

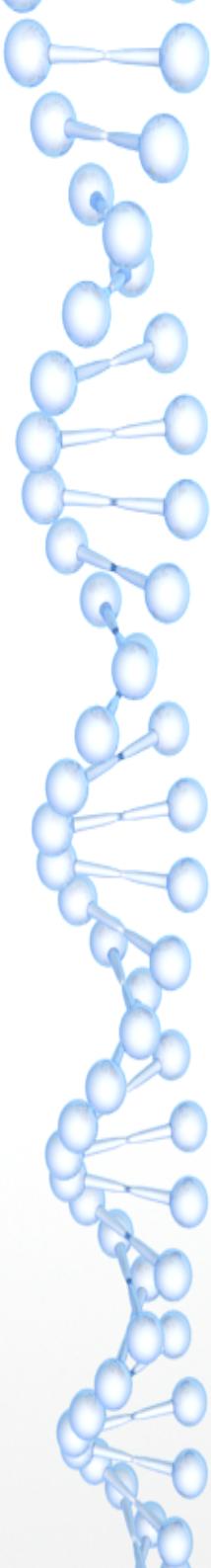
Visualizing scientific data using Python

A practical approach



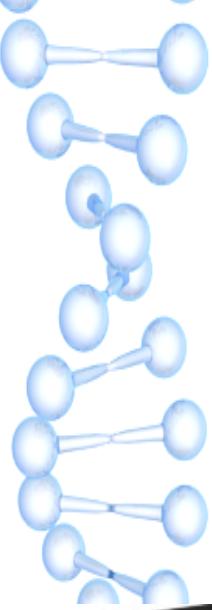
Dr. Augusto César Poot Hernández.
Postdoctoral fellow.
IIMAS, UNAM
augusto.poot@iimas.unam.mx
dragopoot@gmail.com

III Summer School in Bioinformatics.
June 10-24, 2018.
Santiago de Querétaro, Querétaro.



First, an analogy...

Mono Lake, July 1993.
Ken Rockwell.
Taken with a broken camera!!



What is the best tool?

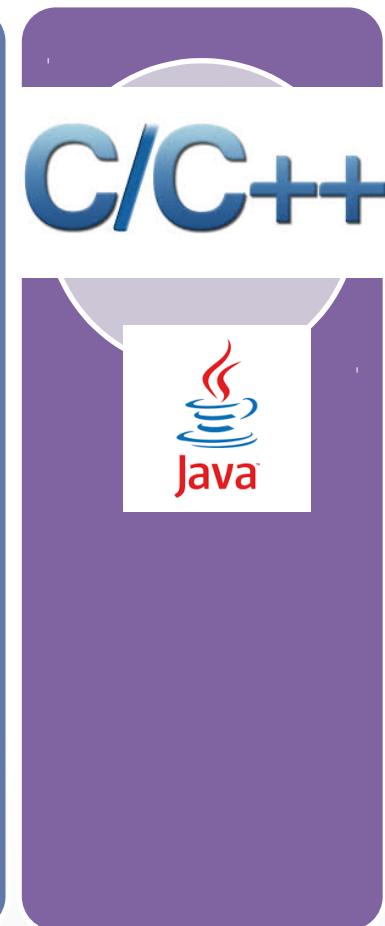
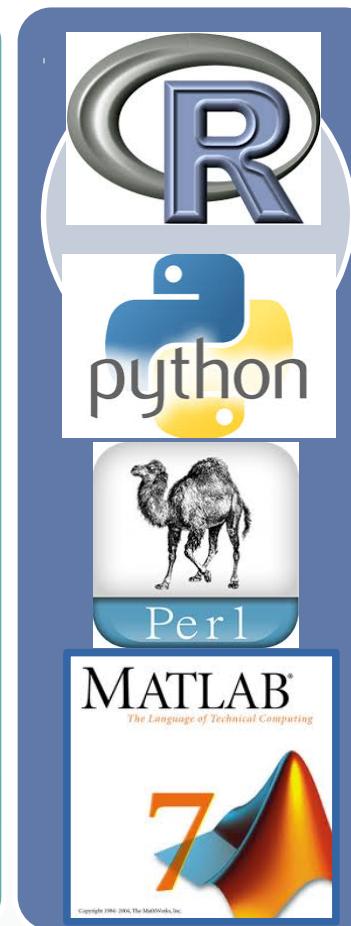
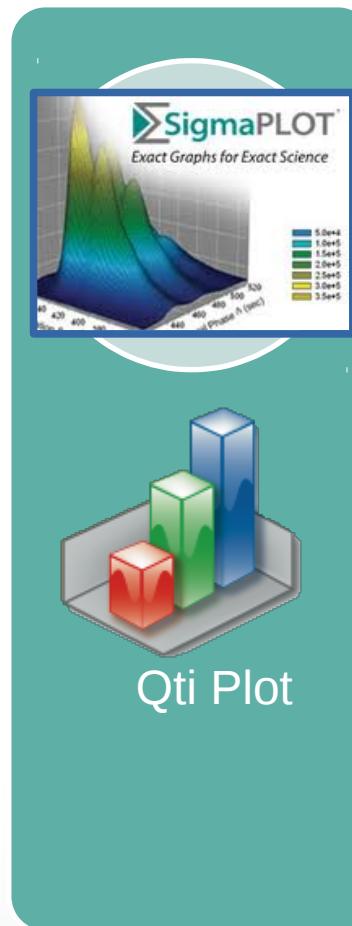
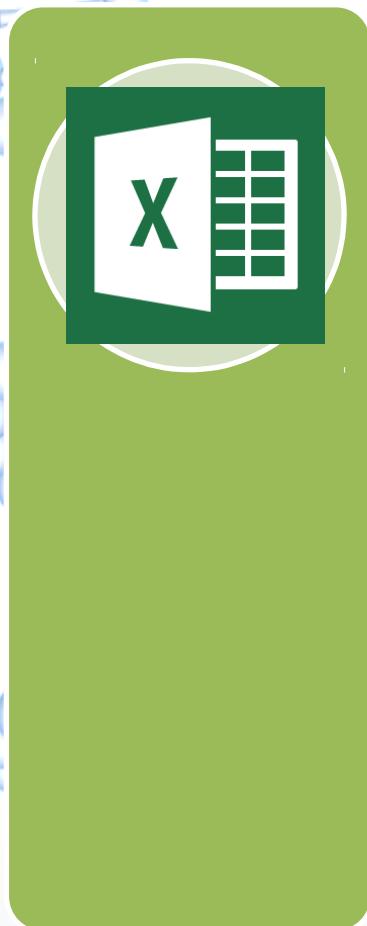


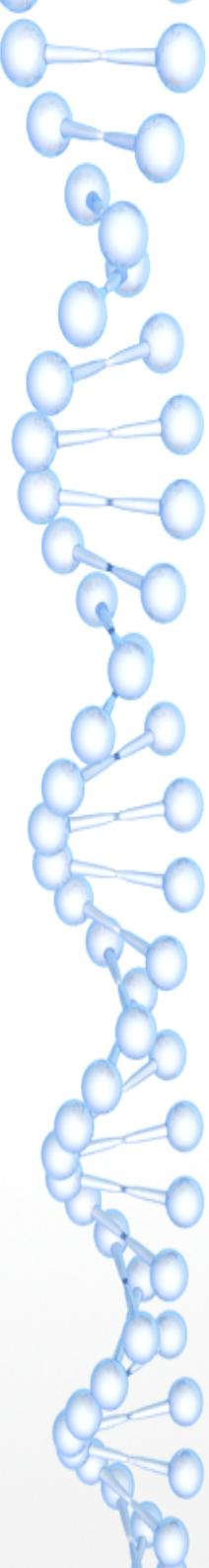
~ \$ 400 USD



~ \$ 5000 USD

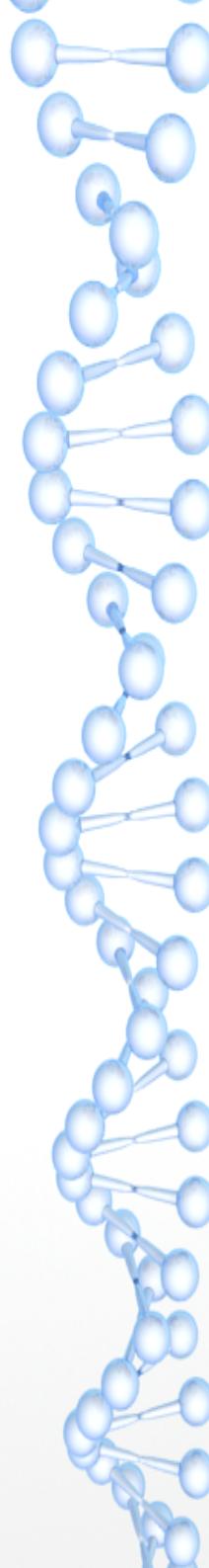
Same occurs with science.





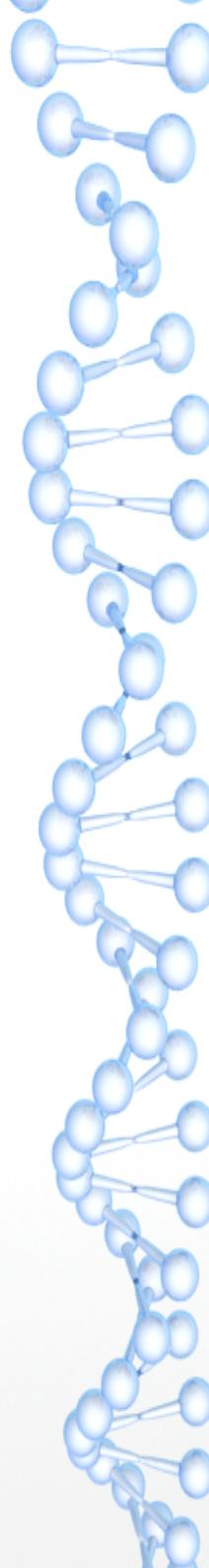
Take home.

- The best tool is the one that meets your needs.
- For this reason, the choice of the best tool is (hopefully) a personal decision.
- The mastery to use that tool comes with (a lot) of practice and perseverance.



My objectives.

