Data Analysis in Python

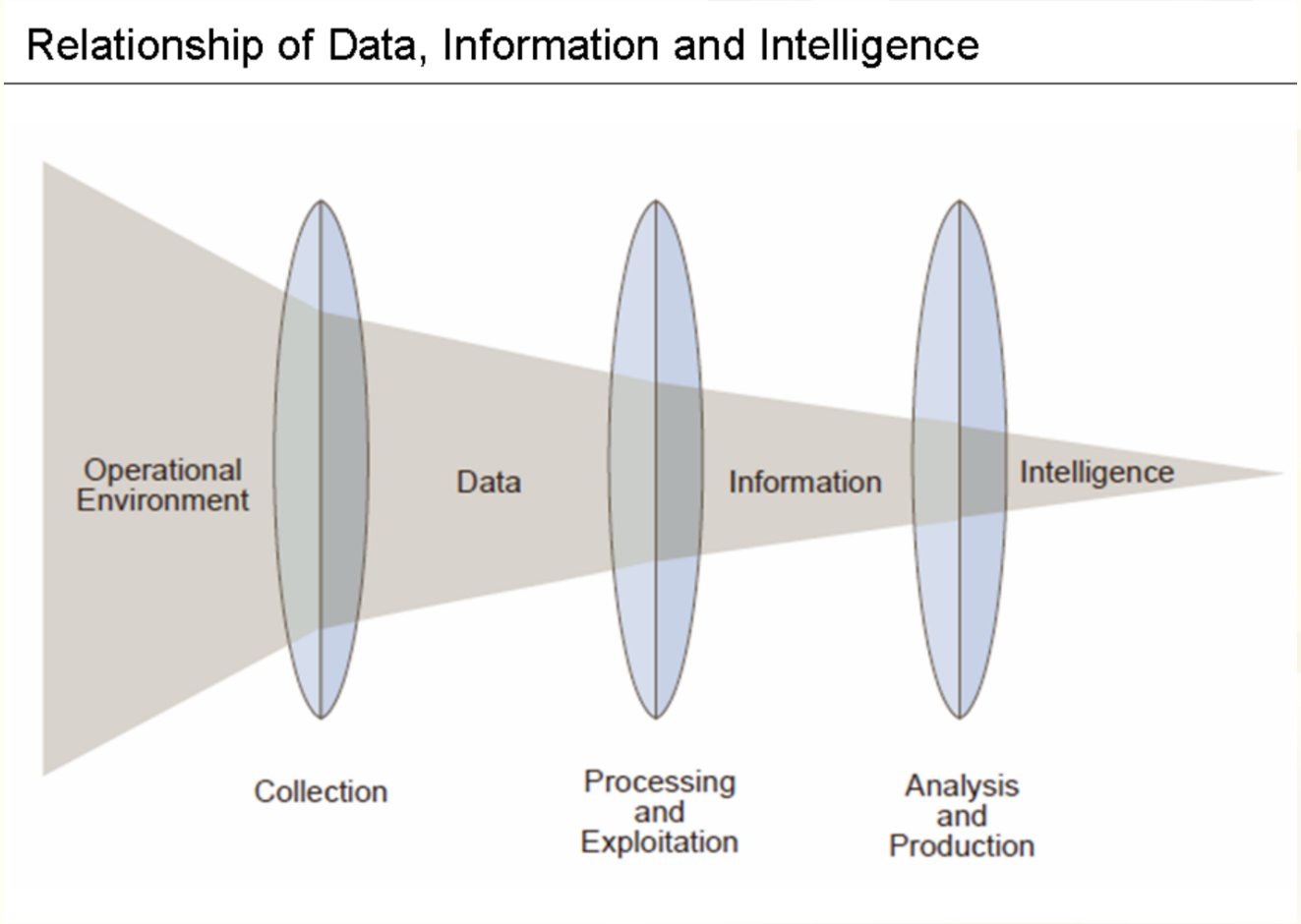
Course for SEA/UAB 2016-2017

José María Lago Alonso

https://github.com/jmlago/DA_python.git

What is Data Analysis?

Analysis of data is a process of inspecting, cleansing, transforming, and modeling data with the goal of discovering useful information, suggesting conclusions, and supporting decision-making.



What is Python?

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.

- Interpreted:
  Compile and execute by any size blocks in any time.
- Object Oriented:
  Uses classes objects and attributes.
- High-Level:
  Can not allocate memory manually.
- Dynamic semantics:
  Python doesn't have static types.

Why use Python?

Strengths

- Glue language
- Simple and easy to learn
- Program modularity and code reuse
- Edit-test-debug really fast
- General purpose language
- Cross platform

Weaknesses

- Slower than compiled languages
- Python2 not compatible with Python3
- Lack of static types
- Can't free memory in usual way

Conda

Anaconda

We are going to use Python3 in all this course !!!

Definition

Anaconda is an easy-to-install free package manager, environment manager, Python distribution, and collection of over 720 open source packages offering free community support.

Why?

Because Anaconda's packages are for data analytics, data science, and scientific computing.
Also Conda makes sure that all packages and environments works fine together.

Installation

- Go to: https://www.continuum.io/downloads
- Choose your OS
- Download the installer
- Follow the steps on the webpage

Warning

For Linux users, you may know if there is another Python installation on your OS and in that case remove the Spyder, for instance you can use:

```
sudo apt-get purge spyder
```

Before install Anaconda.

Spyder

Integrated Development Enviroment

Definition

Scientific PYthon Development EnviRonment

- Similar to RStudio and MATLAB IDE's
- IDE for Science
- Exploratory
- Easy debugging

Verify that all is working OK

- Open SPYDER
- Try to execute the next code:

```
import pandas as pd
import os
import statsmodels
import scipy

print("Everything is working OK!!!")
```

Type the code, select all the code and press CTRL+INTRO

Python Stuff 1

Libraries

In Python we need to set the modules that we are going to use at the beginning of the script. We do in the following way:

```
import somelibrary as somename
somename.somefunctioninthelib()
```

Generic Data Types

Python really has dynamic semantics so a variable is somehow dynamic type, for instance:

```
a = [1,2,3] #--> list of numbers
a = a[1] #--> position in a list
a = "abcd" #--> string
...
```

Python Stuff 2

Functions

We can use functions very easy because of the dynamic semantics. Also in Python the most important thing is **INDENTATION** this is how we determine the loops and the range of the functions.
Beautiful and readable code.

```
def somefunction(param1,param2):
    a = param1*param2
    return a
```

Python Stuff 3

Classes

Here is the OO part.

Easy example:

```
class Complex:
    def __init__(self, realpart, imagpart):
        self.r = realpart
        self.i = imagpart

x = Complex(3.0, -4.5)

x.r, x.i
(3.0, -4.5)
```

Help!

- `StackOverFlow` and `How To "do something" in Python`
- Python documentation
- External libraries documentation

Warning

Be careful of `StackOverFlow` because it's easy to copy and paste the code but if you don't understand what are you doing you'll have several problems.

Let's practice a little bit !!!
Open the `practice0.py`
After this open `oo.py`

Libraries that we need

- Pandas
- BeautifulSoup
- Matplotlib
- Statsmodels
- NumPy
- SciPy
- Scikit-learn
- Plotly
- Sqlite3

And many others...

Numpy 1

What is numpy ?

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Numpy 2

Now we have numeric arrays (not lists) and we can apply operations:

Numpy arrays

```
import numpy as np
a = np.array(1,2,3,4)     # WRONG
a = np.array([1,2,3,4])   # RIGHT
b = np.array([(1.5,2,3), (4,5,6)])
print(b)
        array([[ 1.5,  2. ,  3. ],
               [ 4. ,  5. ,  6. ]])
c = np.array( [ [1,2], [3,4] ], dtype=complex )
print(c)
        array([[ 1.+0.j,  2.+0.j],
               [ 3.+0.j,  4.+0.j]])
```

Numpy 3

As we can see in the previous slide there are many `dtypes` for an array of NumPy. More precisely a `dtype` follow the next properties:

Properties

- Type of the data (integer, float, Python object, etc.)
- Size of the data (how many bytes is in e.g. the integer)
- Byte order of the data (little-endian or big-endian)
- If the data type is structured (an aggregate of other data types).

Dtypes sample

int16,bool,float64,complex,str,unicode,buffer,object,...

