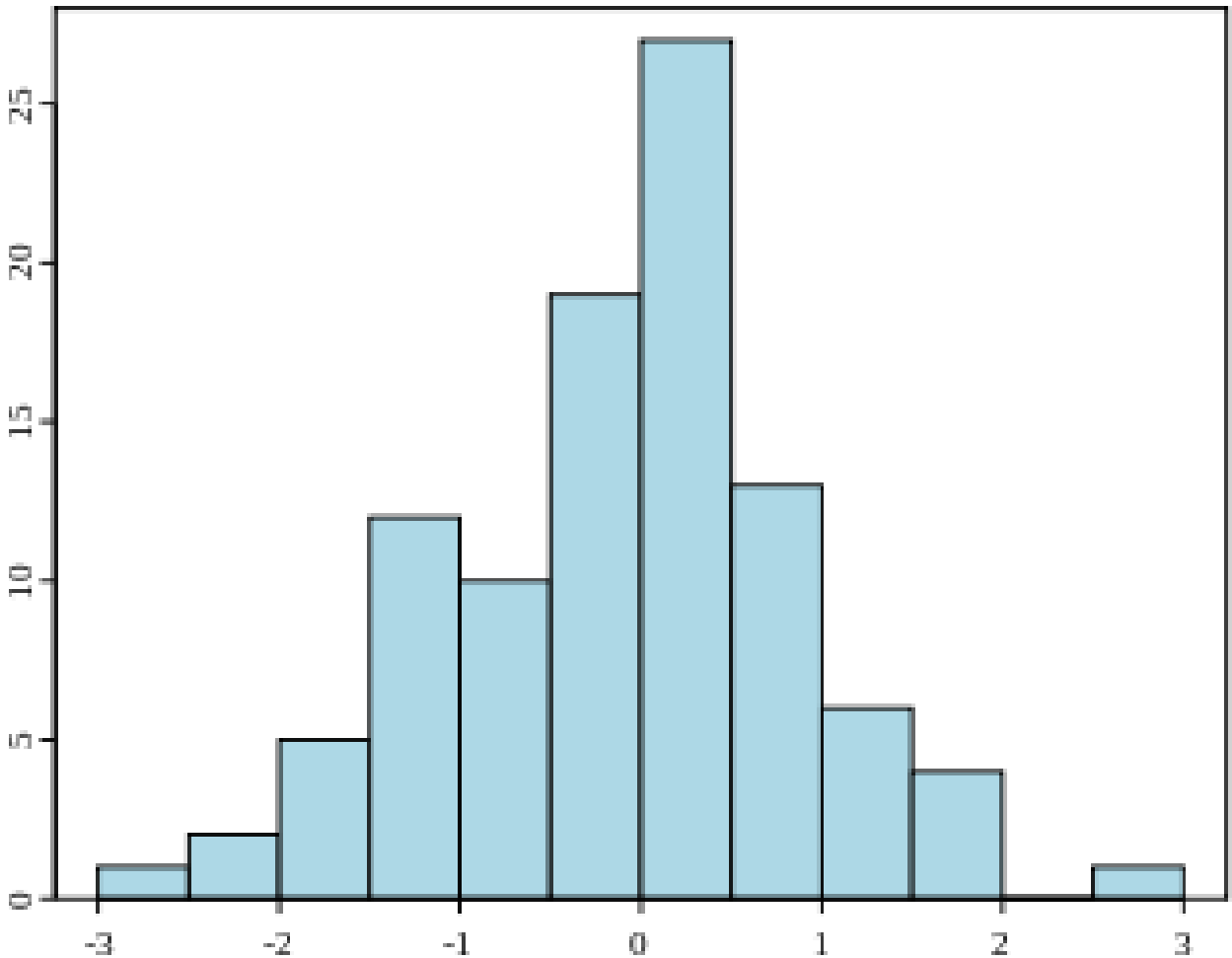
Пример



Outline

Гистограмма

Гистограмма - столбчатая диаграмма, способ графического представления табличных данных



Гистограмма

```
from random import random
import seaborn
from matplotlib.pyplot import show

X = [ sum( [random() for i in range(1000)] )
                          for j in range(1000)]

# подготовим гистограммму и кривую распределения
seaborn.distplot(X)

show()
```