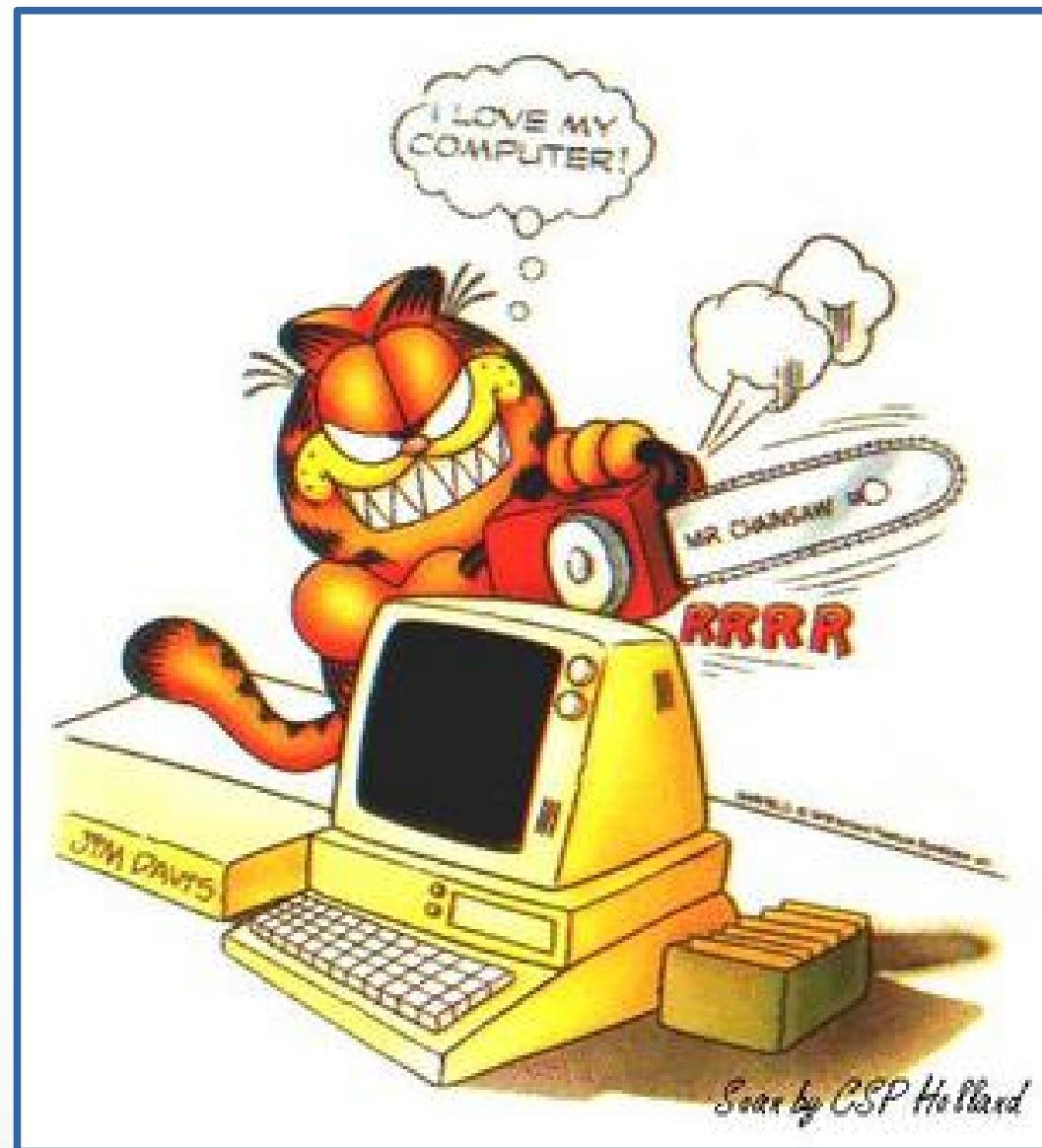
- Create a primer for the use of Python for scientific analysis.
- Convince you that python is a fun and easy to use alternative for this purpose.



Content.

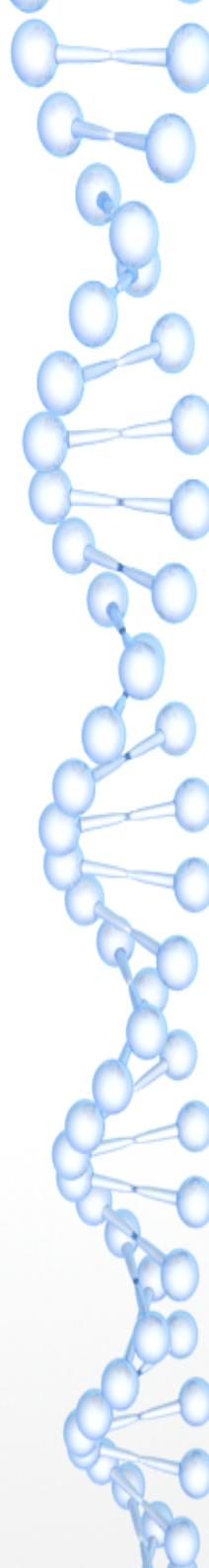
- 1) Python packages for scientific analysis.
- 2) iPython: Enhanced python interpreter.
- 3) Scientific python stack:
 - Numpy
 - Matplotlib
 - Scipy
 - Pandas

Be patient...

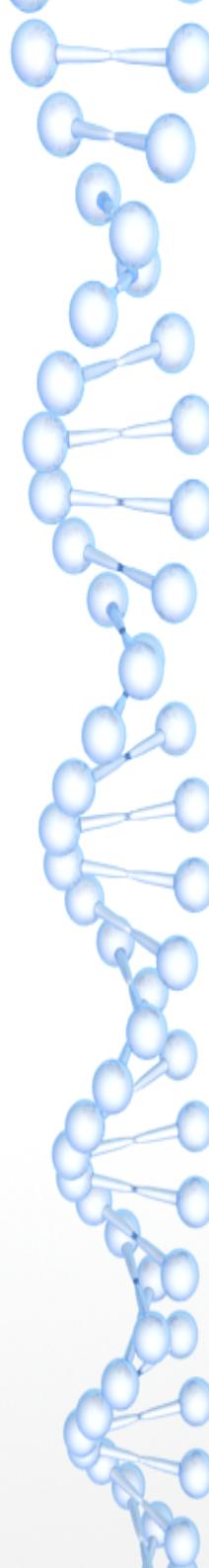




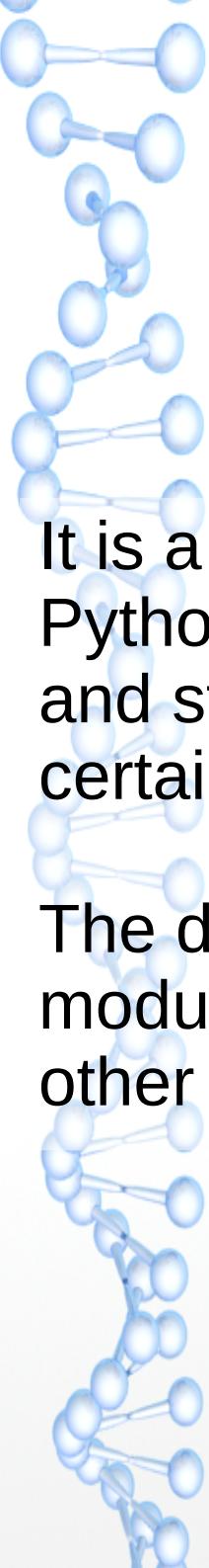
- Interpreted programming language
- Multi-purpose, multi-paradigm and multi-platform programming language.
- Emphasis on clean syntax.
- Excellent to learn and to prototyping.
- It is free software.
- A wide variety of (also free) modules or libraries.



It is said that Python is a programming language with **Batteries Included**, because the **Standard Library** contains enough functions for almost any application.



However you don't need to re-invent the wheel. **Specialized modules** for certain applications (science) make work much easier and speed up code production and execution.



What is a Python library or module?

It is a file(s) that contains Python code (definitions and statements) related to certain task.

The definitions in the module can be used by other python programs.

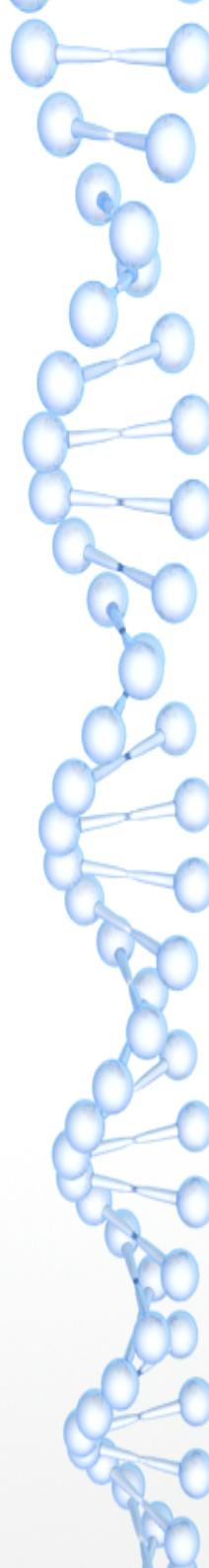
```
In [1]: import time
```

```
In [2]: time.strftime('%d / %m / %Y at %H hrs')  
Out[2]: '04 / 06 / 2018 at 10 hrs'
```

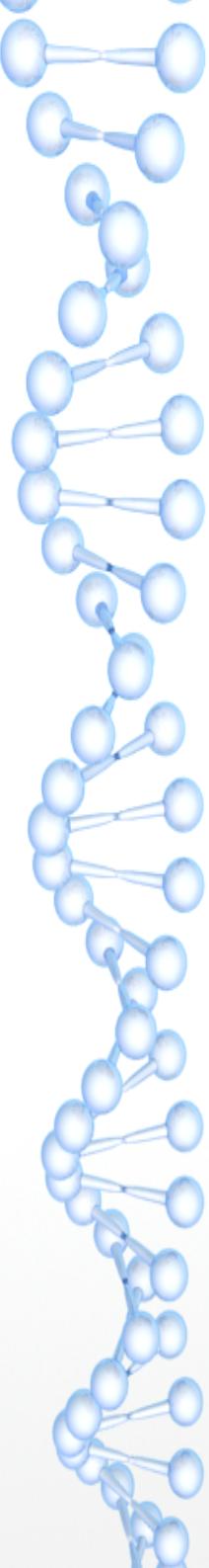
```
In [3]: import math
```

```
In [4]: math.pi  
Out[4]: 3.141592653589793
```

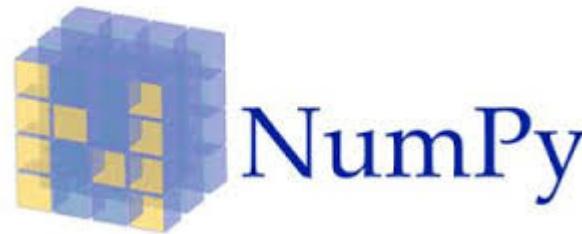
```
In [5]: |
```



There are A LOT scientific related modules
for Python.