Numpy 4

Some useful algebraic functions, better to see as example:

Numpy useful functions

```
a = np.array([[1.0, 2.0], [3.0, 4.0]])
a.transpose() #transposed matrix
np.linalg.inv(a) #inverse matrix
np.dot (a, a) # matrix product
np.trace(a) #trace of the matrix
y = np.array([[5.], [7.]]) #independent term
np.linalg.solve(a, y) #solver of a system
```

Numpy 5

Polynomial fit functions that minimize the SSE:

Polynomial fit functions

```
x = np.array([0.0, 1.0, 2.0, 3.0,  4.0,  5.0])
y = np.array([0.0, 0.8, 0.9, 0.1, -0.8, -1.0])
z = np.polyfit(x, y, 3)#3 is the degree
#of the fitting polynomial
print(z)
        array([ 0.08703704, -0.81349206,
    1.69312169, -0.03968254])
p = np.poly1d(z)
p(0.5)
        0.6143849206349179
p30 = np.poly1d(np.polyfit(x, y, 30))
        RankWarning: Polyfit may be poorly conditioned
```

Matplotlib 1

What is matplotlib ?

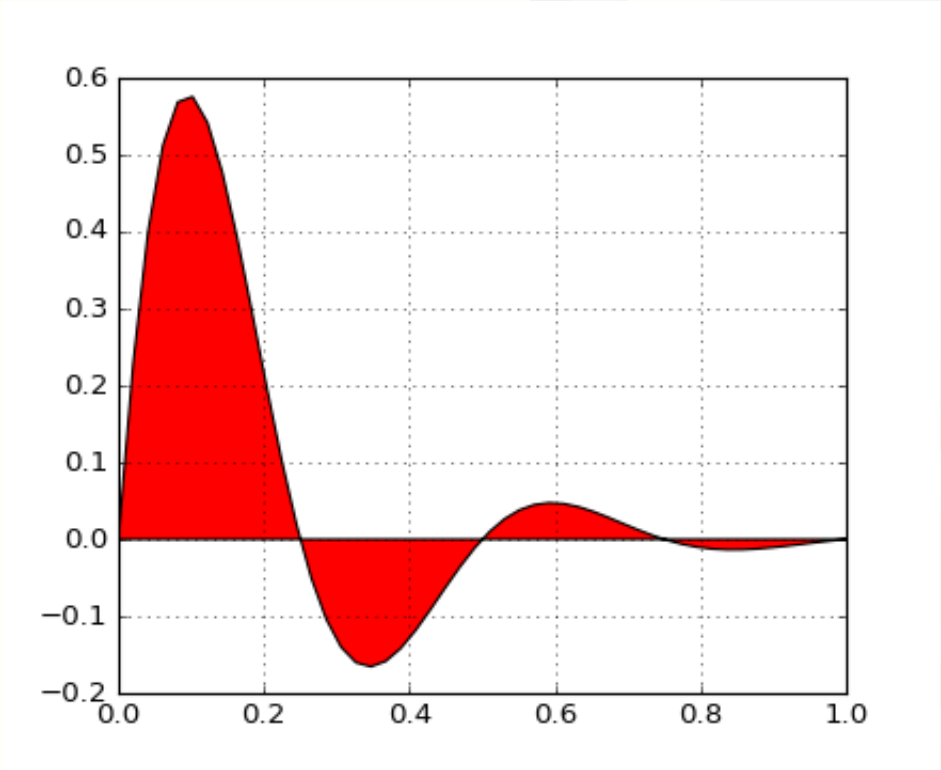
Matplotlib is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

Basic example

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0, 1)
y = np.sin(4 * np.pi * x) * np.exp(-5 * x)
plt.fill(x, y, 'r') #fills the curve with red color
plt.grid(True) #draws the grid
plt.show() #shows the plot
```

Matplotlib 2

Show of the previous plot:



Matplotlib 3

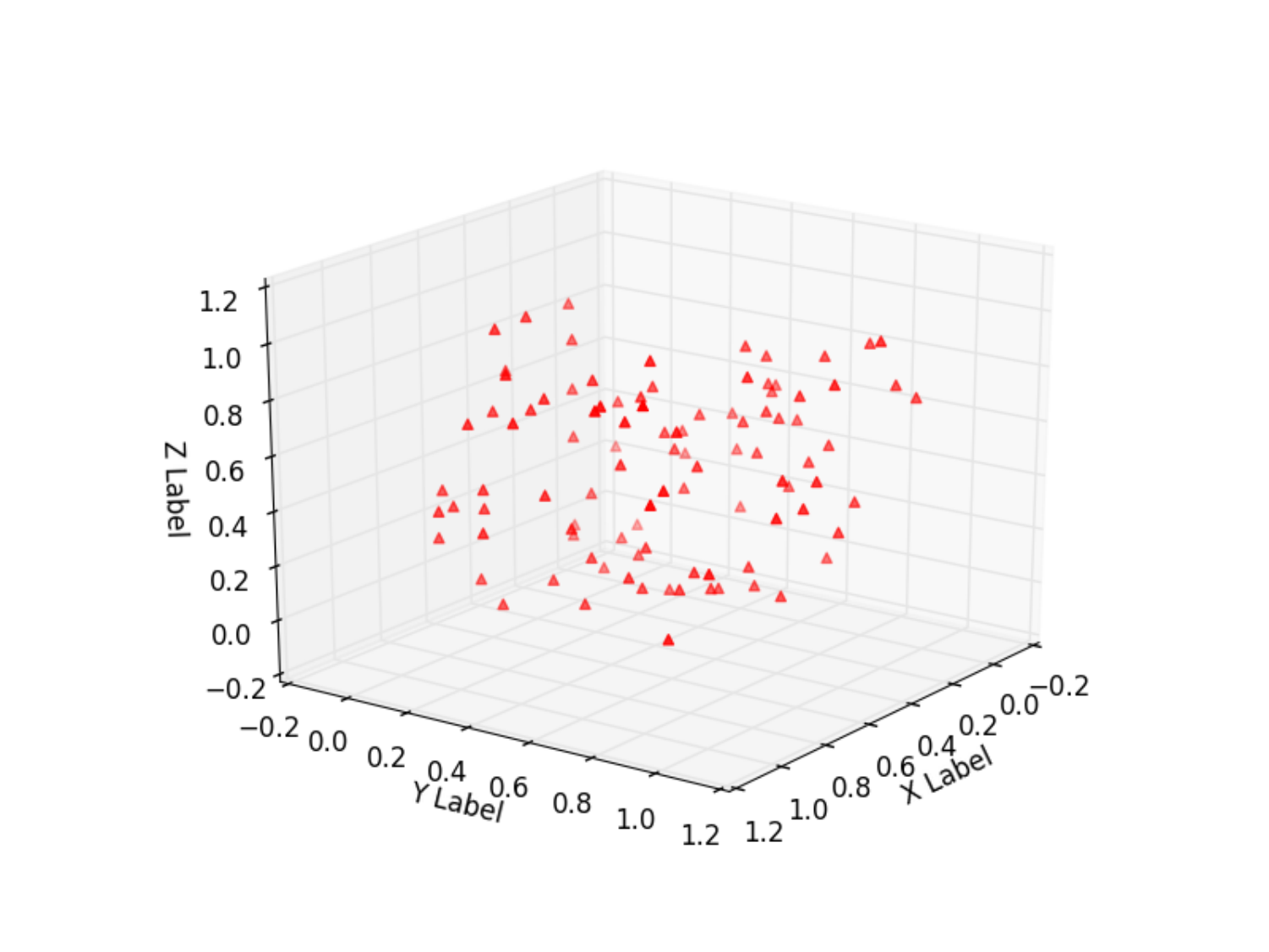
Now we are going to draw some data in 3D just as an example:

3D example

