Reproducible Science with Jupyter

Carsten Fortmann-Grote
Scientific Computing Services
Max-Planck-Institut for Evolutionary Biology, Plön, Germany
carsten.fortmann-grote@evolbio.mpg.de

Hans Fangohr
Scientific Support Unit for Computational Science
Max-Planck Institute for Structure and Dynamics of Matter, Hamburg, Germany
University of Southampton, Southampton, United Kingdom
@ProfCompMod, hans.fangohr@mpsd.mpg.de

MAX PLANCK INSTITUTE
FOR EVOLUTIONARY BIOLOGY

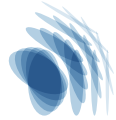
UNIVERSITY OF
Southampton

Research Data Management workshop 5 and 6 May 2021

Outline

- Brief introduction Jupyter Notebook
- Some use cases
- Jupyter Notebooks for (more) reproducible science
- Binder, and example

- Format:
  - Presentation
  - Tutorial
  - Discussion
- Informal: Ask questions immediately

Jupyter Notebook

- Purpose: data analysis, exploration, visualisation, …
- Document hosted in web browser (demo)
- Combines
  - text (markdown with LaTeX support)
  - Computer code (Python)
  - Output from code
- Saved in one document
  - `*.ipynb` [IPYthon NoteBook] (JSON format on disk)
  - Can be re-loaded and re-executed

http://github.com/fangohr/jupyter-demo

In [1]:
```
# Import Python and libraries we need later
%matplotlib inline
from numpy import exp, cos, linspace
import pylab
from ipywidgets import interact
```

Mathematical model: We would like to understand $f(t, \alpha, \omega) = \exp(-\alpha t)\cos(\omega t)$

Code: Here is an implementation:

In [2]:
```
def f(t, alpha, omega):
    """Computes and returns exp(-alpha*t) * cos(omega*t)"""
    return exp(-alpha * t) * cos(omega * t)
```

Interactive exploration: We can execute the function for values of t, α and ω:

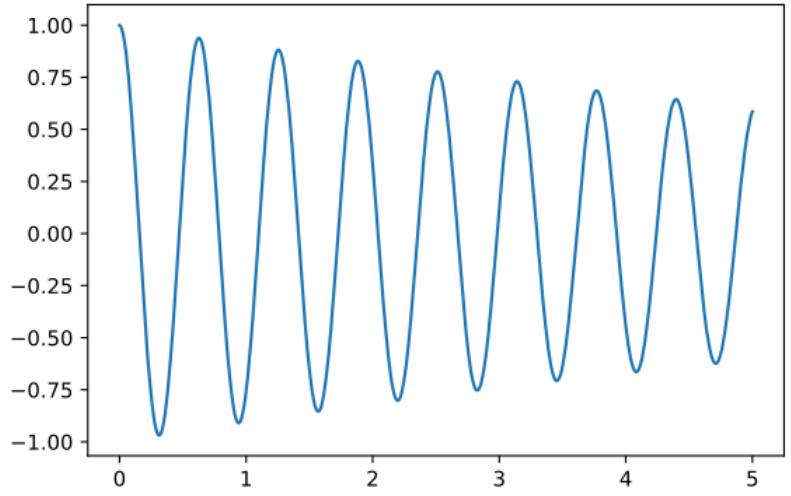
In [3]:
```
f(t=0.1, alpha=1, omega=10)
```

Out[3]:
```
0.48888574340060287
```

Or produce a plot (in a function `plot_f` so it can be re-used for different parameters):

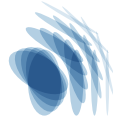
In [4]:
```
def plot_f(alpha, omega):
    ts = linspace(0, 5, 500)     # 500 points in interval [0, 5]
    ys = f(ts, alpha, omega)
    pylab.plot(ts, ys, '-')
```