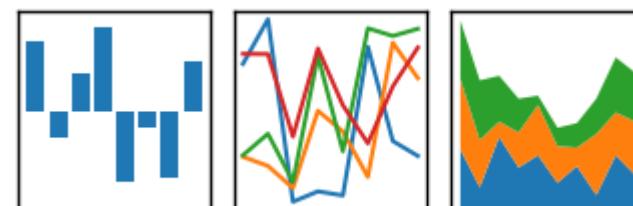


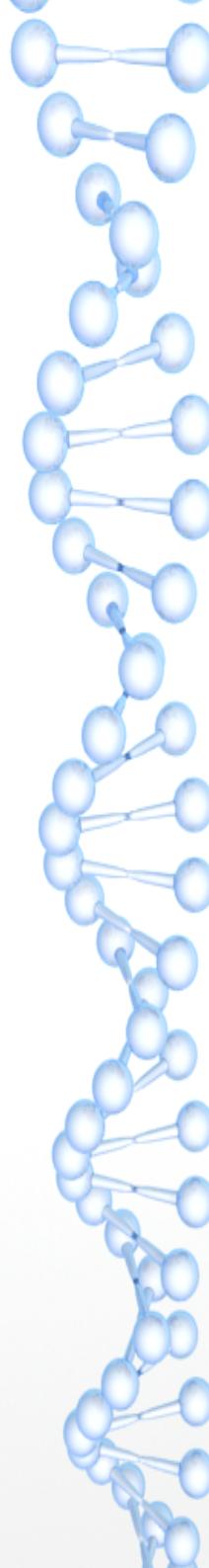
Fantastic four



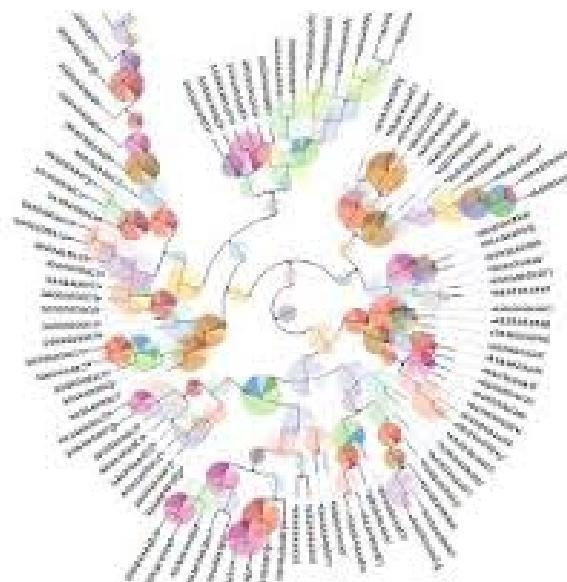
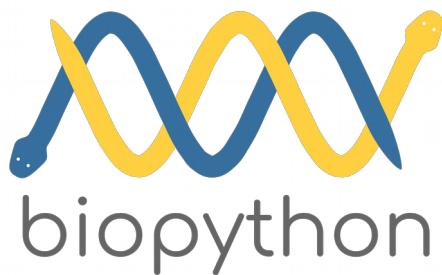
pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$





The X-men



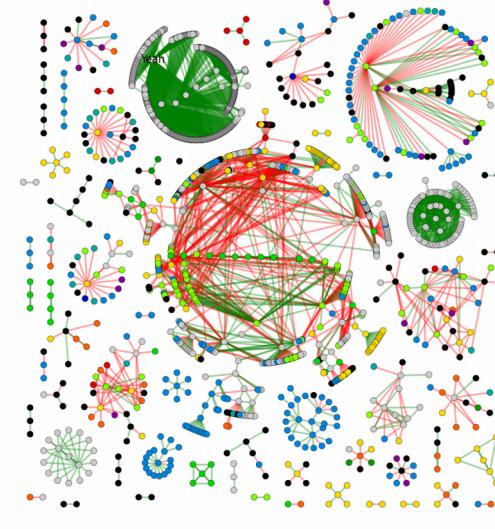
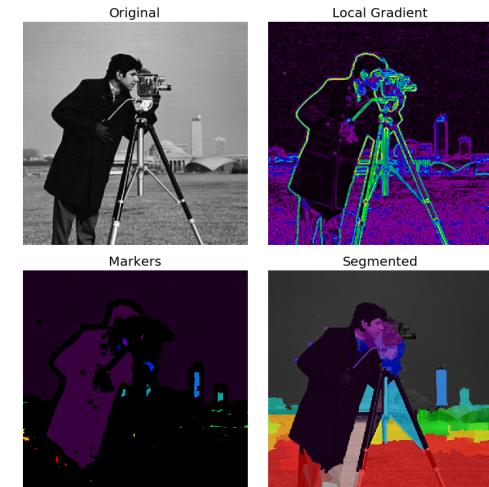
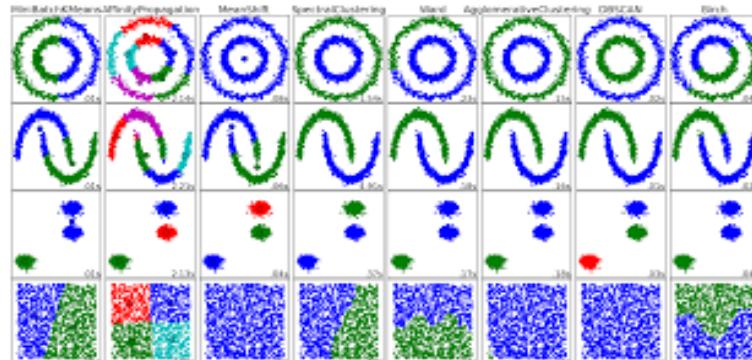
ETE toolkit



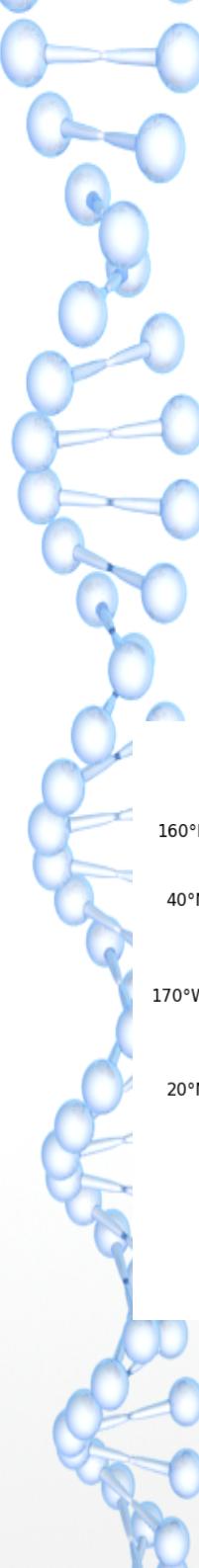
DendroPy

Python tools for epidemiology

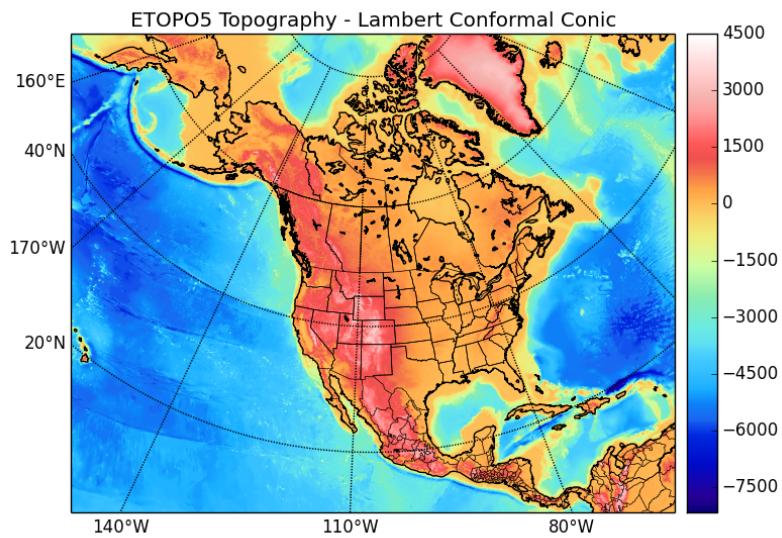
The Avengers



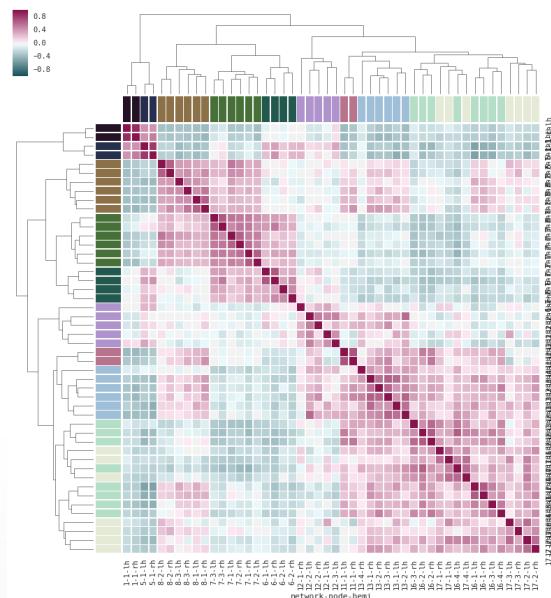
NetworkX

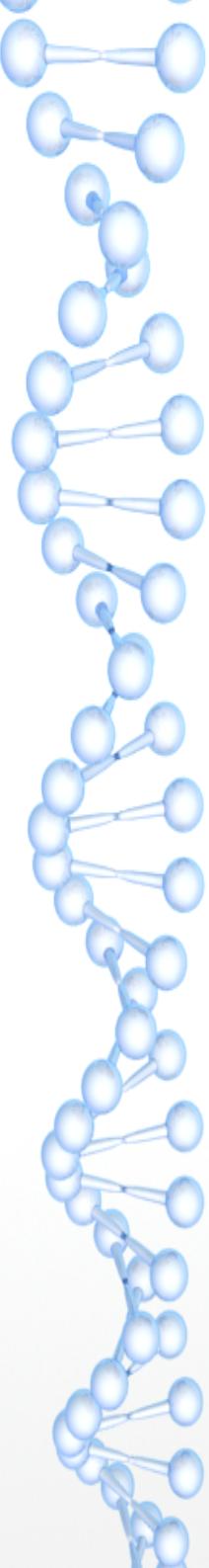


Basemap



Seaborn





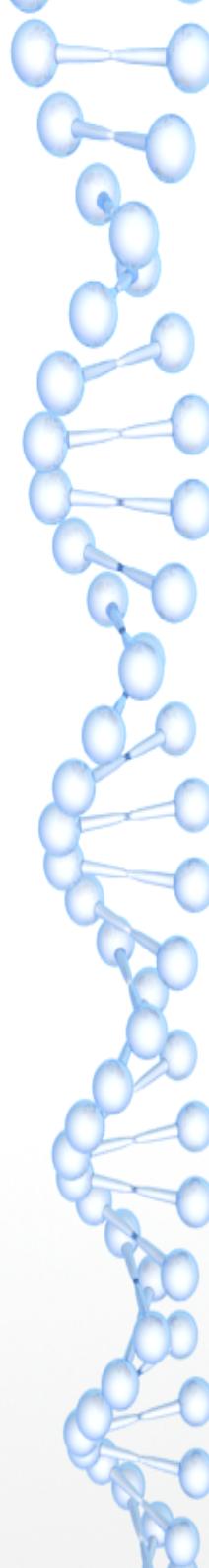
On the edge of machine learning...