```
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
n = 100
xs = np.random.uniform(0,1,n)
ys = np.random.uniform(0,1,n)
zs = np.random.uniform(0,1,n)
ax.scatter(xs, ys, zs,color="r",marker="^")
ax.set_xlabel('X Label')
ax.set_ylabel('Y Label')
ax.set_zlabel('Z Label')
plt.show()
```

Matplotlib 4

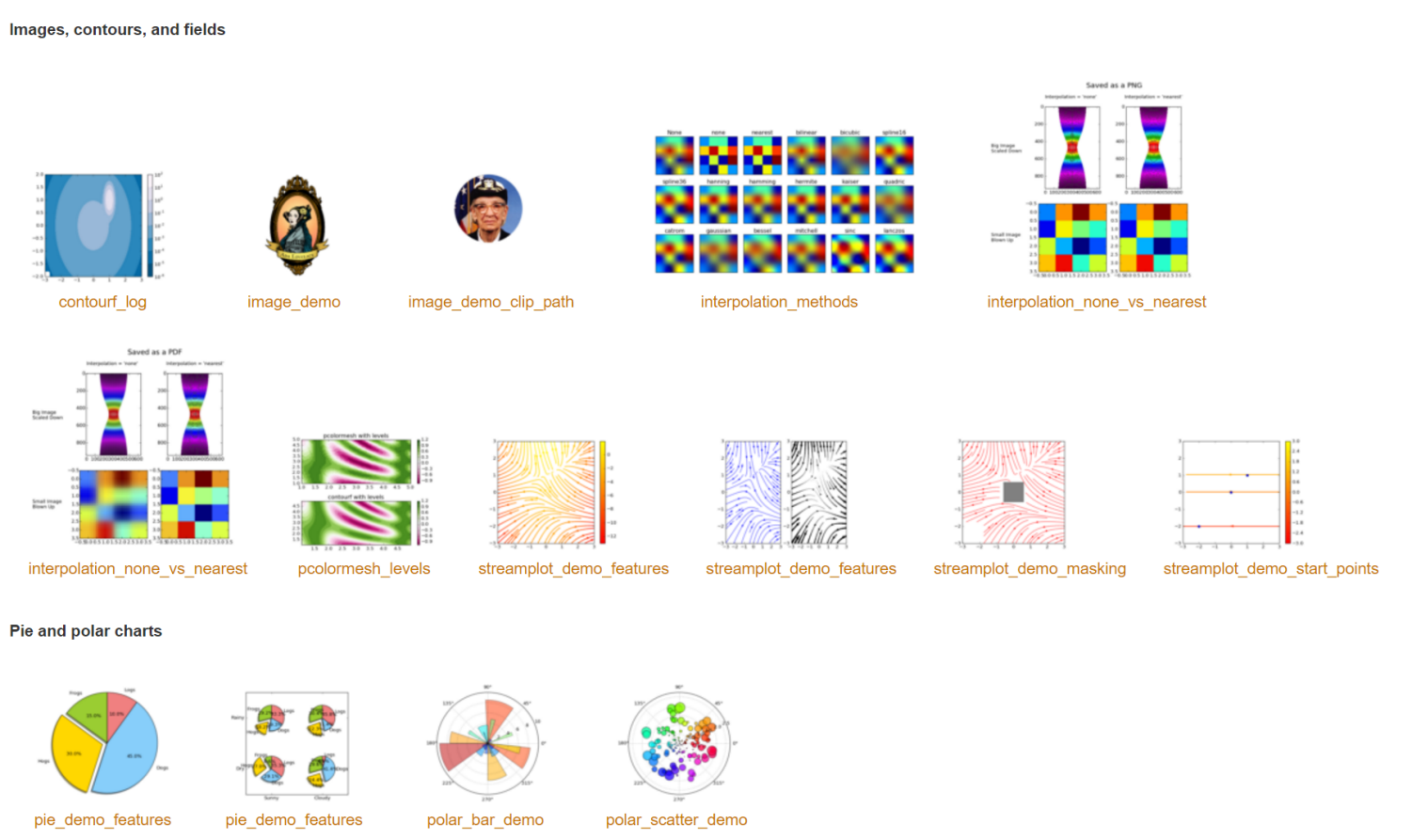
Show of the previous plot:



Matplotlib 5

The most important part of `matplotlib` is...

The gallery: `http://matplotlib.org/gallery.html`



Practice again !!!

Open `practice1.py`

Pandas 1

This library is the CORE of Data Analysis in Python.
And It's most powerful tool...

DataFrame class

This is our table and our database structure in Python.
Has many many attributes and most libraries were build using this structure as a link.

Example

```
import pandas as pd
d = [['Pepito',22],['Juanito',43],['Pablito',20]]
c = ["Name","Age"]
df = pd.DataFrame(data=d,columns = c)
```

Pandas 2

Knowing a little bit more about `DataFrame` class:

Continuing Example

```
df.Age #--> Sometimes correct
df["Age"] #--> Correct
df["Age"].mean()
df["Age"].var()
```

Let's take a look at the documentation:
http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html

Pandas 3

Getting information in different formats:

Some format to DataFrame

```
##All this sentences return a DataFrame object
pd.read_csv(file)
pd.read_excel(file)
pd.read_html(path) # directly from the web
pd.read_json(path) # directly from the web
pd.read_sas(file)
pd.read_sql(file)
pd.read_sql_table(file)
pd.read_stata(file)
df.from_csv(file)
df.from_dict(dict)
...
```

To see how it works, just open **easy_plot.py**

Pandas 4

Querys inside `DataFrame` class:

Query in Python

```
x = df[(df["label1"] == "some1") &
(df["label2"] == "some2")]["label3"]
```

This could be a set of numbers coming from the DataFrame as a request. The request is like we want all elements on the DataFrame that have in the column named `label1` the value `some1` **and** that have in the column named `label2` the value `some2`. And from this set of rows that follow the answer we want the values of column named `label3`. Then put all this values in a variable named `x`.

DataFrame to LaTeX

```
df2.to_latex().replace("\n","")
# generates a table in LaTeX format
```

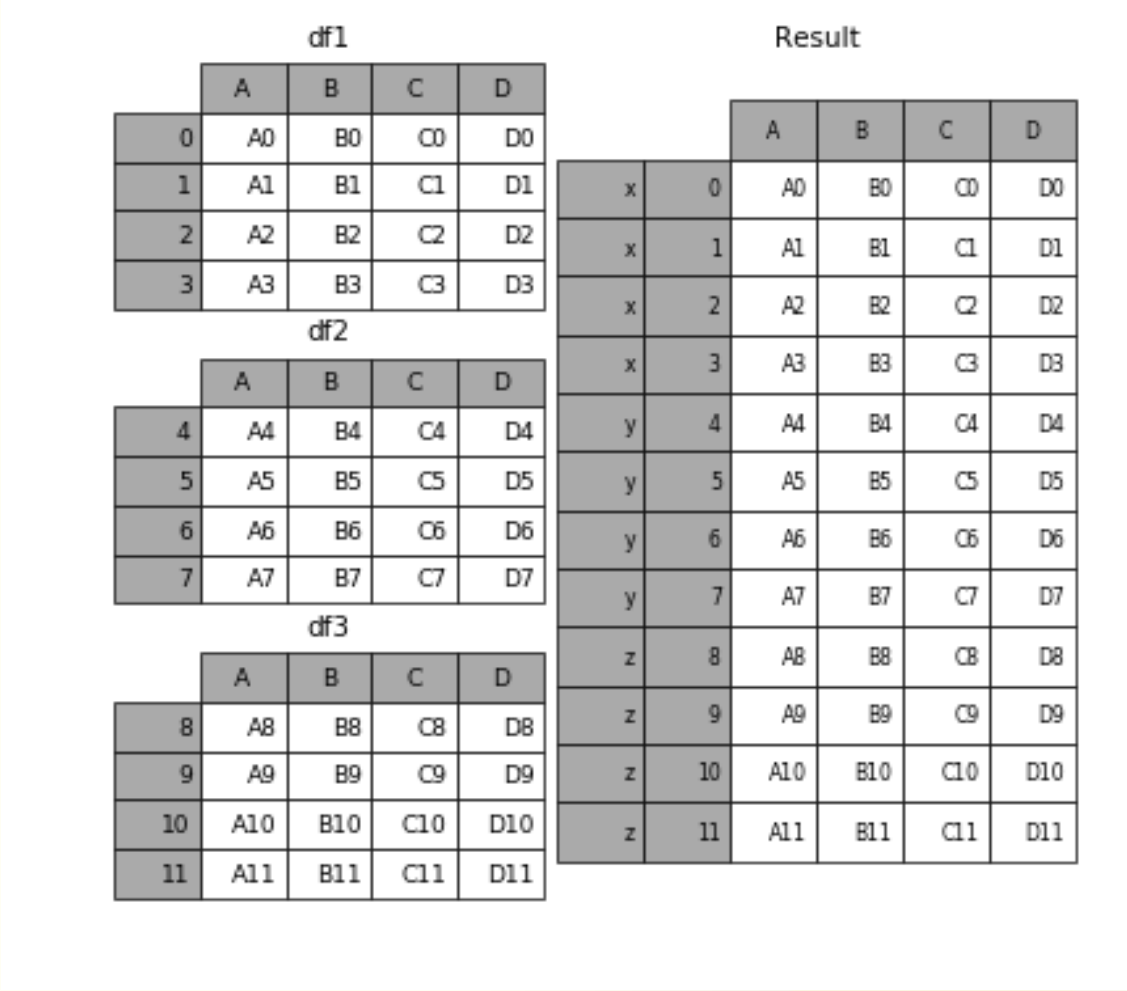
Pandas 5

Few more posibilities

```
df.dropna(axis=1, how='any',inplace=True) #drop NA values
agg_col = df.groupby('label1').aggregate(sum)
agg_col.index = ['bar1', 'bar2', 'bar3']
agg_col.plot(kind='bar') #see other kinds in documentation
```

Pandas 6

