



# Scientific Analysis

Scientific Programming with Python

Christian Elsasser

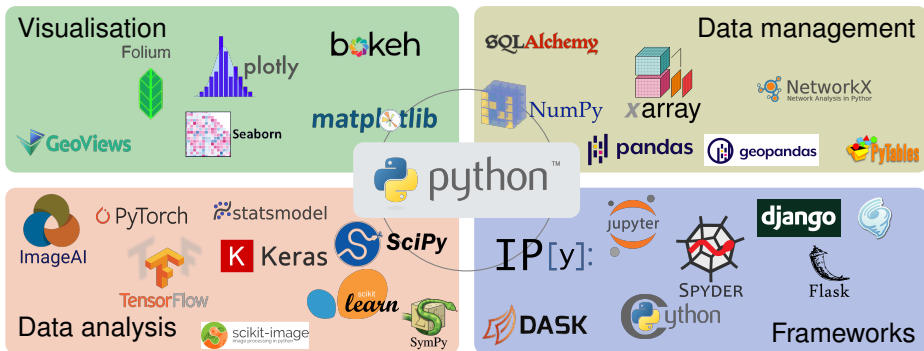


This work is licensed under the *Creative Commons Attribution-ShareAlike 3.0 License*.

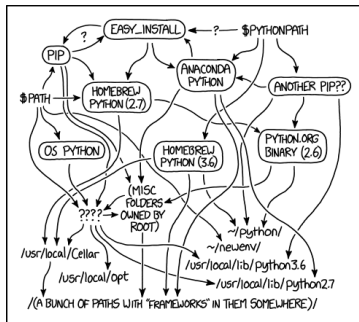


## Python offers a large ecosystem for scientific analytics and beyond

Domain specific modules



## We often treat modules like black boxes installed somehow on our machine



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED  
THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

[xkcd]

The goal of this session is to deep-dive into some of the fundamental functionalities



## Your Favourite Tools

*You are ...*

- ▶ **analysing geographical data**
  - ▶ geopandas
  - ▶ shaply
  - ▶ rasterio
- ▶ **doing Machine Learning**
  - ▶ scikit-learn
  - ▶ Keras, TensorFlow, PyTorch
  - ▶ ...
- ▶ **doing financial & economical modelling**
  - ▶ quantecon
  - ▶ statsmodels
- ▶ **dealing with images**
  - ▶ scikit-image
  - ▶ image AI

**It is pretty difficult to satisfy all wishes!!!**

⇒ Focus on **fundamental tools** (SciPy & NumPy) that are common to many areas!





## Table of Contents

**We focus on common challenges among the scientific disciplines:**

- ▶ Root-finding
- ▶ Optimisation
- ▶ Numerical integration & differentiation
- ▶ Linear Algebra
- ▶ Distributions

**You can find more details in the SciPy Lectures [here!](#)**



## SciPy – or Where the Fun Really Starts

- ▶ Offering a large number of functionality for numerical computation
  - ▶ `scipy.linalg` → Linear Algebra
  - ▶ `scipy.optimize` → Numerical optimisation (incl. least square)
  - ▶ `scipy.integrate` → Numerical integration
  - ▶ `scipy.stats` → Statistics including a large set of distributions
  - ▶ `scipy.spatial` → Spatial analysis like creation of Voroni sets, etc.
  - ▶ ...
  - ▶ more at <http://docs.scipy.org/doc/scipy/reference/>
- ▶ Eco-system of more advanced packages for data analysis, *e.g.*
  - ▶ `scikits.learn`: Machine-learning algorithms
  - ▶ `scikits.image`: Image processing
  - ▶ `pytables`: data structure (based on HDF5)
  - ▶ ...

**Remark:** `import scipy as sp` only imports the most basic tools ⇒ `from scipy import stats`

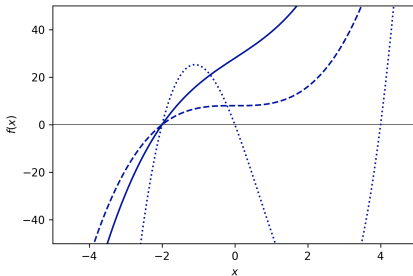
## Use case 1 – Root-finding in non-linear functions

### Problem:

- ▶ Finding roots of non-linear functions
- ▶ ... under sometimes non-trivial situations
- ▶ Fix point identification *i.e.* Find  $x$  such that  $x = f(x)$

### Goal:

- ▶ Understand what algorithms are available
- ▶ Understand their advantages and disadvantages as well as performance considerations



**Libraries discussed:** Optimisation (Root-finding part)



## Root-finding Algorithms

### Questions to ask:

- ▶ **Smooth** objective function?
- ▶ (Analytical) **derivatives** of first and second order available?
- ▶ Search **constraint** on a certain interval?
- ▶ Does a (or multiple) root **exist**?
- ▶ **Fix-point** formulation of the problem possible?