Artificial Neural Networks

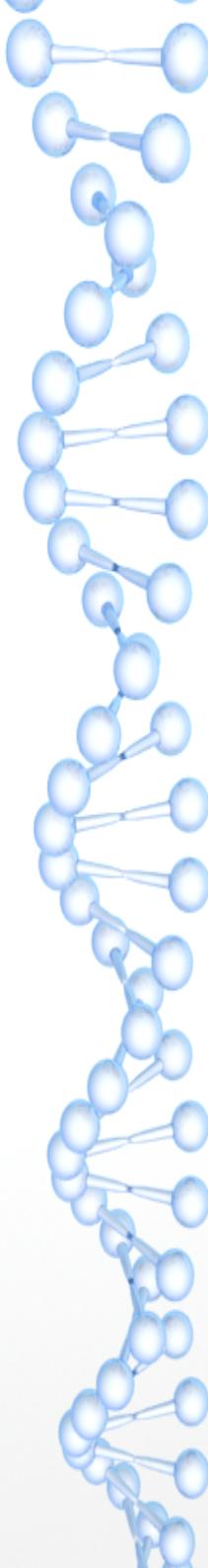


TensorFlow

API for python

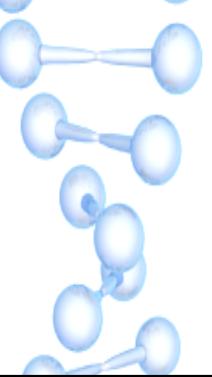


And a lot more...



How are we going to work?

- We will follow script-like files (.py) that contain the code to test in class.
 - !!! These files are not written to be executed as a script.
- We will execute (write) the lines of code interactively on iPython interpreter.
- We will explain the statements and the results.



Python 2 Vs Python 3 (Some differences)

```
In [1]: print "Hello World"  
Hello World
```

```
In [2]: 3/2  
Out[2]: 1
```

```
In [3]: 3./2  
Out[3]: 1.5
```

```
In [4]: range(0,10)  
Out[4]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
In [5]: xrange(0,10)  
Out[5]: xrange(10)
```

```
In [6]: dicta = {'a': 'first', 'b': 'second'}
```

```
In [7]: dicta.keys()  
Out[7]: ['a', 'b']
```

```
In [1]: print("Hello World")  
Hello World
```

```
In [2]: 3/2  
Out[2]: 1.5
```

```
In [3]: 3//2  
Out[3]: 1
```

```
In [4]: range(0,10)  
Out[4]: range(0, 10)
```

```
In [5]: list(range(0,10))  
Out[5]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
In [6]: dicta = {'a': 'first', 'b': 'second'}
```

```
In [7]: dicta.keys()  
Out[7]: dict_keys(['a', 'b'])
```

```
In [8]: list(dicta.keys())  
Out[8]: ['a', 'b']
```

print

division

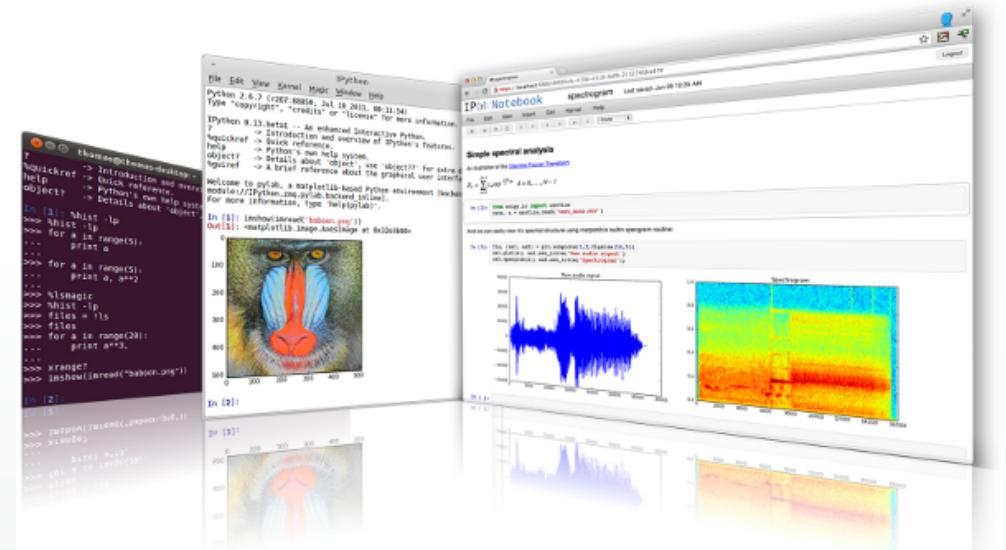
range

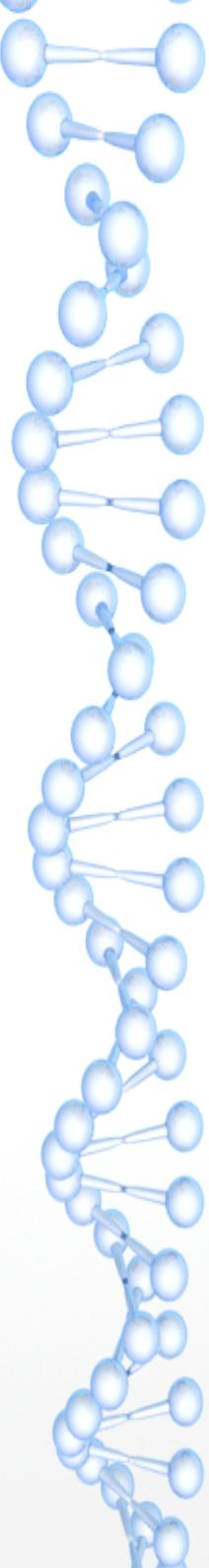
dictionaries

IP[y]: IPython

Interactive Computing

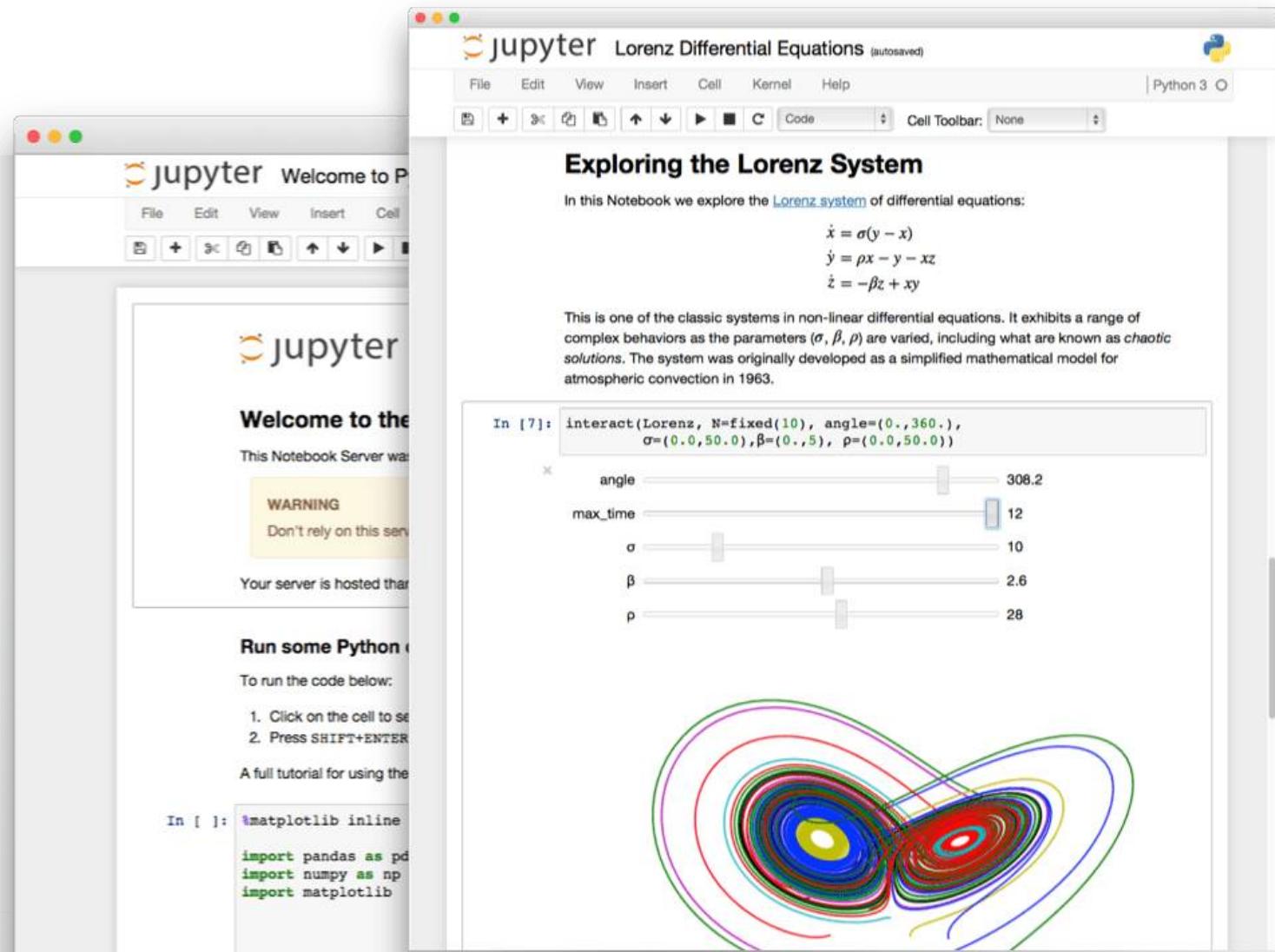
- A powerful interactive shell for Python.
- Interactive data visualization.
- Easy to use, high performance tools for parallel computing.





Jupyter notebooks

Open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text.



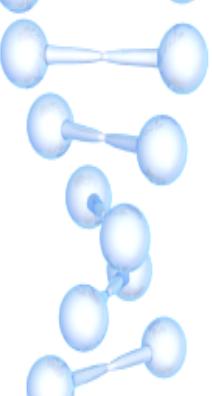
The screenshot shows the Jupyter Notebook interface with the title "Lorenz Differential Equations". The notebook displays the following content:

- Exploring the Lorenz System**: A section describing the Lorenz system of differential equations with the equations:
$$\begin{aligned}\dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy\end{aligned}$$
- A note explaining that it is one of the classic systems in non-linear differential equations, exhibiting chaotic behavior as parameters are varied.
- An IPython code cell (In [7]) containing:

```
interact(Lorenz, N=fixed(10), angle=(0.,360.),
         σ=(0.0,50.0), β=(0.,5), ρ=(0.0,50.0))
```

with sliders for angle, max_time, σ, β, and ρ.
- A 3D plot of the Lorenz attractor, showing the characteristic butterfly shape.
- Code cells at the bottom:

```
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib
```



Scientific PYthon Development EnviRonment.



File Edit Search Source Run Debug Interpreters Tools View ?