```
df = pd.concat(df1,df2,df3)
```



df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df2

	A	B	C	D
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7

df3

	A	B	C	D
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

Result

		A	B	C	D
x	0	A0	B0	C0	D0
x	1	A1	B1	C1	D1
x	2	A2	B2	C2	D2
x	3	A3	B3	C3	D3
y	4	A4	B4	C4	D4
y	5	A5	B5	C5	D5
y	6	A6	B6	C6	D6
y	7	A7	B7	C7	D7
z	8	A8	B8	C8	D8
z	9	A9	B9	C9	D9
z	10	A10	B10	C10	D10
z	11	A11	B11	C11	D11

Pandas 7

```
result = pd.concat([df1, df4], axis=1)
```

df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df4

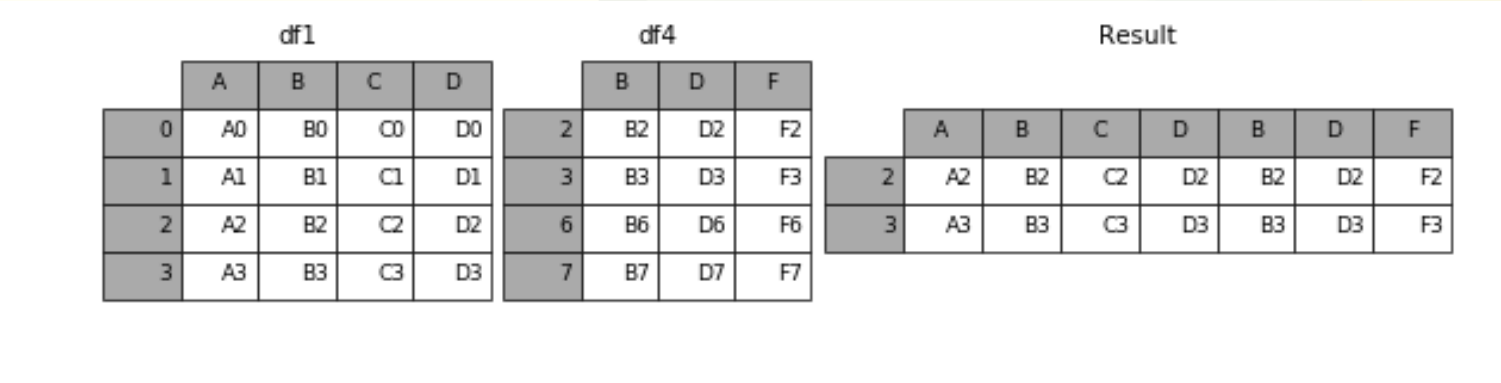
	B	D	F
2	B2	D2	F2
3	B3	D3	F3
6	B6	D6	F6
7	B7	D7	F7

Result

	A	B	C	D	B	D	F
0	A0	B0	C0	D0	NaN	NaN	NaN
1	A1	B1	C1	D1	NaN	NaN	NaN
2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3
6	NaN	NaN	NaN	NaN	B6	D6	F6
7	NaN	NaN	NaN	NaN	B7	D7	F7

Pandas 8

```
result = pd.concat([df1, df4], axis=1, join='inner')
```



df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df4

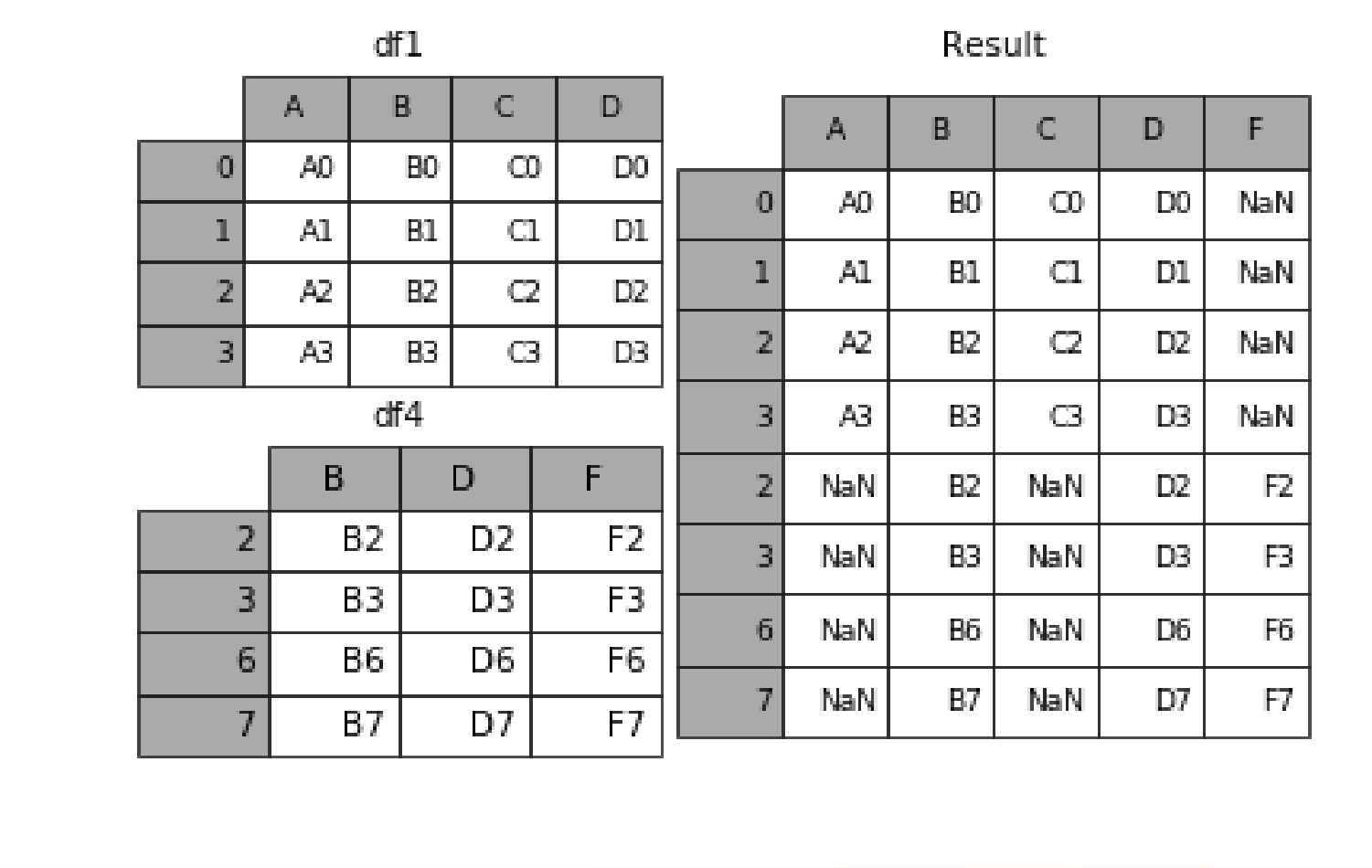
	B	D	F
2	B2	D2	F2
3	B3	D3	F3
6	B6	D6	F6
7	B7	D7	F7

Result

	A	B	C	D	B	D	F
2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3

Pandas 9

```
result = df1.append(df4)
```



df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df4

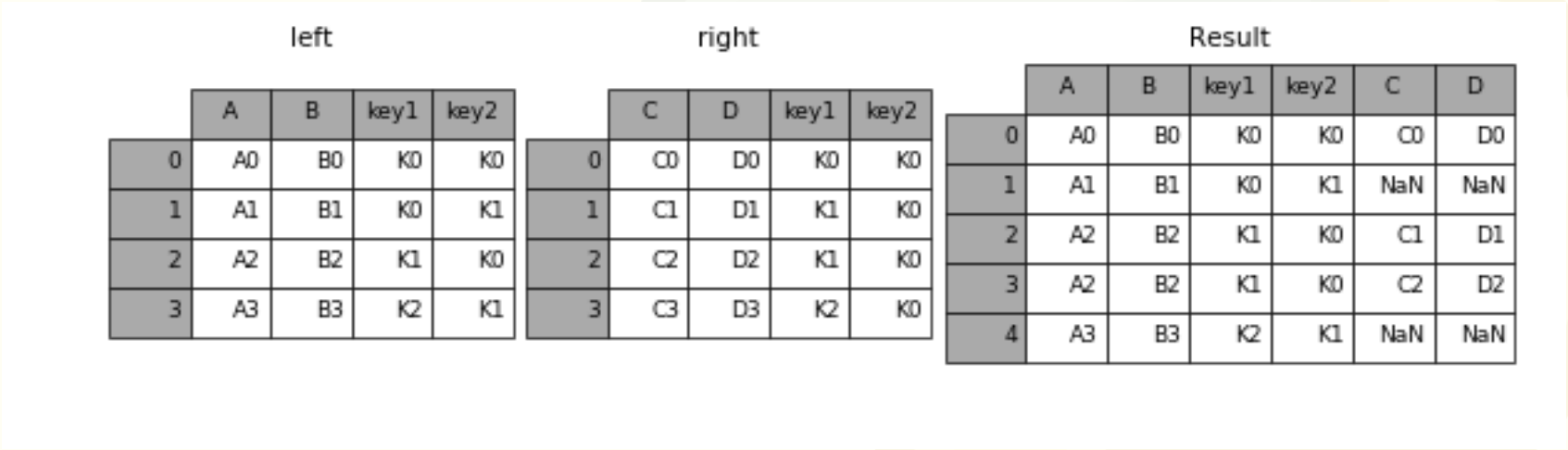
	B	D	F
2	B2	D2	F2
3	B3	D3	F3
6	B6	D6	F6
7	B7	D7	F7

Result

	A	B	C	D	F
0	A0	B0	C0	D0	NaN
1	A1	B1	C1	D1	NaN
2	A2	B2	C2	D2	NaN
3	A3	B3	C3	D3	NaN
2	NaN	B2	NaN	D2	F2
3	NaN	B3	NaN	D3	F3
6	NaN	B6	NaN	D6	F6
7	NaN	B7	NaN	D7	F7

Pandas 10