Гистограмма

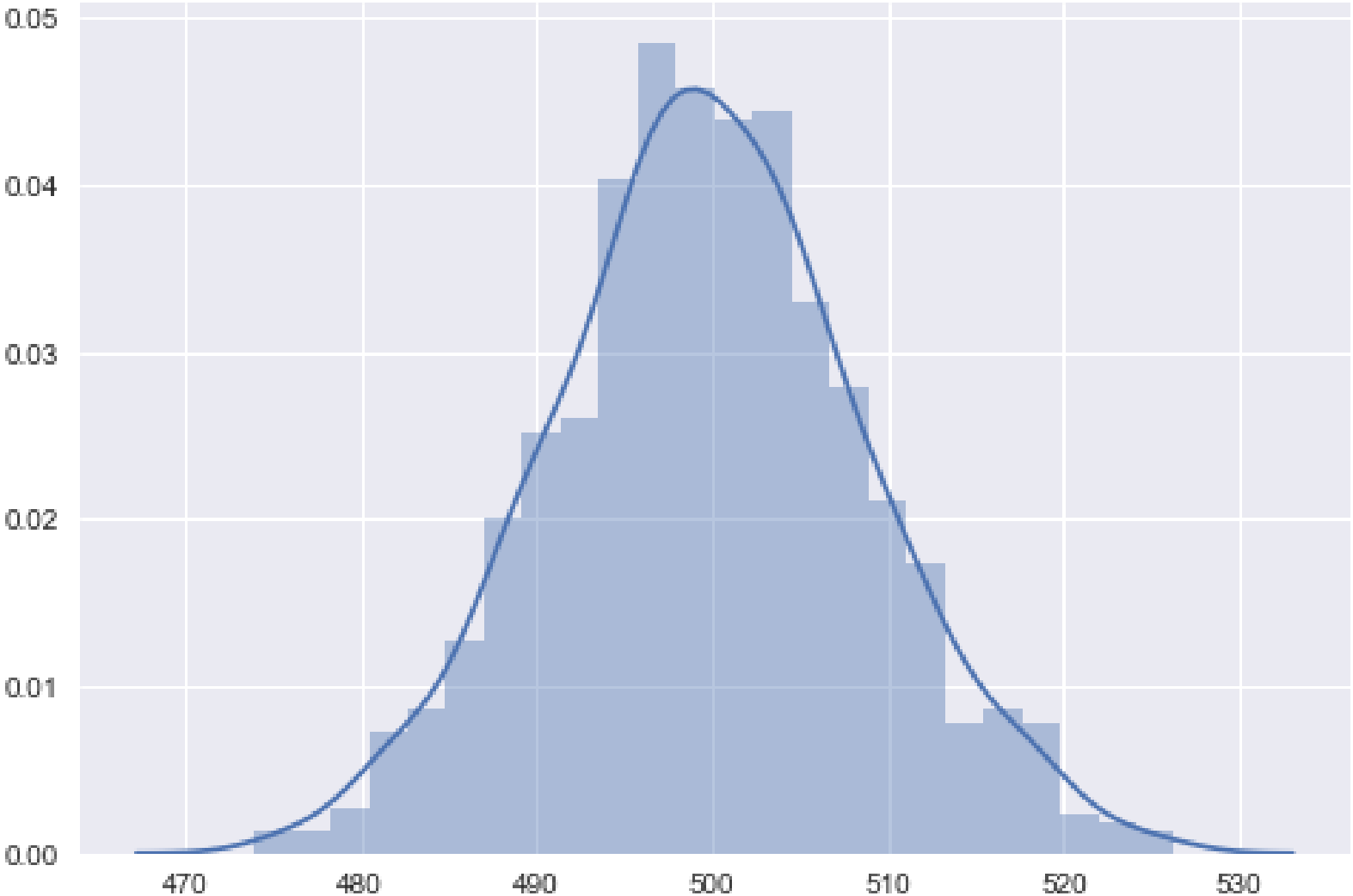