Editor - /home/francesco/Projects/Digital Signal Processing/My/hw17.py

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Sat Jul  5 15:46
4 @author: francesco
5 """
6
7
8 import numpy
9 import pylab
10
11 def db2lin(db):
12     return numpy.power(10.0
13
14 def lin2db(lin):
15     return 10.0*numpy.log10
16
17 kHz = 1000.0
18 MHz = kHz * kHz
19 GHz = MHz * kHz
20
21 print "q1"
22 SNR_db = 20
23 W = 6 * MHz
24
25 SNR = db2lin(SNR_db)
26 C = W * numpy.log2(1 + SNR)
27 print "C = %g bps" % C
28
29 print "q3"
30
31 def check_orth(m1, m2, n):
32     orth = numpy.dot(numpy.array(m1), numpy.array(m2))
33     if (numpy.abs(orth) > 1):
34         result = "KO"
35     else:
36         result = "OK"
37     print "%s (%g)" % (result, orth)
38
39 def plot_orth(m1, m2, n):
40     pylab.figure(1)
41     pylab.clf()
42     pylab.plot(n, numpy.asarray(map(m1, n)), 'k')
43     pylab.hold(True)
44     pylab.plot(n, numpy.asarray(map(m2, n)), 'b')
45     pylab.plot(n, numpy.asarray(map(m1, n)) * numpy.asarray(map(m2, n)), 'r')
46     pylab.hold(False)
```

Figure 1

Variable explorer

Name	Type	Size	Value
A_size	int	1	32
C	float64	1	39949268.896510765
E	float64	(3, 3)	array([[0.866, 0. , 1.732], [0.5 , 0.5 , 3.], [-0.866, 0. , -1.732]])
Fmax	float	1	1000000.0
Fmin	float	1	400000000.0
Fs	float	1	2000000.0
GHz	float	1	1000000000.0
H_dft	list	333773	<list @ 0xB0B6B02CL>
H_dfft	list	1024	<list @ 0xB2A616ECL>
K	list	5	<list @ 0xB2A971ACL>
M	float64	1	10.0
MHz	float	1	1000000.0
M_range	list	4	<list @ 0xR0RC59CCI>

Object inspector Variable explorer File explorer

Console

Python 1