```
result = pd.merge(left, right,
how='left', on=['key1', 'key2'])
```



left

	A	B	key1	key2
0	A0	B0	K0	K0
1	A1	B1	K0	K1
2	A2	B2	K1	K0
3	A3	B3	K2	K1

right

	C	D	key1	key2
0	C0	D0	K0	K0
1	C1	D1	K1	K0
2	C2	D2	K1	K0
3	C3	D3	K2	K0

Result

	A	B	key1	key2	C	D
0	A0	B0	K0	K0	C0	D0
1	A1	B1	K0	K1	NaN	NaN
2	A2	B2	K1	K0	C1	D1
3	A2	B2	K1	K0	C2	D2
4	A3	B3	K2	K1	NaN	NaN

Pandas 11

```
result = pd.merge(left, right,
how='outer', on=['key1', 'key2'])
```

left

	A	B	key1	key2
0	A0	B0	K0	K0
1	A1	B1	K0	K1
2	A2	B2	K1	K0
3	A3	B3	K2	K1

right

	C	D	key1	key2
0	C0	D0	K0	K0
1	C1	D1	K1	K0
2	C2	D2	K1	K0
3	C3	D3	K2	K0

Result

	A	B	key1	key2	C	D
0	A0	B0	K0	K0	C0	D0
1	A2	B2	K1	K0	C1	D1
2	A2	B2	K1	K0	C2	D2

Pandas 12

```
result = pd.merge(left, right,
how='inner', on=['key1', 'key2'])
```

left

	A	B	key1	key2
0	A0	B0	K0	K0
1	A1	B1	K0	K1
2	A2	B2	K1	K0
3	A3	B3	K2	K1

right

	C	D	key1	key2
0	C0	D0	K0	K0
1	C1	D1	K1	K0
2	C2	D2	K1	K0
3	C3	D3	K2	K0

Result

	A	B	key1	key2	C	D
0	A0	B0	K0	K0	C0	D0
1	A1	B1	K0	K1	NaN	NaN
2	A2	B2	K1	K0	C1	D1
3	A2	B2	K1	K0	C2	D2
4	A3	B3	K2	K1	NaN	NaN
5	NaN	NaN	K2	K0	C3	D3

... and many other ways to fusion DataFrames.

Practice again!!!

Open the `practice2.py`

RegEx 1

Definition

A regular expression is, in theoretical computer science and formal language theory, a sequence of characters that define a search pattern. Usually this pattern is then used by string searching algorithms for "find" or "find and replace" operations on strings.

Metachars table 1

`.` matches any single character, for example: `a.b` matches `acb` and many others.
`[ ]` matches a single character that is contained within the brackets, for example: `[abc]` matches a,b or c.
`[^ ]` matches a single character that is not contained within the brackets.
`^` matches the starting position within the string.
`$` matches the ending position of the string or the position just before a string-ending newline.
`( )` defines a marked subexpression.

RegEx 2

Metachars table 2

\n matches what the nth marked subexpression matched, where n is a digit from 1 to 9.

* matches the preceding element zero or more times.

{min,max} matches the preceding element at least min and not more than max times.

? matches the preceding element zero or one time.

+ matches the preceding element one or more times.

| matches either the expression before **or** the expression after the operator.

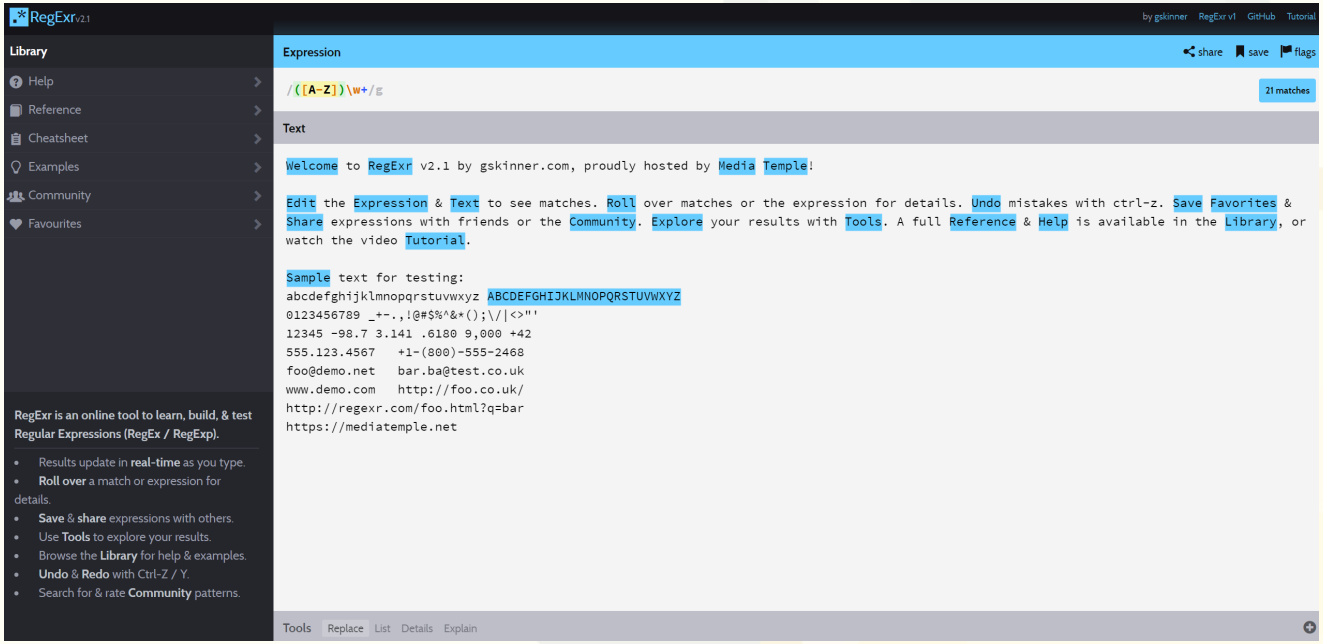
RegEx 3

Library re

```
import re
string = "I want to know all the 3 numbers\
on the string 1,-2.4"
pattern = "(-?([\d])+)(\.?[\d]+)"
print(re.search(string,pattern))
        3,1,-2.4
```

RegEx 4

To practice on the RegEx topic, the best tool is: `http://regexr.com/`



Practice again!!!

Open the `practice3.py`

Real examples

Now we are in the correct path to test our knowledge in real examples.
We are going to follow a real example from this point of the course, and see dome topics of DA:

- Web Scrapping
- Machine Learning

For this we need to learn more strategies and test more Python libraries but now we can understand the examples.
For the first part, Web Scrapping, we will get the information from:
`http://www.coches.net/`

This is not compulsory, so if you want to gather data from other web-pages you can, so feel free to tune your examples, but first...

Web-Scraping

Idea

Extract data from the web with a useful format.

Phylosophy

This is the artistic part, so in many cases, more you are digging in the `HTML` better results you will have in your Python script.

Practise

Open your browser (I recommend Chrome or Firefox) and your text editor:

- Windows: Download `SublimeText`
- Mac: You can download `SublimeText` or maybe you have some good editor by default.
- Linux: Use some text editor like `gedit`

Legality

Before digging into the scraping we need to know what is:

Terms of Service

Terms of service (also known as ToS or TOS and TOU) are rules by which one must agree to abide in order to use a service. Terms of service can also be merely a disclaimer, especially regarding the use of websites.

Before scrap something you **need to know** the terms of service of the web-page.

Google Example

`HTTP 403 Forbidden status` while trying to scrap very fast.
Google has **nice attitude** so they say that you are violating TOS maybe because someone could be hacking you so be aware (but you are banned from Google until the unusual traffic stops).

URL analysis

Take a look at the URL of the page:

```
url = "http://www.coches.net/segunda-mano/?MakeId=28\
&ModelId=0&PrvId=0&Version=\
&BodyTypeId=0&FuelTypeId=0&MaxKms=9999999\
&MinKms=0&MaxYear=9999&MinYear=0&fi=SortDate&or=-1\
&MaxPrice=0&SearchOrigin=2&text=mercedes__benz"
```

But... this is the **BEST** URL that we can get? (**NO** and this is why we need time for search in the web)

BeautifulSoup

Definition

Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.

Usage

To see how it works we need and example because we need to analyze the `xml` format and the best tool for that and somehow for scraping is Chrome.

It's better to see how it works with an example

Open practice4.py

Example 1

On this example we are going to subtract information from a website and see some descriptive statistics.

Libraries for the example

First of all, we will need:

Code

```
import pandas as pd #--> Data structure
from bs4 import BeautifulSoup #-->Parse the webpage
from urllib.request import urlopen #-->Web connect protocol
```

HTML/XML Code

Look deep into the web and try to understand structure.

Warning

The kind of things that could stop us from getting information are from this type:
The content of the page is charged dynamically and there is no information on the `URL` and the `HTML` content is charged when happens some event on the web. We can deal with this but is not on the scope of the course.

Figure: Sublime Text Editor (Windows/Mac/Linux).

Figure: Gedit Editor(Linux).

Using BeautifulSoup

Code

```
page = urlopen(url)
soup = BeautifulSoup(page,"lxml")

##After looking at the html code
title = soup.find('div',attrs=
{"class":"mt-SerpList"}).findAll('span')
price = soup.find('div',attrs=
{"class":"mt-SerpList"}).findAll('strong')
attribute = soup.find('div',attrs=
{"class":"mt-SerpList"}).findAll('li')
```

Set lists and build DataFrame

Code

```
titles = []
prices = []
locality = []
...
for i in range(len(title)):
    if title[i].string[:6] != "Añadir":
        titles.append(title[i].string)
...
```

We need to **take care** about what information and how this information is supplied.
Somehow we need to personalize the loops because of the `XML/HTML` structure.

Set DataFrame

Code

```
d = []
for i in range(len(titles)):
    d.append([titles[i],prices[i],locality[i]
    ,gas[i],year[i],numkm[i]])

a = pd.DataFrame(data=d,columns=["Title","Price",
"Locality","GasType","Year","Km"])
```

Now we have to options to continue:

- Build a library of functions for reuse the code.
- Start making plots and statistics.

Reusability of the Code

Tips

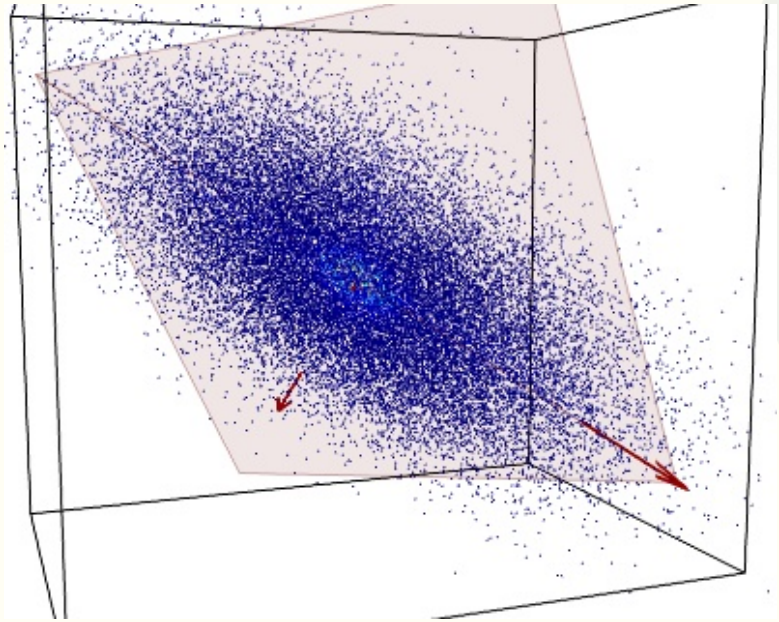
- Keep the code DRY. Dry means "Don't Repeat Yourself".
- Make a class/method do just one thing.
- Write unit tests for your classes AND make it easy to test classes.
- Remove the business logic or main code away from any framework code
- Try to think more abstractly and use Interfaces and Abstract classes.
- Code for extension. Write code that can easily be extended in the future.
- Don't write code that isn't needed.
- Try to reduce coupling.
- Be more Modular

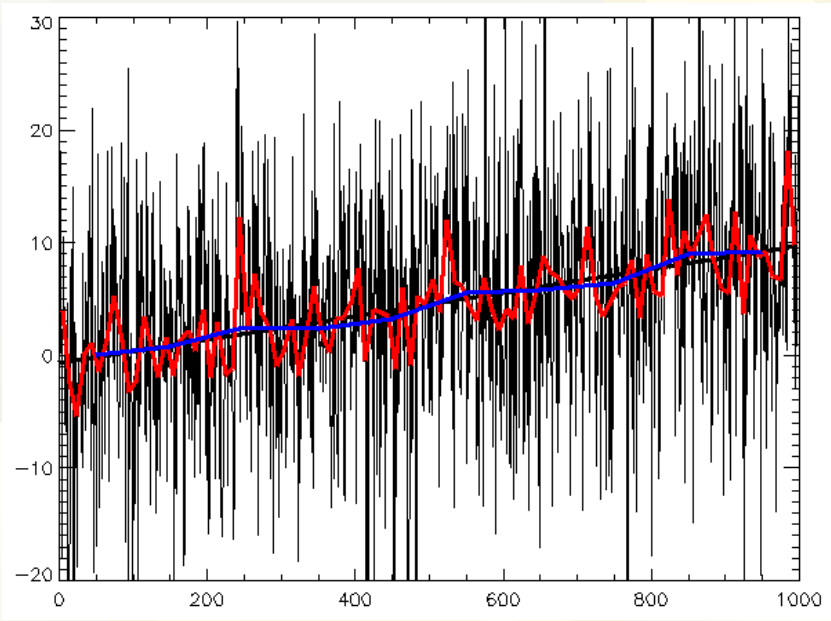
Open `cocheslib.py` **!!!**

Statistics for DA

An Useful tool

The statistics is one of the most useful tools in DA, in this course we are going to see just two interesting tools: **Principal Component Analysis** and some about **Temporal Series**





Principal Component Analysis

What is PCA?

PCA (Principal Component Analysis) is the change the n original variables to $n' = 1, 2, 3, ...$ a short number of new variables (principal components), **loosing the minimal amount of information** by projecting the data in the best subspace

We are going to see a particular case of PCA that is **Spectral Decomposition**

Linear Algebra

To continue we need to refresh some concepts of linear algebra:

- The eigenvector v_i has eigenvalue λ_i if : $Av_i = \lambda_i$
- Eigenvectors are orthogonal: $< v_i, v_j >= 0$ if $i \neq j$
- V is an orthogonal matrix if $V^t V = VV^t = I$

SD

Spectral Theorem

Let A be a matrix $n \times n$ symmetric, then A decomposes this way:

$$A = DED^t = \sum_{i=1}^{n} \lambda_i d_i d_i^t$$

Where E is a diagonal matrix of the **eigenvalues** of A sorted in decreasing order $\lambda_1 > \lambda_2 > \lambda_3 > ... > \lambda_n$ and D is an **orthogonal** matrix whose columns are the **eigenvectors** of A.

Q: How we can use this with the data?
A: Statistics

Dimension Reduction

It's clear that we can split A in the following way:

$$A = D\sqrt{E}\sqrt{E}D^t = QQ^t = \sum_{i=0}^{n} q_i q_i^t$$

So if the $range(A) = m < n$ then we can reduce the dimension of the problem to m insted of n.

Remark

$\sum_{i=0}^{n} q_i q_i^t = \sum_{i=1}^{n} \lambda_i d_i d_i^t$

So as the eigenvalues are sorted, if $\lambda_1 > \lambda_2 > ... > \lambda_i \approx 0 > ... > \lambda_n$ we can also reduce the dimension of the problem but loosing a little bit of information in the process.

How to use all this theory

We have some data with this shape:

obs	X_1	X_2	...	X_n	Y
1	x_{11}	x_{21}	...	x_{n1}	y_1
...	...	...	...	...	...
p	x_{1p}	x_{2p}	...	x_{np}	y_p
new	x_1*	x_2*	...	x_n*	$y*$

We can use all the theory we have learned before using the **covariance matrix** or the **correlation matrix** that by definition are square and symmetric matrices.
The empirical covariance is:

$$Cov(X_i, X_j) = \frac{1}{n} \sum_{k=1}^{n} (x_{ik} - \overline{X_i}) \cdot (x_{jk} - \overline{X_j})$$

Doing this in Python

