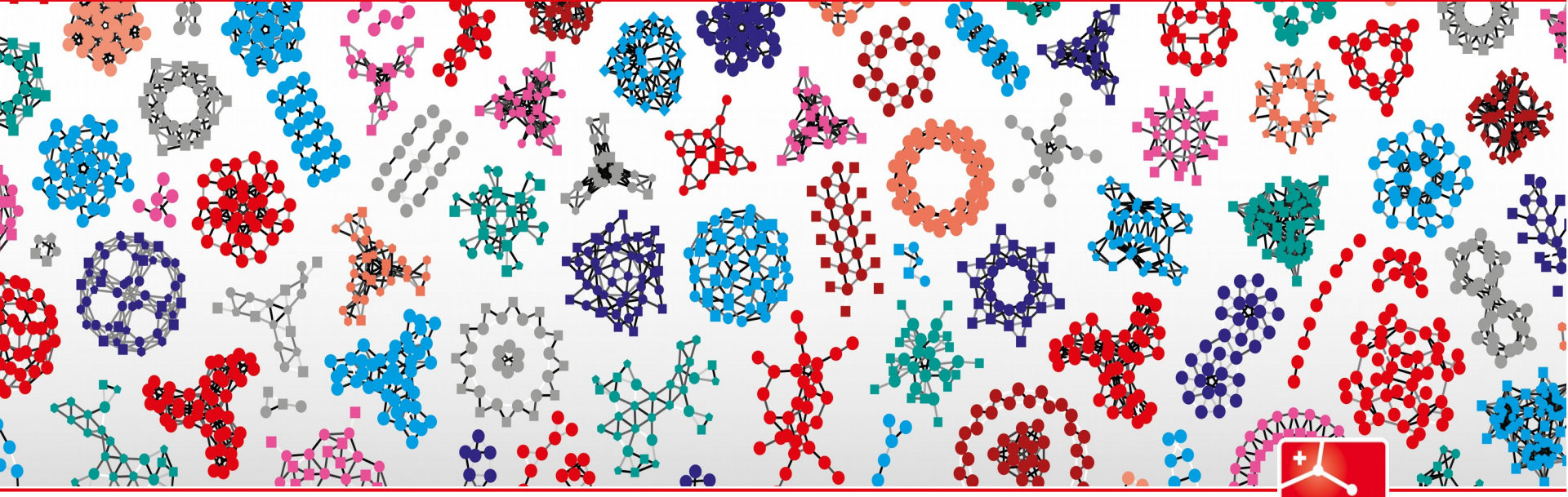


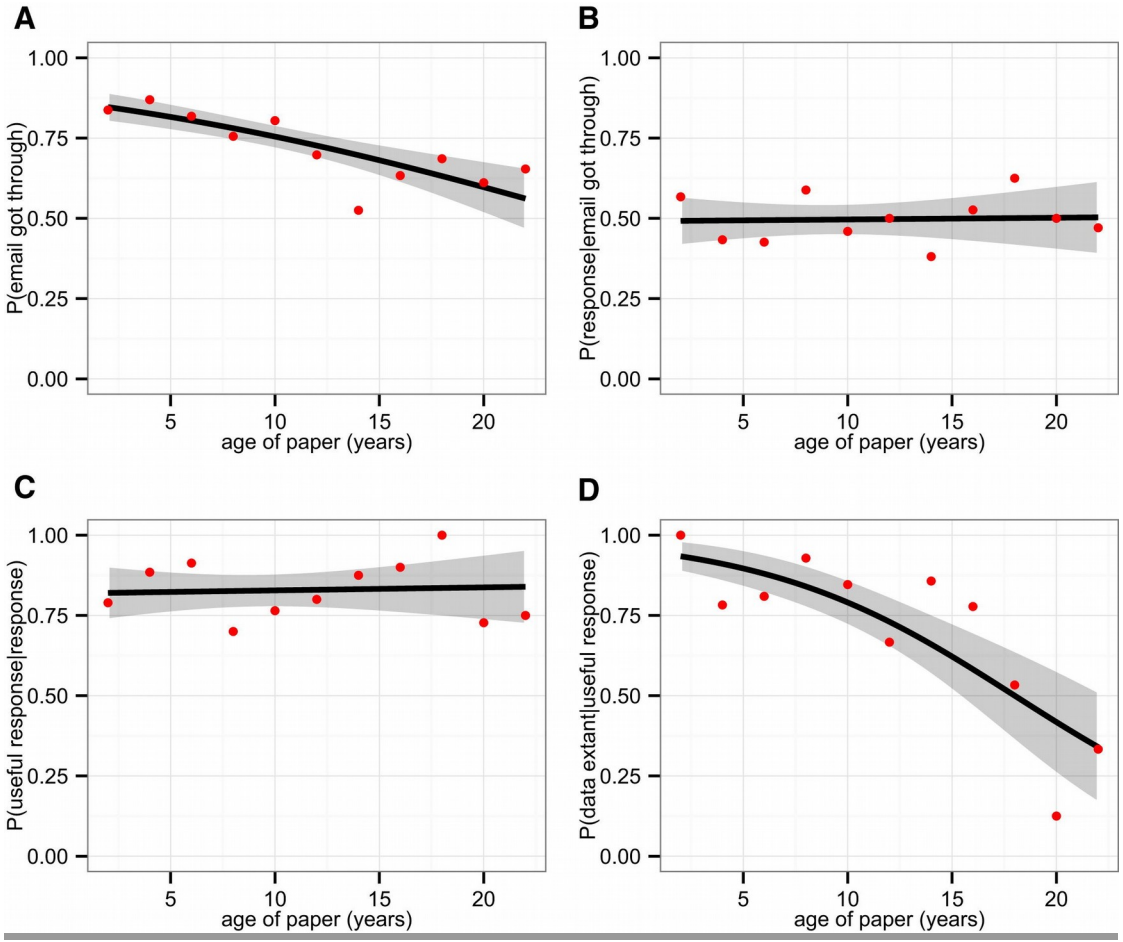
First steps with Python in life sciences