```
OK (-8.6484e-16)
OK (-1.17854e-15)
OK (3.05058e-14)
q5
s[n+0.250000] = -2.250000 (N=1)
s[n+0.000000] = -2.000000 (N=4)
s[n+0.500000] = -3.125000 (N=2)
s[n+-0.250000] = -0.968750 (N=2)
q7
W = 125000.000000
bps = 500000.000000 (M: 4)
bps = 750000.000000 (M: 6)
bps = 1250000.000000 (M: 10)
>>>
```

Permissions: RW End-of-lines: LF Encoding: UTF-8 Line: 6 Column: 1 Memory: 77 %



ANACONDA
Powered by Continuum Analytics®

Python distribution that includes
hundreds of packages related to
science.

Anaconda Navigator

File Help

Sign in to Anaconda Cloud

Home Environments Learning Community Documentation Developer Blog Feedback

Refresh

Applications on base (root) Channels

lab jupyterlab 0.32.1 An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture. Launch

jupyter notebook 5.5.0 Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis. Launch

orange3 3.13.0 Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox. Launch

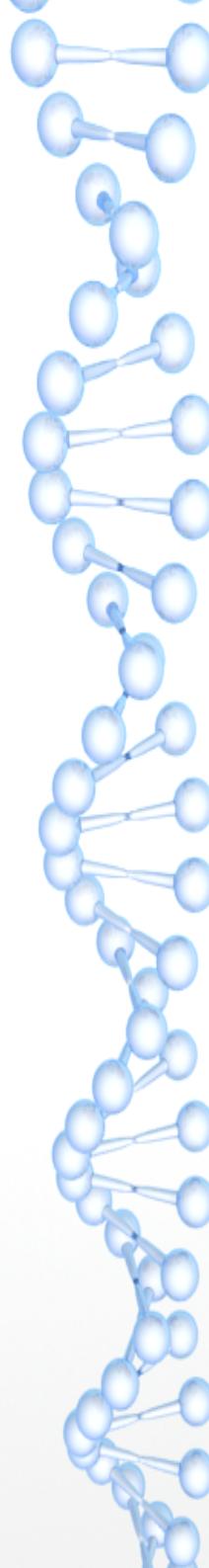
IP[y]: qtconsole 4.3.1 PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more. Launch

spyder 3.2.8 Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

glueviz 0.13.3 Multidimensional data visualization across files. Explore relationships within and among related datasets.

rstudio 1.1.423 A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

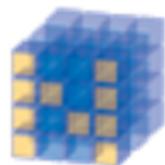
The Anaconda Navigator interface displays a grid of scientific applications. Each application card includes its name, version, a brief description, and a 'Launch' button. The sidebar on the left provides navigation links for Home, Environments, Learning, Community, Documentation, Developer Blog, and Feedback, along with social media links for Twitter, YouTube, and GitHub.



The Fantastic Four

The scipy scientific environment.

SciPy (pronounced “Sigh Pie”) is a Python-based ecosystem of open-source software for mathematics, science, and engineering. In particular, these are some of the core packages:



NumPy
Base N-dimensional array package



SciPy library
Fundamental library for scientific computing



Matplotlib
Comprehensive 2D Plotting