```
import numpy as np
import scipy as sp
x=np.random.normal(size=(25,25)) #setting the values
covx = np.cov(x) #computing the covariance matrix
v1,w1 = np.linalg.eig(covx) # getting eigenvalues

nonrep = [] #the little eigenvalues to reduce the dimension
for i in range(len(v1)):
    if v1[i] < 0.5:
        nonrep.append(i)

C = x
C = sp.delete(C,nonrep,1) #dimension reduced
```

Try this with some real data!!!

Temporal Series Analysis

What is TSA?

Time series analysis comprises methods for analyzing time series data in order to extract meaningful statistics and other characteristics of the data. Time series forecasting is the use of a model to predict future values based on previously observed values.

Q: Why we need time series instead of linear models?
A: It is **time dependent**. So the basic assumption of a linear regression model that the observations are independent doesn't hold in this case.

Usage

It's better to see the usage with an example, but basically what we will do is to find a forecaster for the temporal series.
Open `TSA.py` !!!

Machine Learning

What is ML

Machine learning is the subfield of computer science that gives computers the ability to learn without being explicitly programmed.
The main task that ML perform is interpolation.

ML is another useful tool for DA, and we can classify all this methods into three branches:

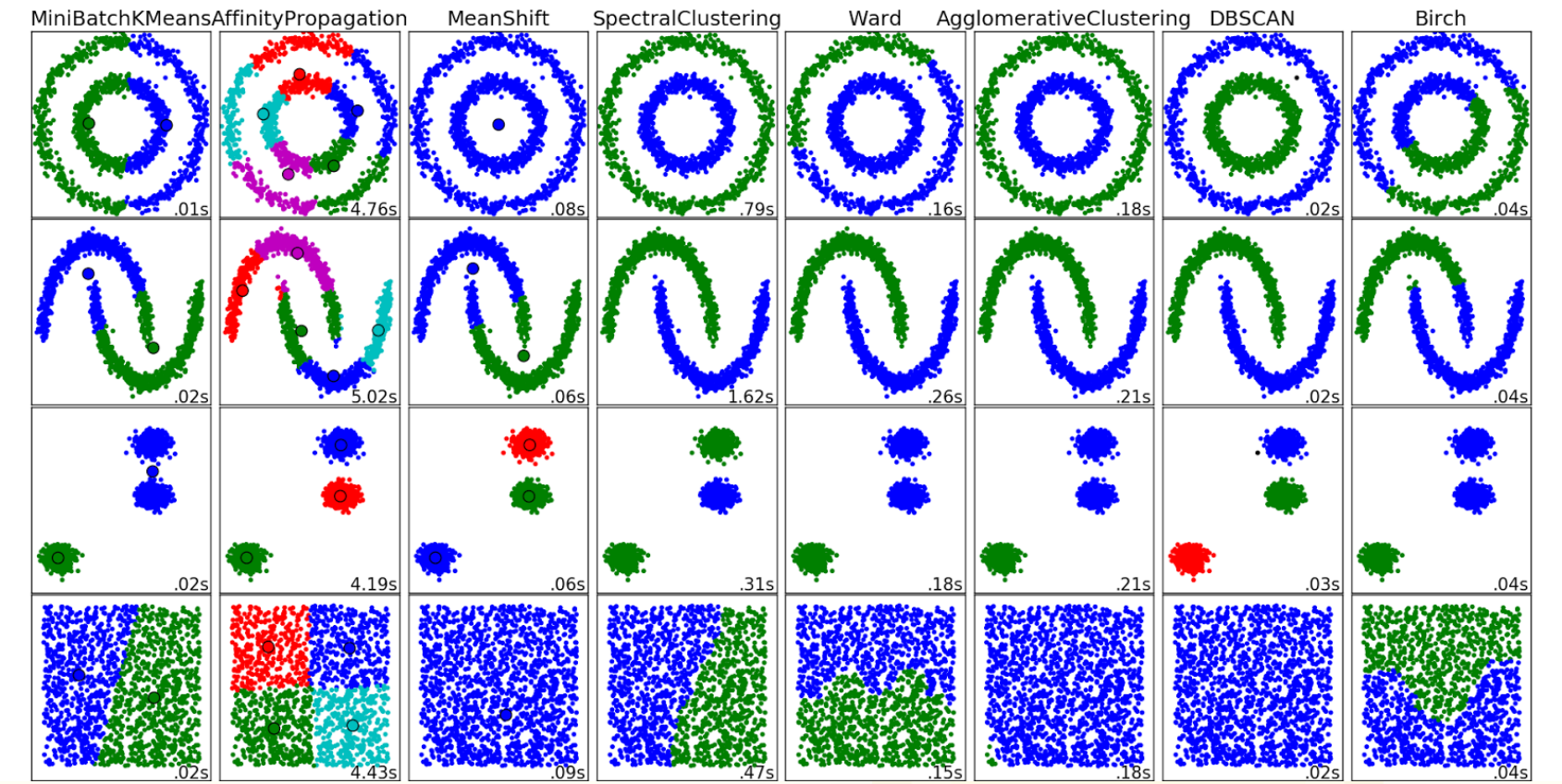
- **Classifying**: inputs are divided into classes, and the learner must produce a model that assigns unseen inputs to one or more (multi-label classification) of these classes.
- **Clustering**: a set of inputs is to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.
- **Regressions**: predicting a continuous-valued attribute associated with an object.

Types of learning

There are many ways to train the models, but the most famous are:

- **Unsupervised learning**: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).
- **Supervised learning**: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.
- **Reinforcement learning**: A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent). The program is provided feedback in terms of rewards and punishments as it navigates its problem space.

Clustering Algorithms



DBSCAN algorithm 1

Density-based spatial clustering of applications with noise (DBSCAN) **is a data clustering algorithm** proposed by Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu in 1996. It is a **density-based** clustering algorithm: given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions (whose nearest neighbors are too far away). DBSCAN is one of the most common clustering algorithms and also most cited in scientific literature.
The **DBSCAN algorithm views clusters as areas of high density separated by areas of low density**. Due to this rather generic view, **clusters found by DBSCAN can be any shape, as opposed to k-means which assumes that clusters are convex shaped**. The central component to the DBSCAN is the concept of **core samples**, which **are samples that are in areas of high density**. A cluster is therefore a set of core samples, each close to each other (measured by some distance measure) and a set of non-core samples that are close to a core sample (but are not themselves core samples).

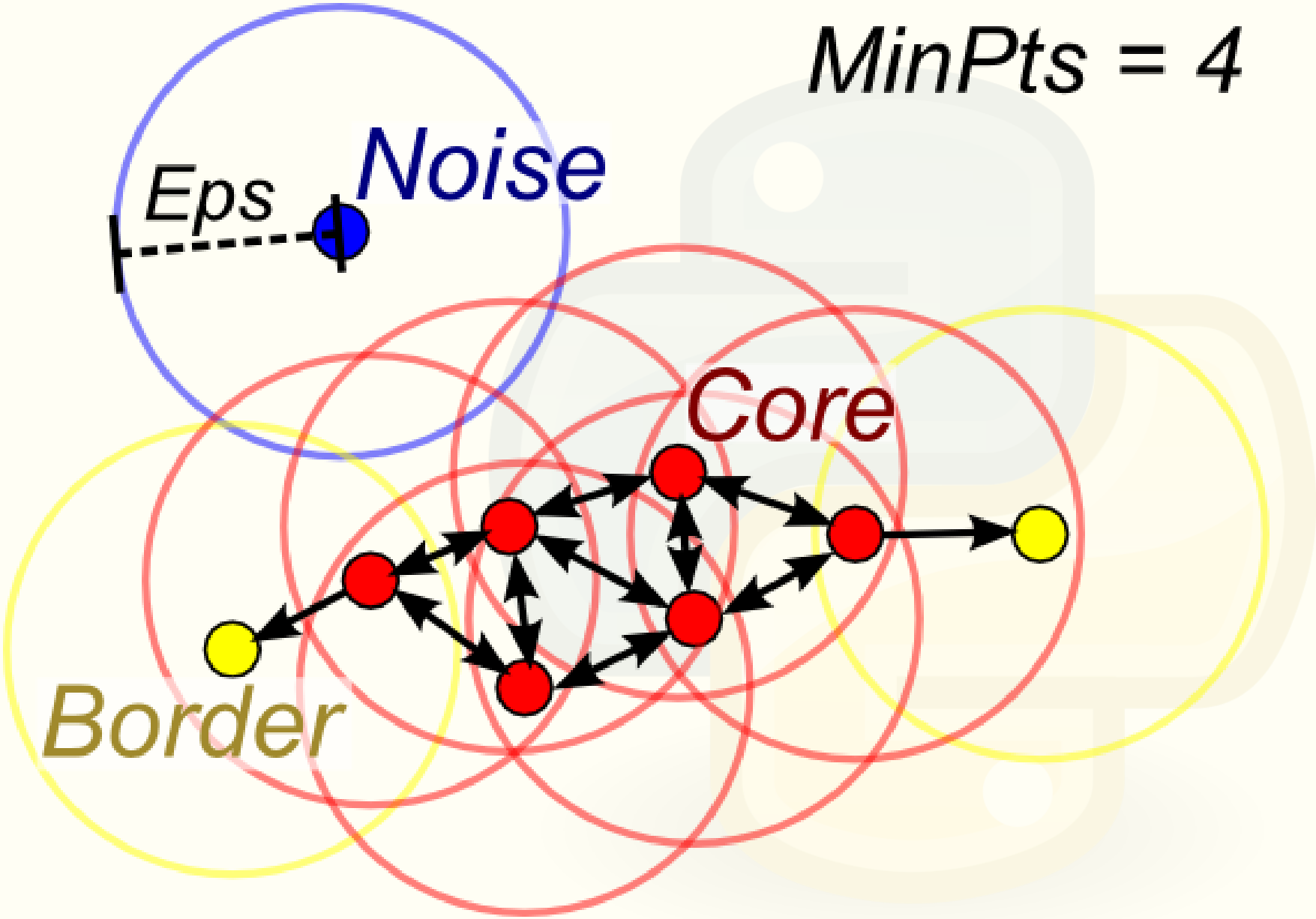
DBSCAN algorithm 2