Wandrille Duchemin
Robin Engler
Orlin Topalov

Swiss Institute of Bioinformatics

elixir SWITZERLAND

The reproducibility crisis



Survey of 516 studies:

- - 17% data availability per year.
- Only 19% retrieval rate after 10 year...

Vines et al. (2014) Curr. Biol. doi.org/10.1016/j.cub.2013.11.014

The **FAIR** guiding principles

Findable: Metadata and data should be easy to find for both humans and computers (unique global identifier, rich description, machine readable and searchable).

Accessible: The data can be retrieved using a standard communication protocol (e.g. https or sftp). Where needed, authentication and authorization procedures are available and documented.

Interoperable: the (meta)data should be based on standardized vocabulary and ontologies (categories and their relations), so that it integrates with existing applications and workflows.

Reusable: Metadata and data should be well described so that data can be replicated and/or combined in different research settings (rich metadata, clear license term, origin of data, data meets domain-relevant community standards).

M.D.Wilkinson, et al. The FAIR Guiding Principles for scientific data management and stewardship. Sci Data. 2016;3:160018. doi:10.1038/sdata.2016.18. https://www.go-fair.org

FAIR applied to code

```
def f(x, y):
    if x==[]: return y
    if y==[]: return x
    return [x[0]] + f(x[1:], y) if x[0]<y[0] else [y[0]] + f(x, y[1:])

def S(a, n):
    m = n//2
    return a if n<=1 else f(S(a[:m], m), S(a[m:], n-m))

```

← Don’t do this

FAIR applied to code

```python
def f(x, y):
    if x==[]: return y
    if y==[]: return x
    return [x[0]] + f(x[1:], y) if x[0]<y[0] else [y[0]] + f(x, y[1:])

def S(a, n):
    m = n//2
    return a if n<=1 else f(S(a[:m], m), S(a[m:
```

← Don’t do this

Do that →

```python
def merge(x, y):
    """
    *RECURSIVE FUNCTION*
        Takes 2 sorted lists and
        returns the sorted list merging them.
    """
    ## if one of the 2 lists are empty, just return the other one
    if x==[]:
        return y
    if y==[]:
        return x

    # x and y are sorted
    if x[0]<y[0] :
        # lowest element of x goes first
        #    --> merged list is first element of x + merging of y and the remainder of x
        return [x[0]] + merge(x[1:], y)
    else:
        # lowest element of y goes first
        # same logic as above
        return [y[0]] + merge(x, y[1:])


def sort(a):
    ''' *RECURSIVE FUNCTION*
    Takes a list <a>, and returns a sorted version.
    Uses the merge sort algorithm.
    Implementation freely inspired from: http://www.rosettacode.org/wiki/Sorting_algorithms/Merge_sort#Python
    '''
    n = len(a)
    m = n//2

    if n<=1: # list empty or with a single element --> return as is
        return a
    else: ## split the list in two. Sort separately and then merge the 2 sorted halves
        return merge(sort(a[:m]), sort(a[m:]))
```

FAIR applied to code

- Write code that also acts as documentation, and clearly communicates the analysis.
- Apply the standard you would expect of a ‘wet-lab’ protocol.
- Will a reasonably competent colleague understand your code ?
- Will you understand your code in 6 month ??