IPython
Enhanced Interactive Console

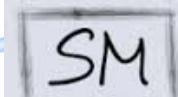
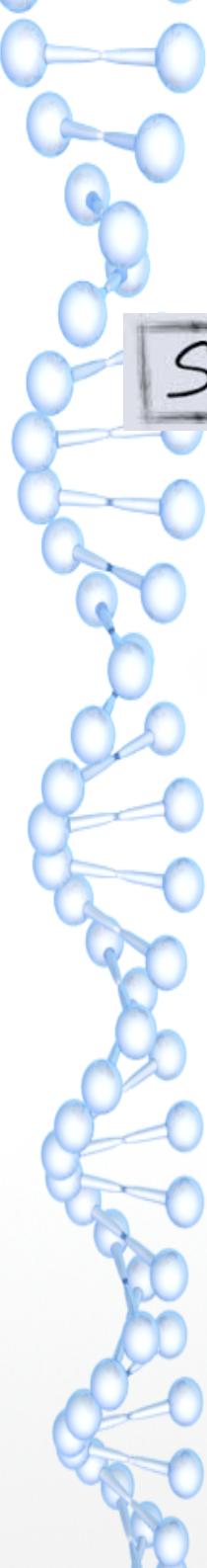


Sympy
Symbolic mathematics



pandas
Data structures & analysis

Numpy stack



StatsModels
Statistics in Python



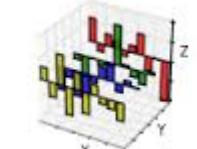
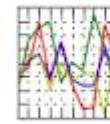
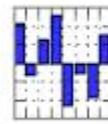
And more!!!



scikit-image
image processing in python

pandas

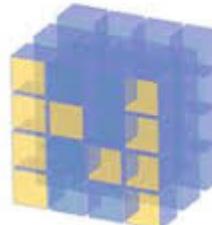
$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



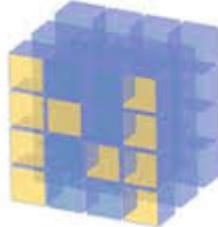
SciPy



matplotlib



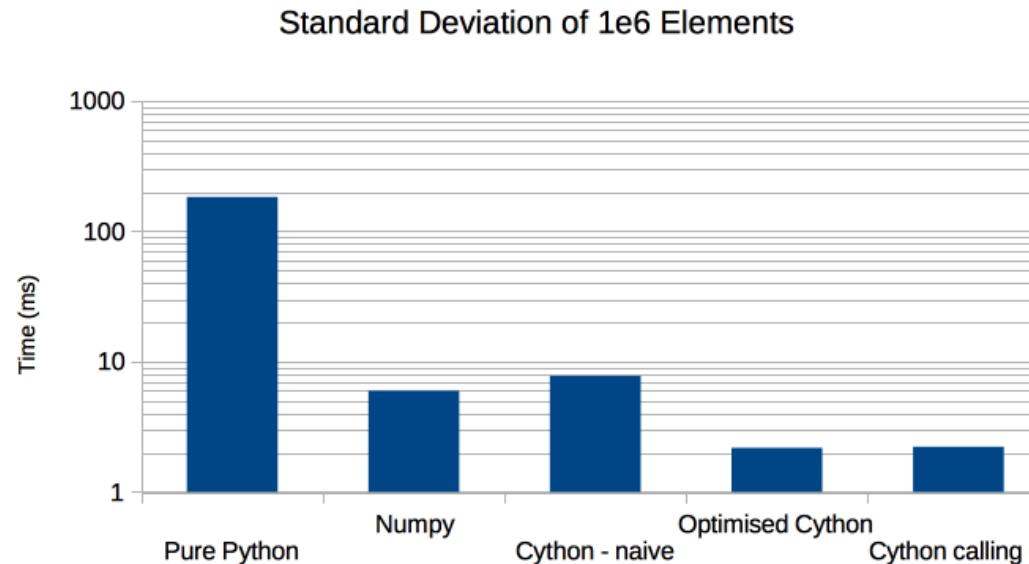
NumPy

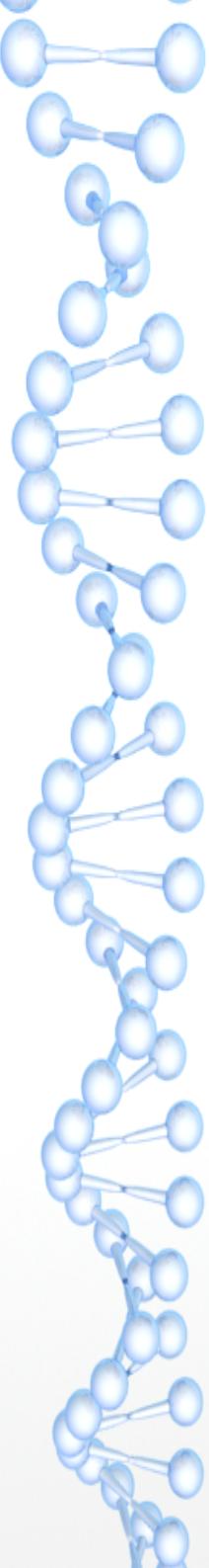


NumPy

N-dimensional numeric arrays (*ndarrays*).

- Implemented in c/c++ , fortran.
- Functions related to *ndarrays*.
- Linear algebra, FFT, random number capabilities.



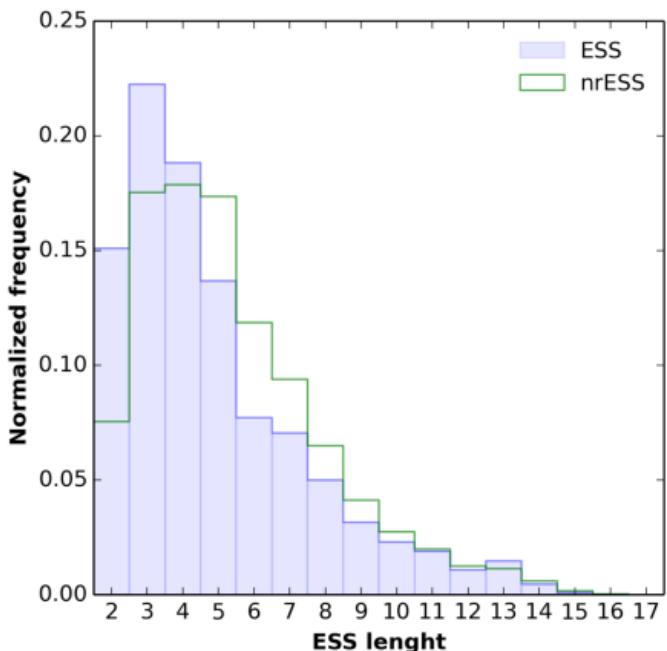
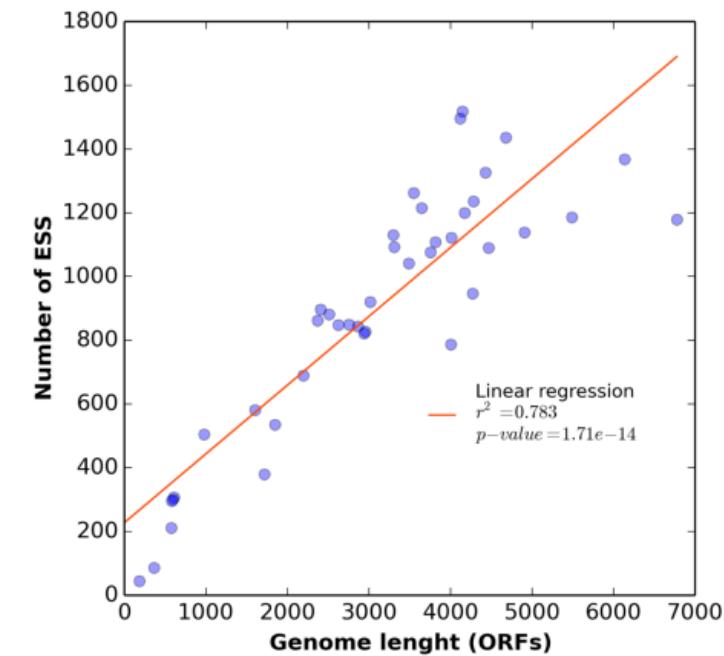
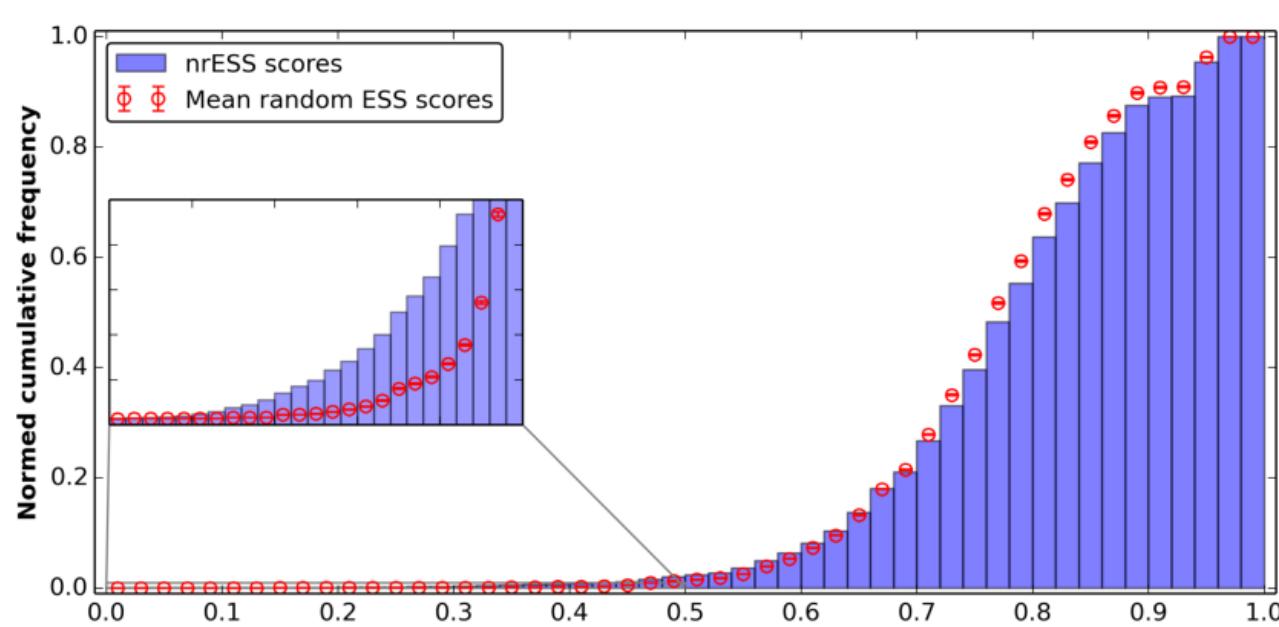
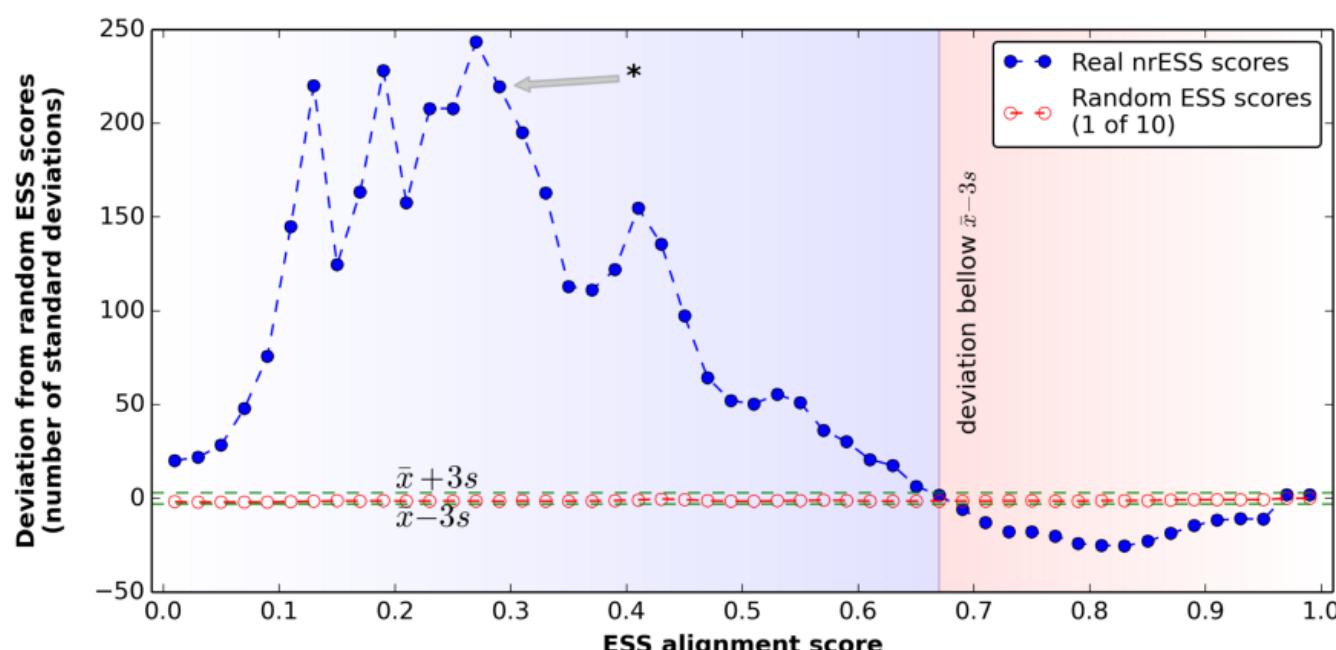


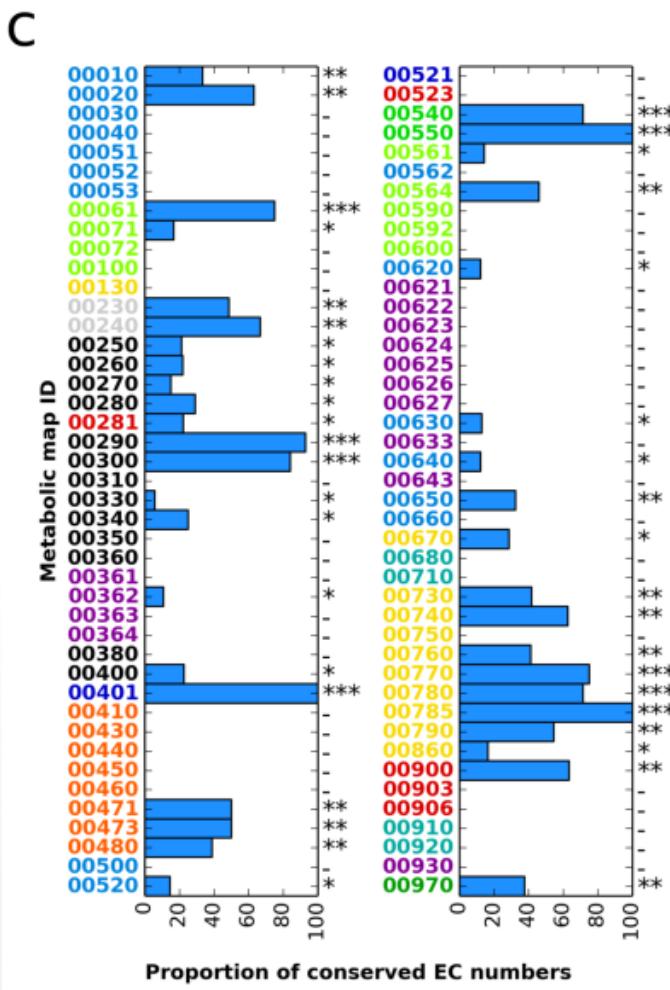
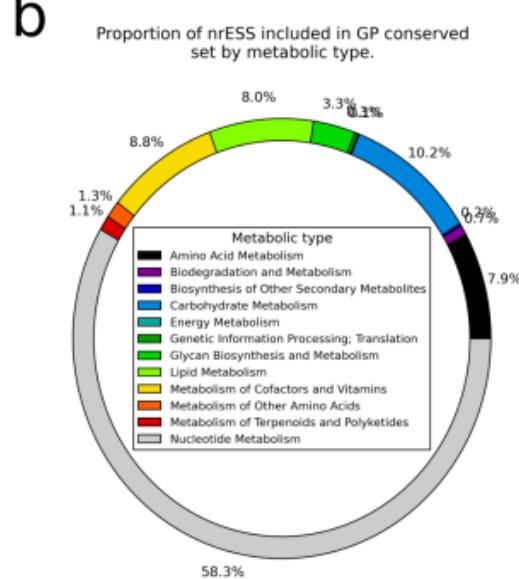
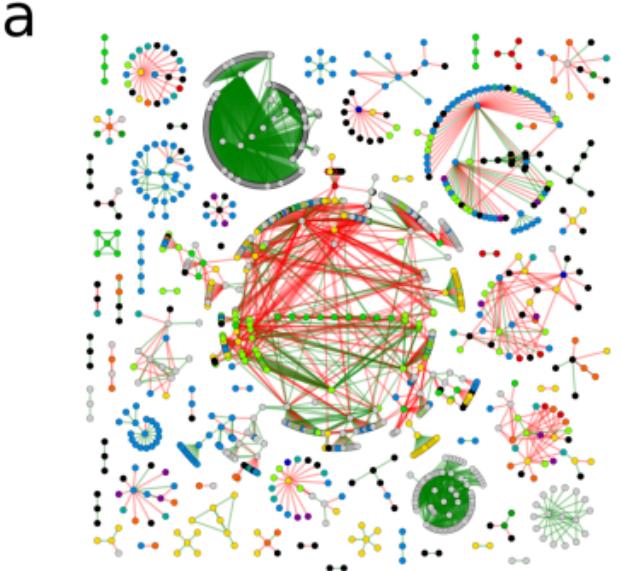
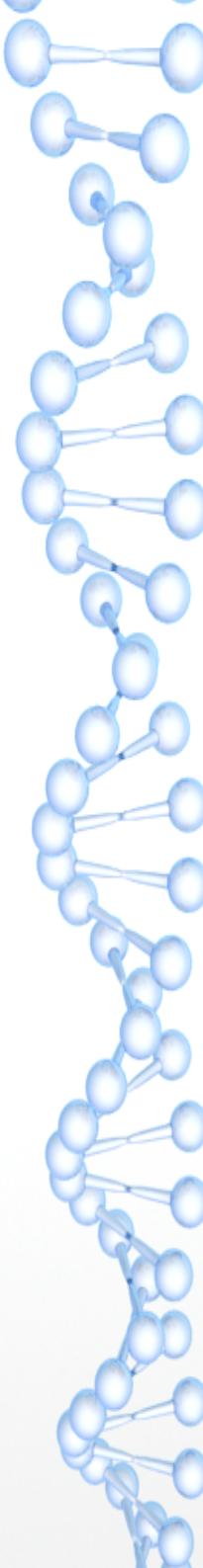
matplotlib



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

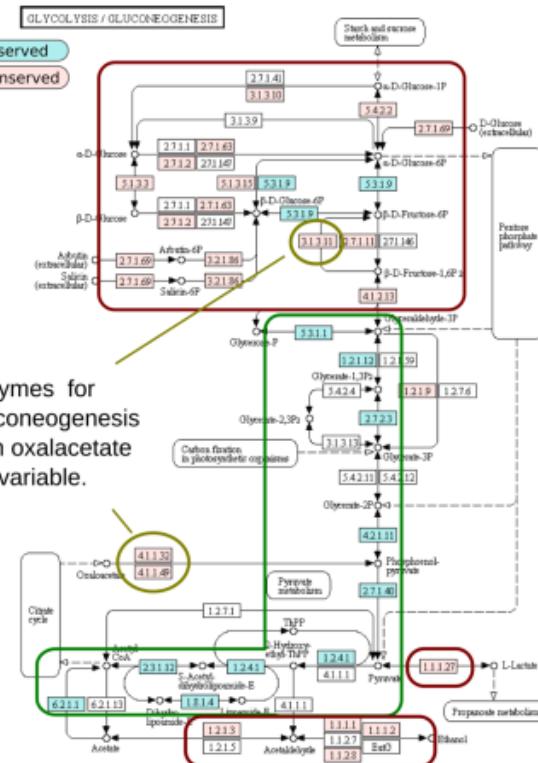
The matplotlib website at:
<https://matplotlib.org/index.html>

a**b****c****d**

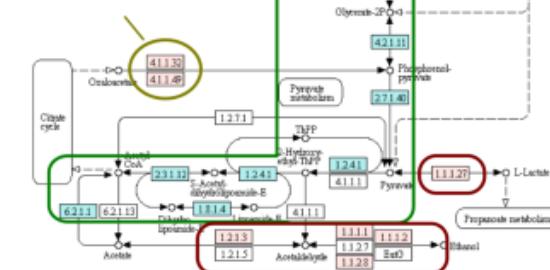


d

Variable: Hexose part of Glycolysis and input pathways. Ethanolic and lactic fermentation.

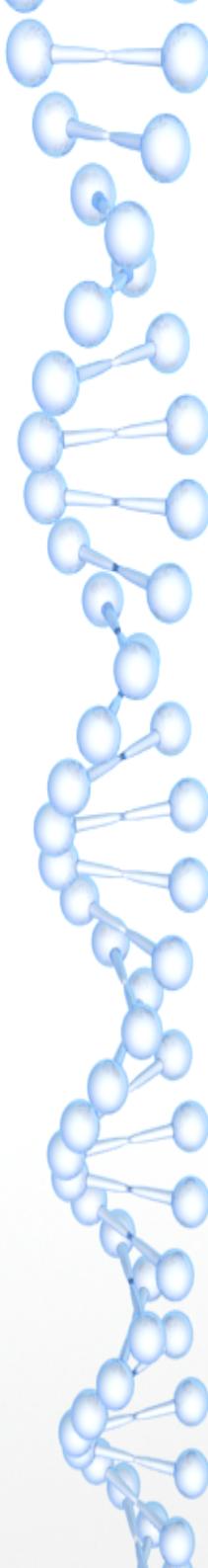


Enzymes for Gluconeogenesis from oxalacetate are variable.

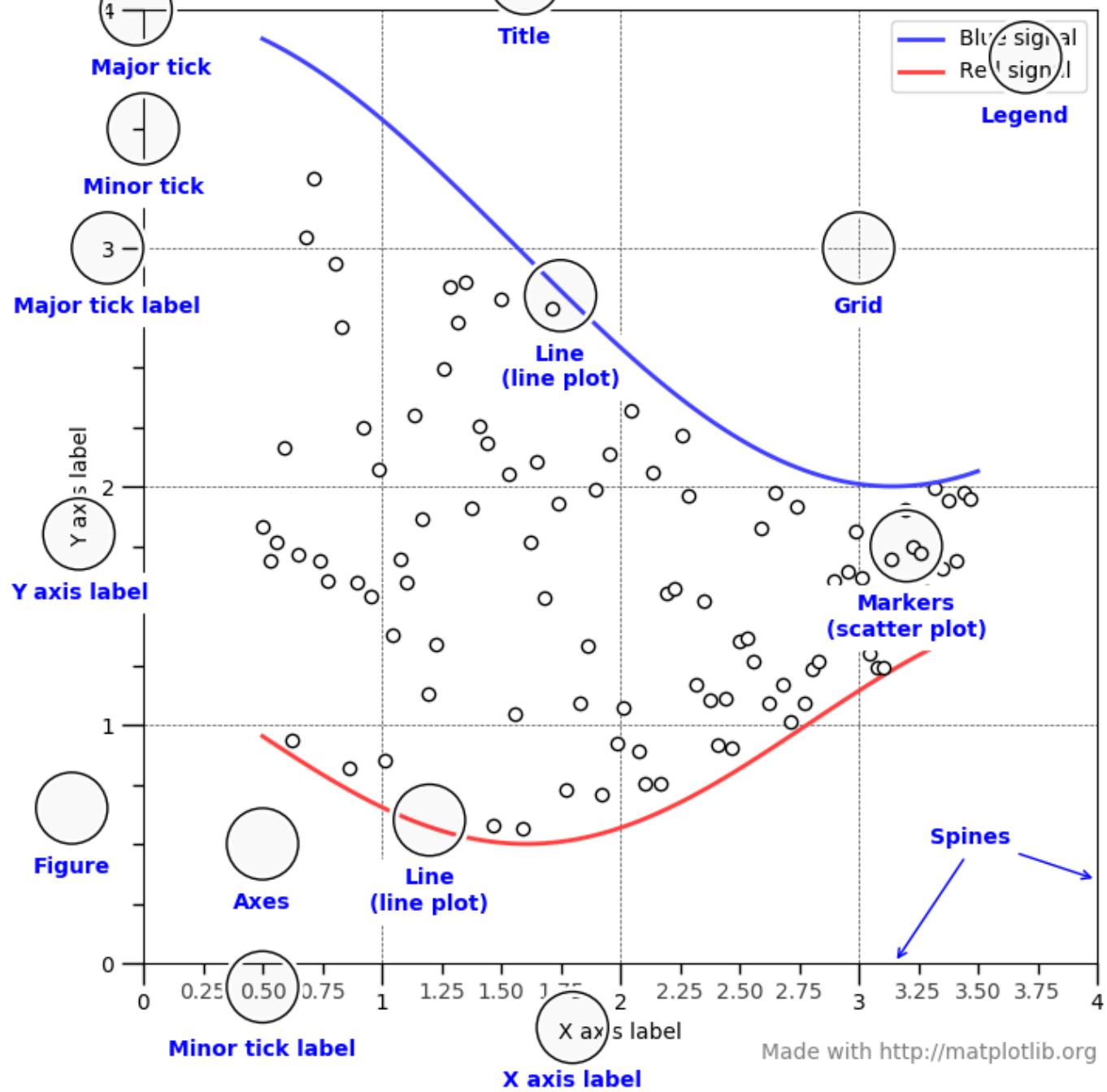


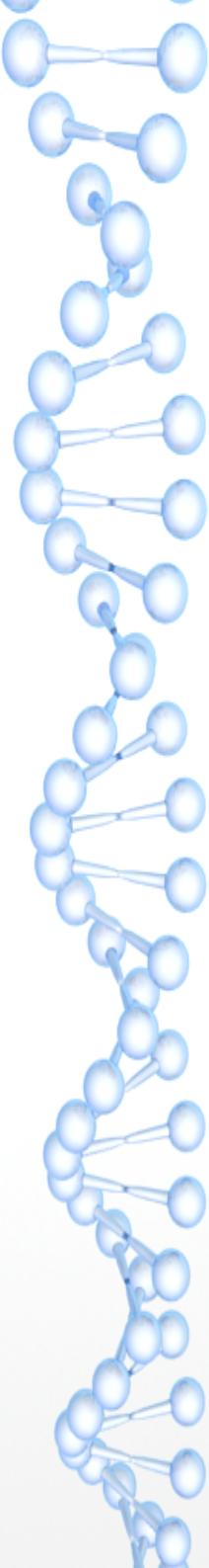
Functionally conserved: Tri-carbon core of the Glycolysis and oxidation of pyruvate to Acetyl CoA.

Poot-Hernandez, et al.
2015. BMC genomics.
16.



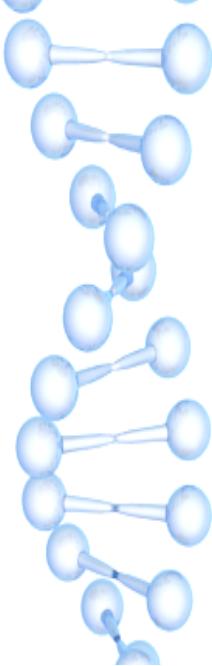
Anatomy of a figure





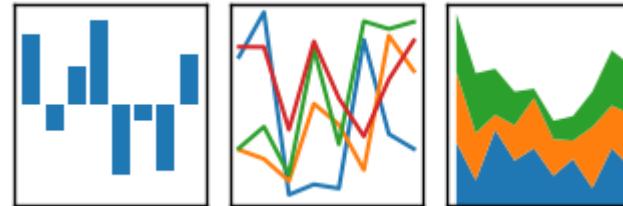
Additional routines needed
in scientific work

- **constants**: physical constants and conversion factors
- **cluster**: hierarchical clustering, vector quantization, K-means