```
DBSCAN(D, eps, min_samples) {
    C = 0
    for each point P in dataset D {
        if P is visited
            continue next point
        mark P as visited
        NeighborPts = regionQuery(P, eps)
        if sizeof(NeighborPts) < min_samples
            mark P as NOISE
        else {
            C = next cluster
            expandCluster(P, NeighborPts, C, eps, min_samples)
        }
    }
}
```

DBSCAN algorithm 3

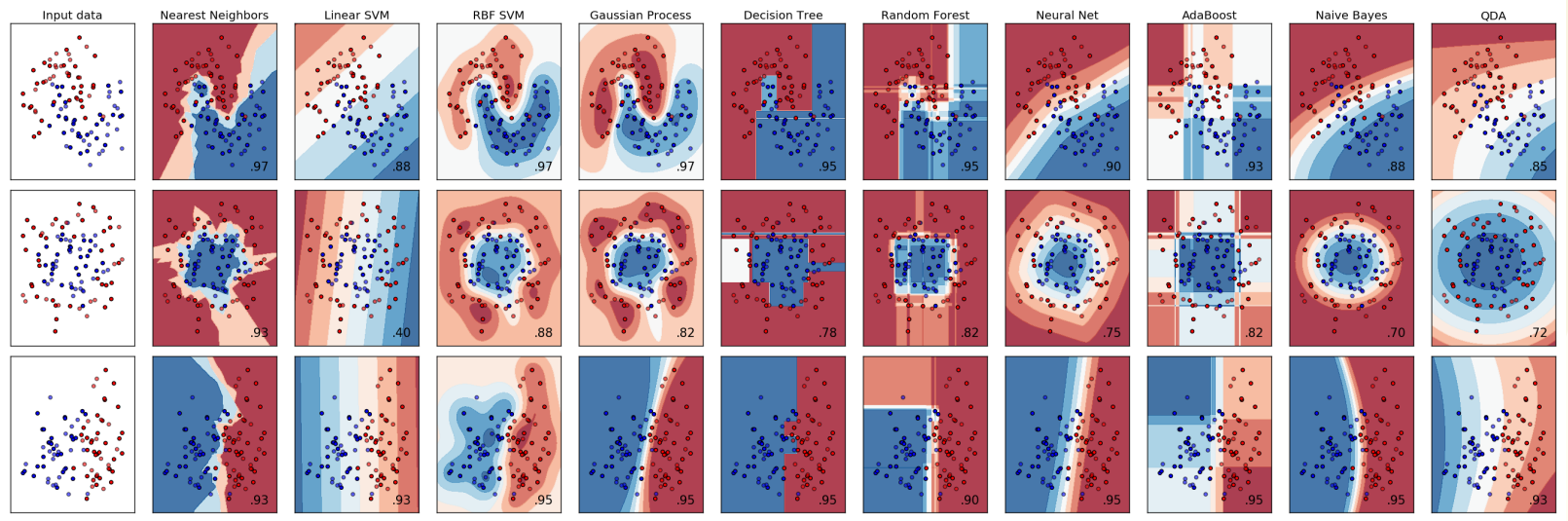
```
expandCluster(P, NeighborPts, C, eps, min_samples) {
    add P to cluster C
    for each point P' in NeighborPts {
        if P' is not visited {
            mark P' as visited
            NeighborPts' = regionQuery(P', eps)
            if sizeof(NeighborPts') >= min_samples
                NeighborPts = NeighborPts joined with
                 ↪  NeighborPts'
        }
        if P' is not yet member of any cluster
            add P' to cluster C
    }
}

regionQuery(P, eps)
    return all points within P's eps-neighborhood (including
     ↪  P)
```

DBSCAN algorithm 4



Scikit-learn 1

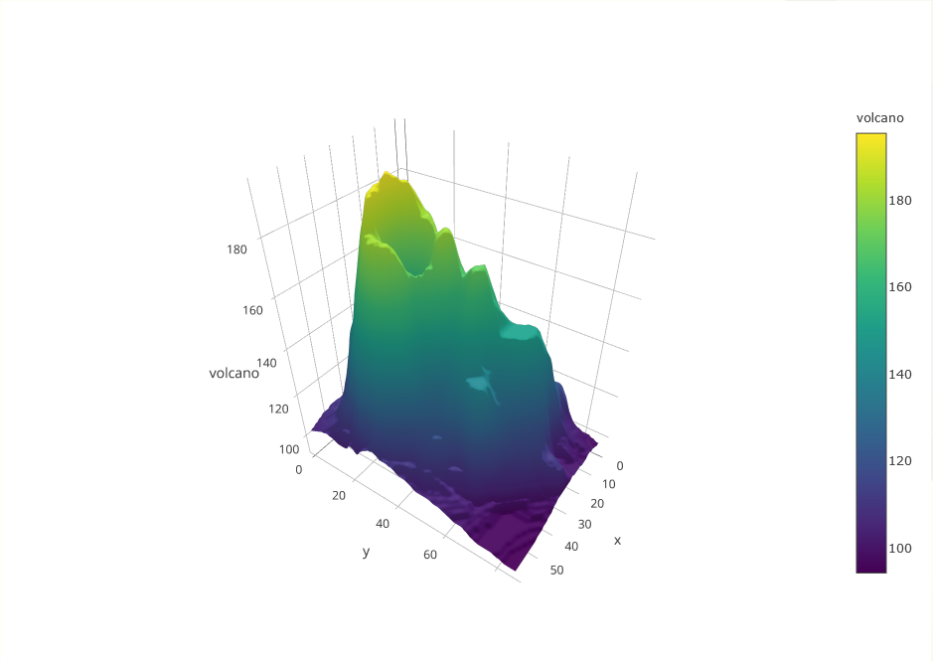
- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license



Plotly (interactive HTML embedded plots)

What is Plotly?

Plotly's Python is a graphing library that makes interactive, publication-quality graphs online.



- High quality plots
- Easy way to plot complex data
- Interactive plots
- HTML embedded plots
- Great community

Plotly 1

First of all, we need to execute the following commands:

```
import plotly.offline as py #this is the class that builds
 ↪  the embedded HTML with the plot
import plotly.graph_objs as go # Here are the classes that
 ↪  do different kinds of plots
```

For instance:

- go.Bar()
- go.Surface()
- go.Histogram()
- go.Box()
- go.Scatter()
- and many others...

https://plot.ly/python/reference/

Plotly 2

There are 4 main structures that we must define to plot something in plotly:

- Traces: all different datasets that we want to plot, and how we want to plot this sets, i.e. `trace1 = go.Bar(...)`
- Layout: object that we can use to change the configuration of the underlying window where we are plotting, for instance, we can change the background color, the height and the width,...
- Data: is a list of traces
- Figure: is a `graph_objs` that uses the data and the layout to plot the data with the configuration defined by the layout.

Now we are ready to open the **`plotly_examples.py` !!!**

Neural Networks and Deep Learning

If we achieve this point, I'll show you how to build manually a Neural Network, why a NN works, and some of the topics of the AI today.