### Available algorithms:

- ▶ **Bracketing** (Bisection)
- ▶ **Quasi-Newton** (Secant)
- ▶ **Newton** (Newton)
- ▶ **Higher-order Householder** (Halley)
- ▶ **Hybrid** (Brent)





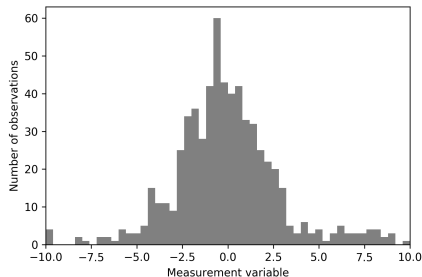
## Use case 2 – Maximum-likelihood estimation

### Problem:

- ▶ Parameter estimation of a distribution
- ▶ Evaluation of different models and if there are significant differences

### Goal:

- ▶ Understand available minimisation algorithms and their advantages and disadvantages
- ▶ Functionalities of distributions



**Libraries discussed:** Optimisation (Minimisation), Distributions



## Maximum-Likelihood Estimation

### Fundamentals:

- ▶ For a given sample of (observed) values  $x_i$  find the parameters  $\theta_j$  that are maximising the likelihood of the observation based on the distribution  $f(x|\theta)$

▶

$$\mathcal{L} = \prod_i f(x_i|\theta)$$

- ▶ Problem equivalent to minimise:

$$-\log \mathcal{L} = -\sum_i \log(f(x_i|\theta))$$

### Concrete case:

- ▶ Estimation of the daily returns by using a Gaussian distribution

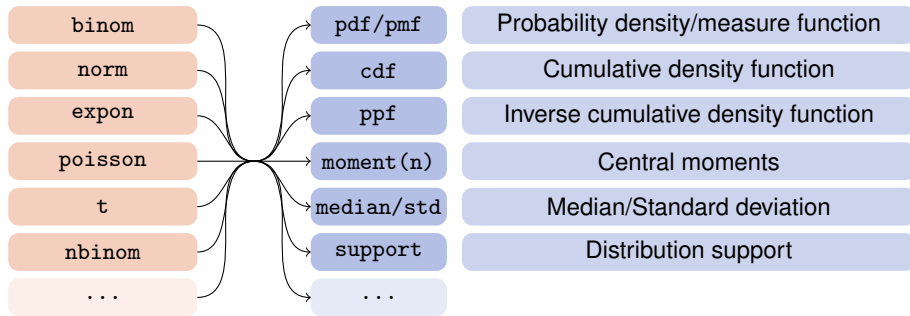
$$f(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- ▶ Single Gaussian case is trivial as the problem can be solved analytically with  $\hat{\mu} = \bar{x}$  and  $\hat{\sigma} = \sqrt{x^2 - \bar{x}^2}$



## Distributions and their functionality

The Scipy implementation of distributions offers a large range of distribution and statistical functionality





## Minimisation Algorithms

### Questions to ask:

- ▶ **Smooth** objective function?
- ▶ **Convex** objective function?
- ▶ Exact **Jacobian vector** or **Hessian matrix** available?
- ▶ **Bound** parameters?
- ▶ **Constraints** optimisation?

### Available algorithms:

- ▶ **Simplex** (Nelder-Mead)
- ▶ **Bi-directional** (Powell)
- ▶ **(Quasi-)Newton** (BFGS)
- ▶ **Trust-method** (Dogleg, Newton)

### Check documentation of

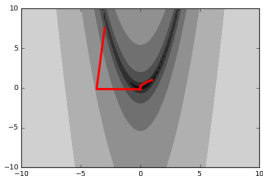
`scipy.optimize.minimize`

- ▶ **Choose the algorithm carefully based on your problem!**
- ▶ **A good conditioning (*i.e.* comparable scaling) is always beneficial**

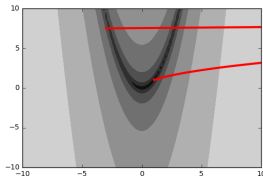
## Minimisation Algorithms – Differences

Comparison of different algorithms with the Rosenbrock function  
 $f(x, y) = (x - 1)^2 + 100(y - x^2)^2$  and starting point  $(-3, 7.5)$

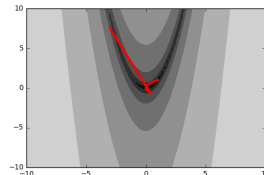
### Nelder-Mead



### BFGS



### Conjugate Gradient



Convergence heavily dependent on the choice of the algorithm and the initial starting point.