- **fftpack**: Discrete Fourier Transform algorithms
- **integrate**: numerical integration routines
- **interpolate**: interpolation tools
- **linalg**: linear algebra routines
- **misc**: miscellaneous utilities (e.g. image reading/writing)
- **ndimage**: various functions for multi-dimensional image processing
- **optimize**: optimization algorithms including linear programming
- **signal**: signal processing tools
- **sparse**: sparse matrix and related algorithms
- **spatial**: KD-trees, nearest neighbors, distance functions
- **special**: special functions
- **stats**: statistical functions
- **weave**: tool for writing C/C++ code as Python multiline strings



pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$

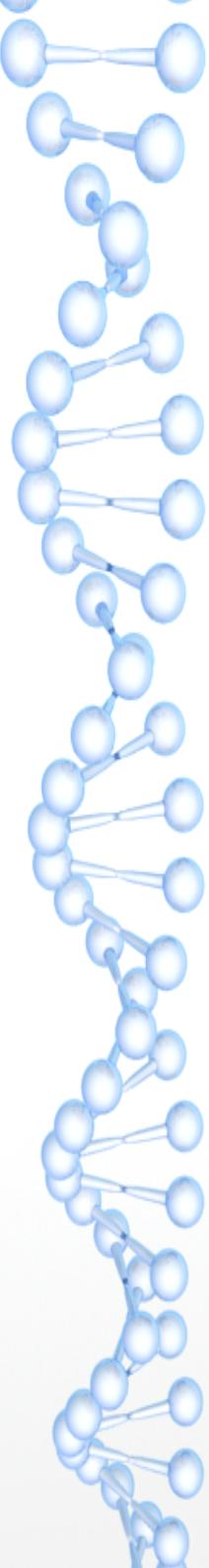


- Data structures and data analysis tools.

Data structures at a glance

Dimensions	Name	Description
1	Series	1D labeled homogeneously-typed array
2	DataFrame	General 2D labeled, size-mutable tabular structure with potentially heterogeneously-typed columns
3	Panel	General 3D labeled, also size-mutable array





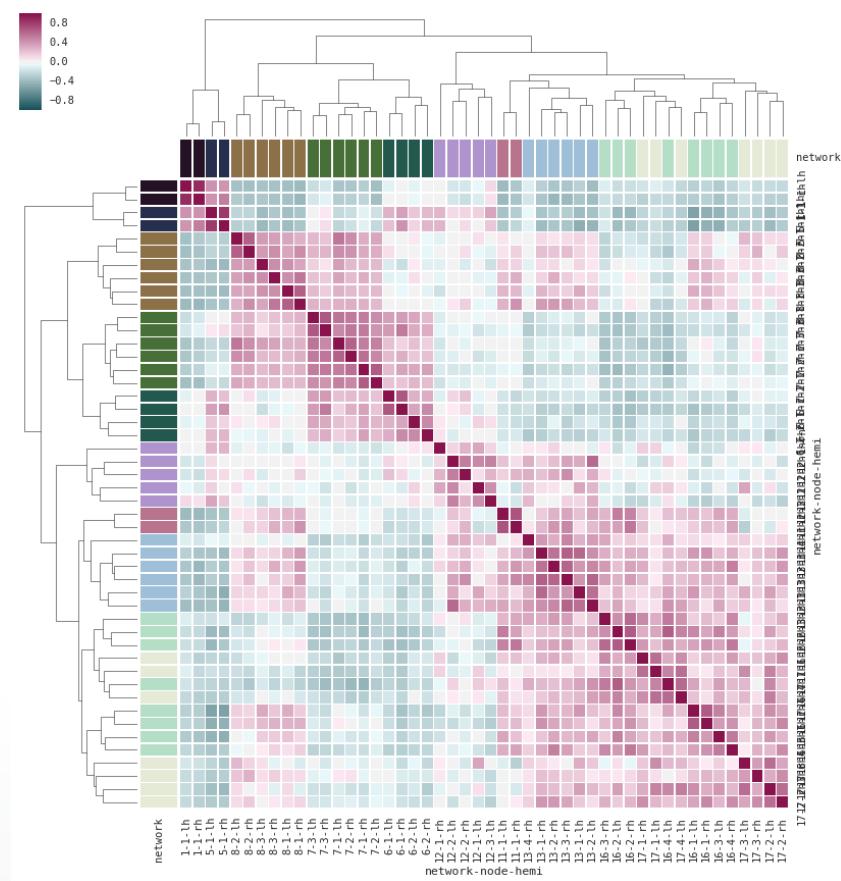
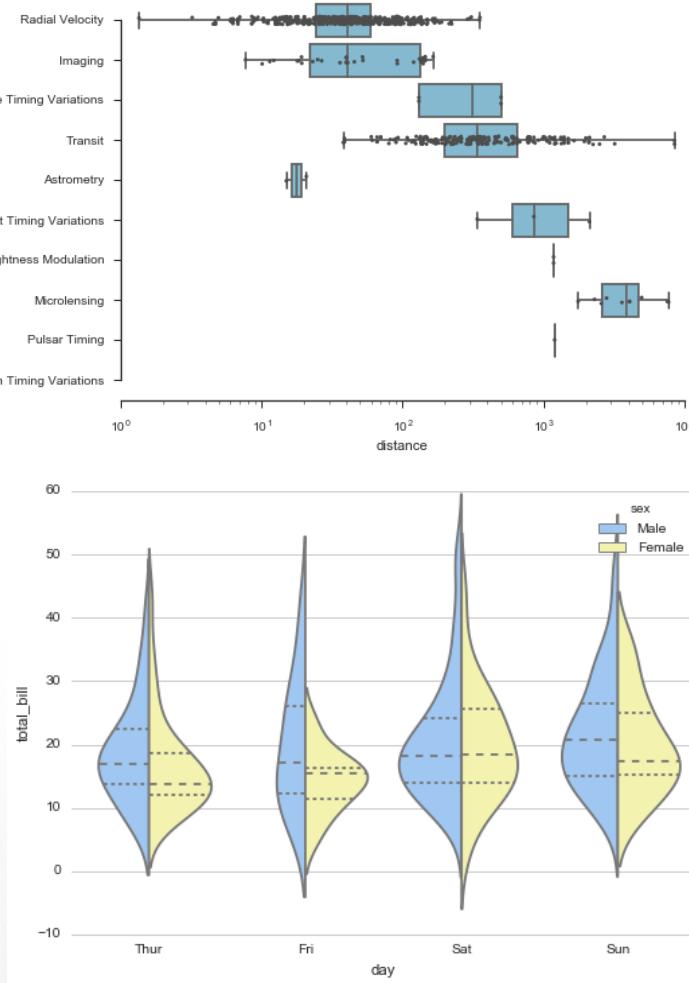
Pandas DataFrame

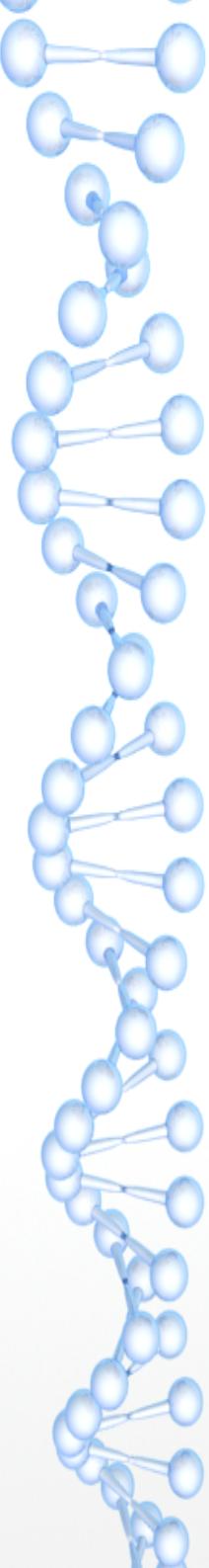
		A	B	C	D
2013-01-01	0.469112	-0.282863	-1.509059	-1.135632	
2013-01-02	1.212112	-0.173215	0.119209	-1.044236	
2013-01-03	-0.861849	-2.104569	-0.494929	1.071804	
2013-01-04	0.721555	-0.706771	-1.039575	0.271860	
2013-01-05	-0.424972	0.567020	0.276232	-1.087401	
2013-01-06	-0.673690	0.113648	-1.478427	0.524988	

	A	B	C	D	E	F
0	1.0	2013-01-02	1.0	3	test	foo
1	1.0	2013-01-02	1.0	3	train	foo
2	1.0	2013-01-02	1.0	3	test	foo
3	1.0	2013-01-02	1.0	3	train	foo

One more?

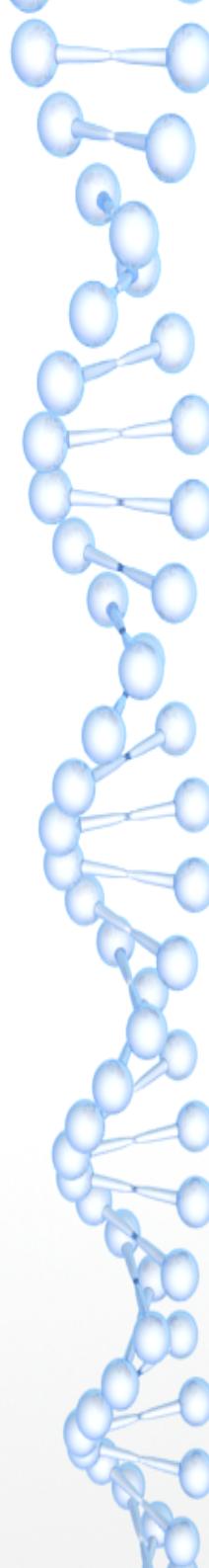
Seaborn is a Python visualization library based on matplotlib. It provides a high-level interface for drawing attractive statistical graphics.





Final recommendations.

- The practice is very important.
- Always read carefully the error messages.
- Do not despair!!!
- There are a lot of tutorials and help on the internet.
- Do not despair!!!
- Have fun!!!



Thank you!!!!