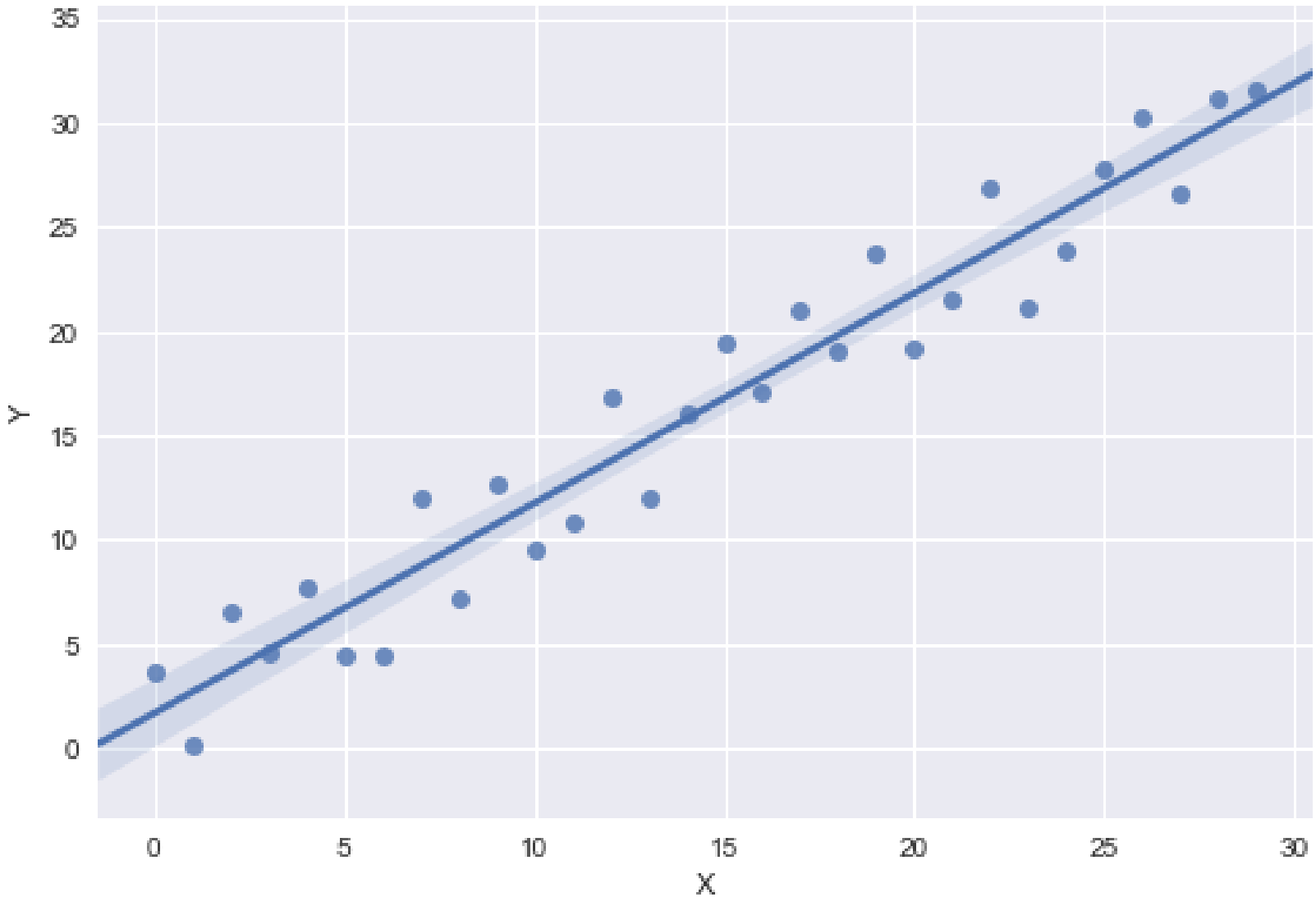