To achieve this, you should:

- Comment as much as needed.
- Use explicit names when naming things (variables, functions, classes).
- Possibly use a support that allows to easily mix code and text, e.g. Jupyter notebook or Jupyter-lab.
- Also, you can look at:

  Schwen LO, Rueschenbaum S (2018) Ten quick tips for getting the most scientific value out of numerical data. PLoS Comput Biol 14(10): e1006141. https://doi.org/10.1371/journal.pcbi.1006141.

How to execute python code

Three main modes of interaction:

- Interactive console.
- Python code file (.py files).
- Jupyter Notebook (.ipynb files).

How to execute python code

Three main modes of interaction:

- Interactive console.
  - Nice for quick debugging, testing syntax,...
- Python code file (.py files).
- Jupyter Notebook (.ipynb files).

How to execute python code

Three main modes of interaction:

- Interactive console.
- Python code file (.py files).
  - Scripts, program, ...
  - most used form
- Jupyter Notebook (.ipynb files).

How to execute python code

Three main modes of interaction:

- Interactive console.
- Python code file (.py files).
- Jupyter Notebook (.ipynb files).
  - Great for data analysis and teaching

Python – using the console

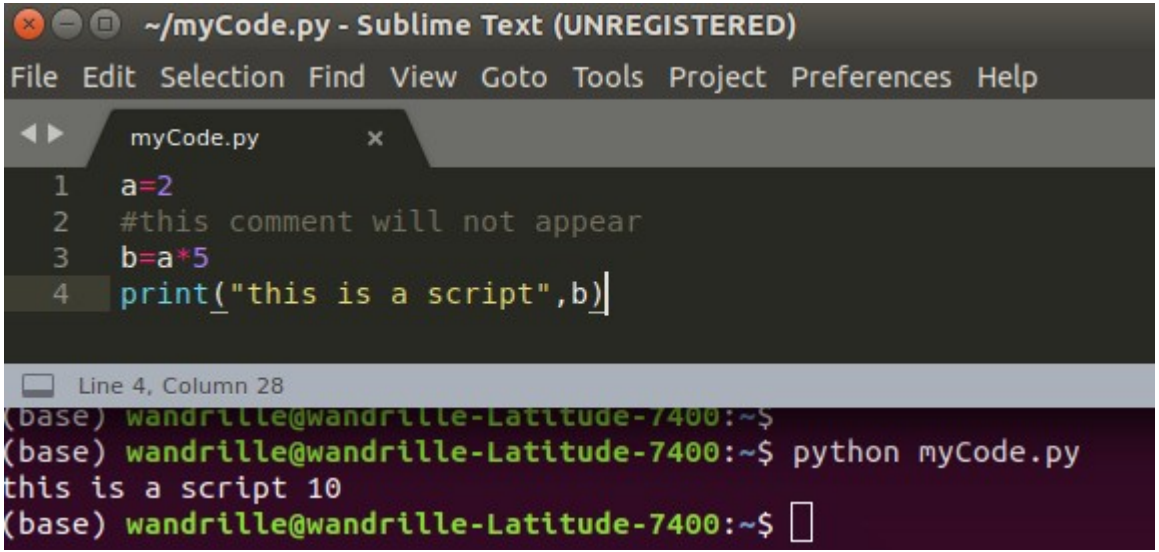
```
(base) wandrille@wandrille-Latitude-7400:~$ python
Python 3.7.4 (default, Aug 13 2019, 20:35:49)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> a=3
>>> b=4
>>> a*b + 7
19
>>> print("this is an interactive console")
this is an interactive console
>>>
```

- **Interactive** : the code is executed as you press ‘Enter’
- Great for quickly **testing** things out.
- But, you keep **no trace** of your workflow/environment.