**More in the tutorial session!**



## Use case 3 – Linear Equation Solving

### Python's matrix handling:

- ▶ Users should rely on the standard `ndarray` – `np.matrix` is deprecated
- ▶ Idea is to have only one type like MATLAB
- ▶ ... but with opposite default (array and not matrix)
- ▶ Inverse and Hermitian now only functions and not any more properties, multiplication via `@` operator

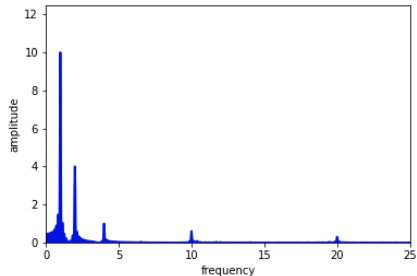
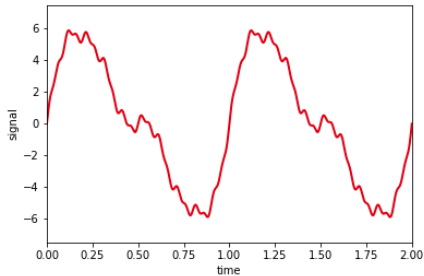
### Linear Algebra Calculus:

- ▶ Numpy offers a light version of SciPy's linear algebra implementation at `np.linalg`
- ▶ Full functionality in `scipy.linalg` like matrix exponential `scipy.linalg.expm`
- ▶ The functions are wrappers of the LAPACK linear algebra package

**Sparse matrices:** SciPy offers under `scipy.sparse` various types and flavours of sparse matrices including corresponding linear algebra calculus `scipy.sparse.linalg`



## Use case 4 – Signal Processing





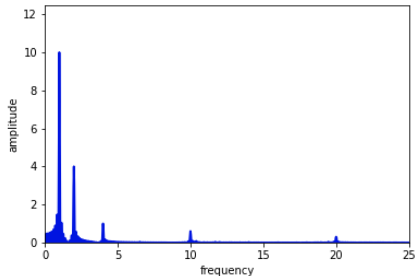
## Use case 4 – Signal Processing

### Problem:

- ▶ Spectrum determination of data or function
- ▶ Fast numerical integration

### Goal:

- ▶ Understand numerical integration and differentiation in SciPy



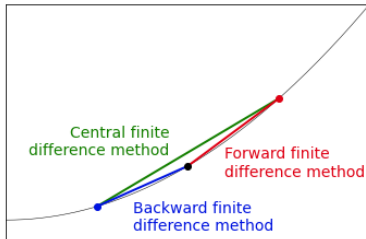
**Libraries discussed:** Differentiation, Integration, Fast-Fourier Transformation



## Numerical Differentiation

### Differentiation

- ▶ Implemented as **Central finite difference method**
- ▶ Using weighting tables based on “Generation of Finite Difference Formulas on Arbitrarily Spaced Grids” (Bengt 1988)





## Numerical Integration

### Integration – Newton-Cotes methods

- ▶ Estimate the integral based on a sample of values  $f(x_i)$  and  $x_i$ 
  - ▶ Trapezoidal rule
  - ▶ Simpson's rule
  - ▶ Romberg's rule
- ▶ Integral based on polynomial between the different points  $x_i$  (spline)

### Integration – Adaptive methods

- ▶ Quad methods based on Gauss–Kronrod quadrature
- ▶ Adaptive distance between evaluation points and able to dealing with “singularities”
- ▶ Based the Fortran library QUADPACK
- ▶ Sample of methods for particular situations *e.g.* to have a weight function  $w$  *i.e.*

$$I = \int_a^b dx f(x) w(x)$$



## Fourier Transformation

### Problem to solve:

- ▶ Calculate for a given function  $f(t)$  and frequency  $\omega$  the amplitude

$$A(\omega) = \int_{-\infty}^{\infty} dt e^{-i\omega t} f(t)$$

- ▶ Depending on the convention you might have an additional factor  $(2\pi)^{-1/2}$ .
- ▶ Idea: Evaluate the above integral numerically.

### Strategy to solve it in Python:

1. Run the integration with the `quad` method
2. Use `np.vectorize` to evaluate the integral in parallel for different  $\omega$  values



## Advanced Python Modules

We omitted any modules with a large and specific purpose → otherwise you would sit here tomorrow

Left to the interested audience to explore them further

- ▶ NLTK ([www.nltk.org](http://www.nltk.org)) → Natural language processing
- ▶ scikit-learn ([scikit-learn.org](http://scikit-learn.org)) → Machine learning
- ▶ scikit-image ([scikit-image.org](http://scikit-image.org)) → Image processing and analysis
- ▶ ...

Rapidly growing and improving landscape of python modules, but with still some “whitish” spots (e.g. time series) ⇒ Reflection of available alternatives?



## Conclusion

- ▶ SciPy together with NumPy offers a large number of fundamental tools for your everyday work in science and beyond . . .
- ▶ . . . and they let you built your own tools for research.
- ▶ Understanding these fundamental libraries is also helpful to understand the “under the hud” part of more specialised libraries.
- ▶ Take the time to understand the content of the package . . .
- ▶ . . . to avoid a reinvention of the wheel

Other relevant (fundamental) libraries will be discussed on Friday by Jonas together with the topic of visualisation.