Диаграмма рассеивания и коэффициент корреляции

```
import seaborn  # для визуализации

# Таблица для хранения стат.данных
from pandas import DataFrame

# поместим в таблицу как столбцы с заголовками X и Y
D = DataFrame( {'X':X, 'Y':Y} )

# Построим диаграмму рассеивания
seaborn.regplot(x='X', y='Y', data=D);

plt.show()
```

Диаграмма рассеивания

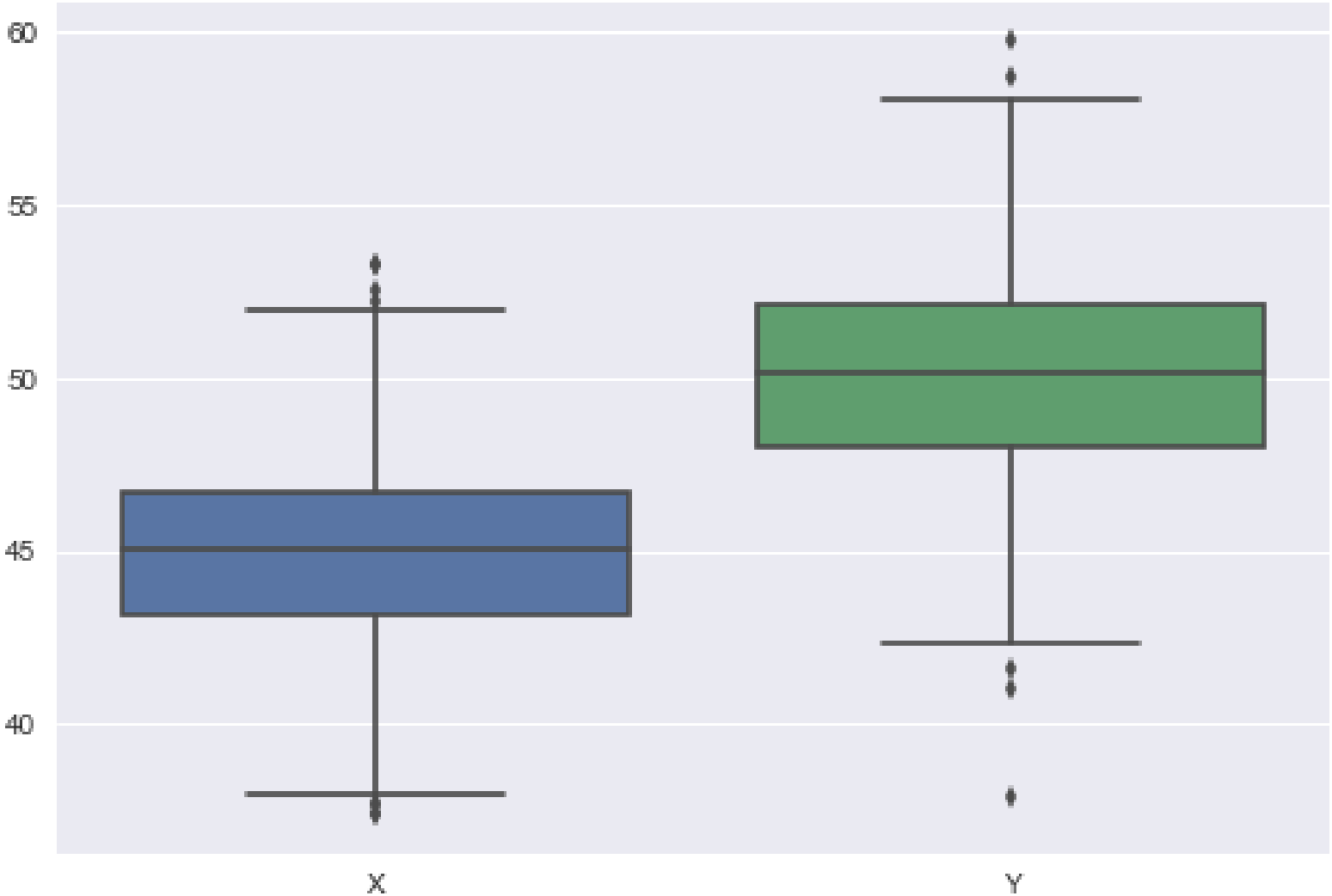


Outline

Диаграмма размаха ("Ящик с усами")

```
X = [ sum([random() for i in range(90)])
                        for j in range(1000) ]
Y = [ sum([random() for i in range(100)])
                        for j in range(1000) ]

D = DataFrame( {'X' : X, 'Y':Y} )
seaborn.boxplot(D)
plt.show()
```

Диаграмма размаха ("Ящик с усами")



О визуализации статистический данных и проверке гипотез с помощью Python:
nahlogin.blogspot.ru/2016/01/pandas.html

Outline

Представление таблиц DataFrame

Содержимое DataFrame можно конвертировать в HTML таблицу

```
D.to_html()
```

Однако такое преобразование непосредственно не пригодно для отображения в ячейке Jupyter.

Используем пакет IPython чтобы отобразить html:

```
from IPython.display import HTML

HTML( D.to_html() )
```

Представление таблиц DataFrame

При создании HTML таблицы из DataFrame стоит использовать CSS классы

```
HTML(D.to_html(classes=["table-bordered",
    "table-striped", "table-hover"]))
```

Будет создана таблица с не яркой границей, чередующими светлыми и тёмными строками, выделением строки под курсором

	x	y1	y2
0	-2.000000	2.000000	2.000000
1	-1.995996	1.847768	1.999842
2	-1.991992	1.855189	1.999367
3	-1.987988	1.695694	1.998576
4	-1.983984	1.731972	1.997469
5	-1.979980	1.681612	1.996046

Стили таблиц DataFrame

настройка внешнего вида таблиц DataFrame
https://pandas.pydata.org/pandas-docs/stable/style.html

Outline

Ссылки и литература

- https://jupyter.org/try
- Numerical methods in engineering with Python 3 / Jaan Kiusalaas.
- Как использовать Jupyter (ipython-notebook) на 100%

Ссылки и литература

Ссылка на слайды

https://github.com/ivtipm/computer-simulation