Only use it to test little bits of code

Python – writing a code in a .py file

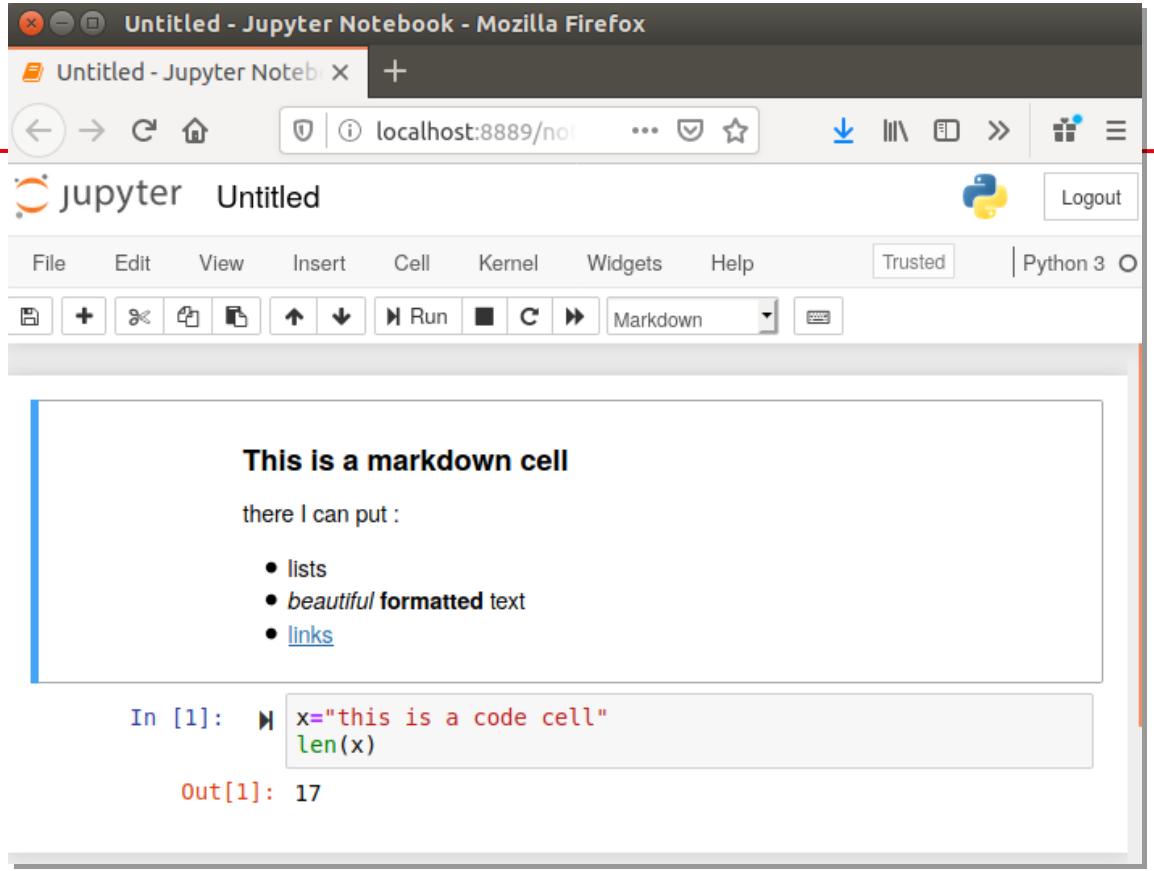
- Write a script in a .py text file, then execute it.
- Main way python code is shared.
- Ideal for standalone programs and modules.
- Code and results are kept separate (may be a good or a bad thing).



→ Good for general purpose coding, “operational script”.

Jupyter notebook / Jupyter Lab

- Browser based interface (but runs locally on your machine).
- Interlace Markdown and code ‘cells’.
- Execute code cell by cell.
- **Commentary, code, and results together in the same file**.
- Visually pleasant and fairly ergonomic.



The ability to have documentation, code and results in a single file makes it ideal for **data analysis**: helps reproducibility of results.

Jupyter notebook / Jupyter Lab – magic commands

- Jupyter add a bunch of magic commands on top of the python syntax
- Start with % (applies to line) or %% (applies to whole cell)

- A lot of very useful recipes:
  - %%time : report execution time of a cell
  - %%bash : run this cell with bash instead of python
  - %%writefile : write cell content to a file

    (eg. to export some code to a script)

https://ipython.readthedocs.io/en/stable/interactive/magics.html