In [5]:
```
plot_f(alpha=0.1, omega=10)     # call function and create plot
```

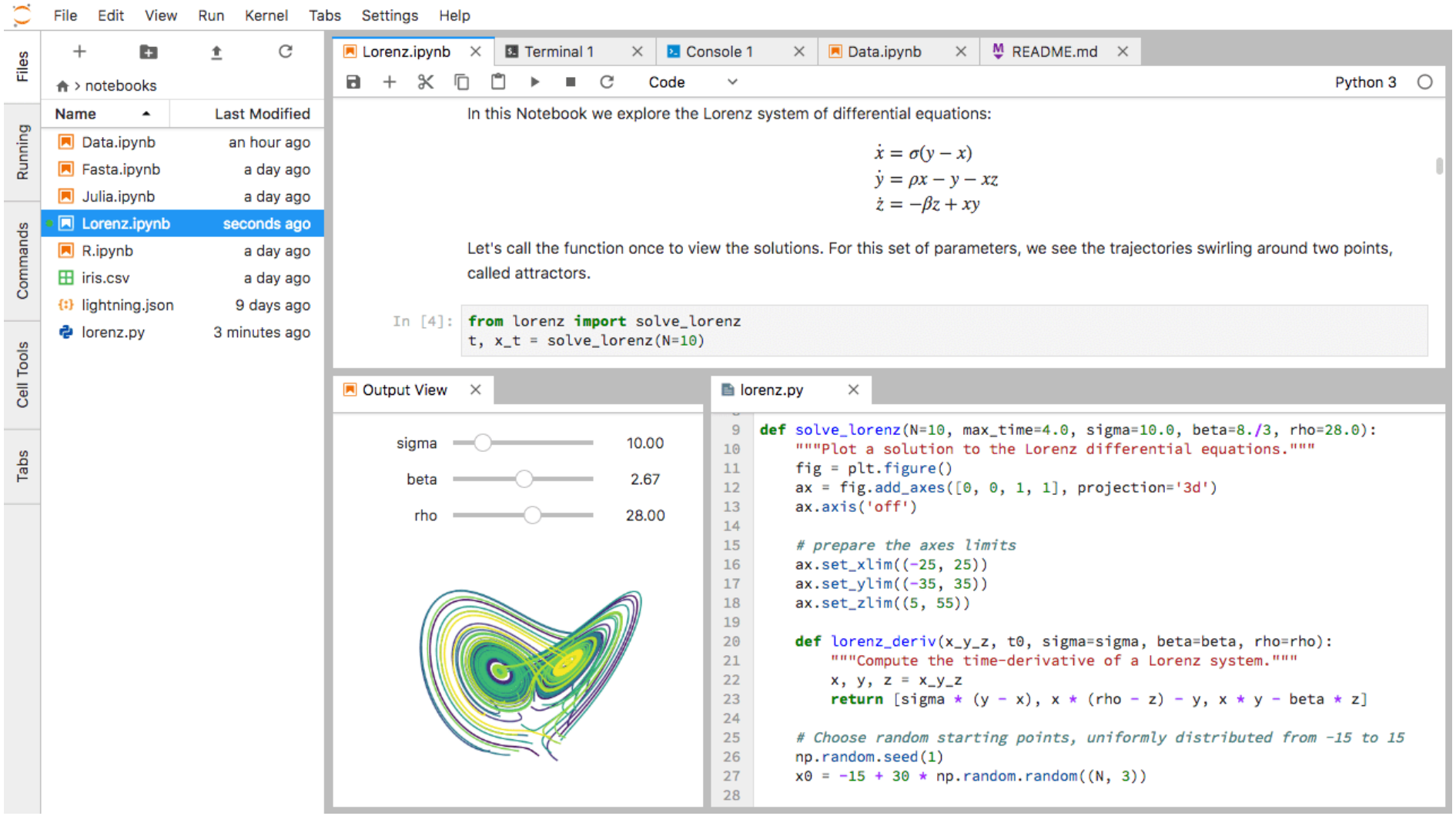


Terminology project Jupyter

- Jupyter Notebook: document in browser / file format
- Jupyter Lab - newer interface in browser
  (includes Jupyter Notebook)
- JupyterHub:
  allows to use Jupyter notebooks remotely

JupyterLab


File Edit View Run Kernel Tabs Settings Help
Files
Running
Commands
Cell Tools
Tabs
notebooks
Name
Last Modified
Data.ipynb an hour ago
Fasta.ipynb a day ago
Julia.ipynb a day ago
Lorenz.ipynb seconds ago
R.ipynb a day ago
iris.csv a day ago
lightning.json 9 days ago
lorenz.py 3 minutes ago
Lorenz.ipynb
Terminal 1
Console 1
Data.ipynb
README.md
Code
Python 3
In this Notebook we explore the Lorenz system of differential equations:
$\dot{x} = \sigma(y - x)$
$\dot{y} = \rho x - y - xz$
$\dot{z} = -\beta z + xy$
Let's call the function once to view the solutions. For this set of parameters, we see the trajectories swirling around two points, called attractors.
In [4]:
from lorenz import solve_lorenz
t, x_t = solve_lorenz(N=10)
Output View
sigma 10.00
beta 2.67
rho 28.00
lorenz.py
def solve_lorenz(N=10, max_time=4.0, sigma=10.0, beta=8./3, rho=28.0):
"""Plot a solution to the Lorenz differential equations."""
fig = plt.figure()
ax = fig.add_axes([0, 0, 1, 1], projection='3d')
ax.axis('off')
# prepare the axes limits
ax.set_xlim((-25, 25))
ax.set_ylim((-35, 35))
ax.set_zlim((5, 55))
def lorenz_deriv(x_y_z, t0, sigma=sigma, beta=beta, rho=rho):
"""Compute the time-derivative of a Lorenz system."""
x, y, z = x_y_z
return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]
# Choose random starting points, uniformly distributed from -15 to 15
np.random.seed(1)
x0 = -15 + 30 * np.random.random((N, 3))

History of Jupyter

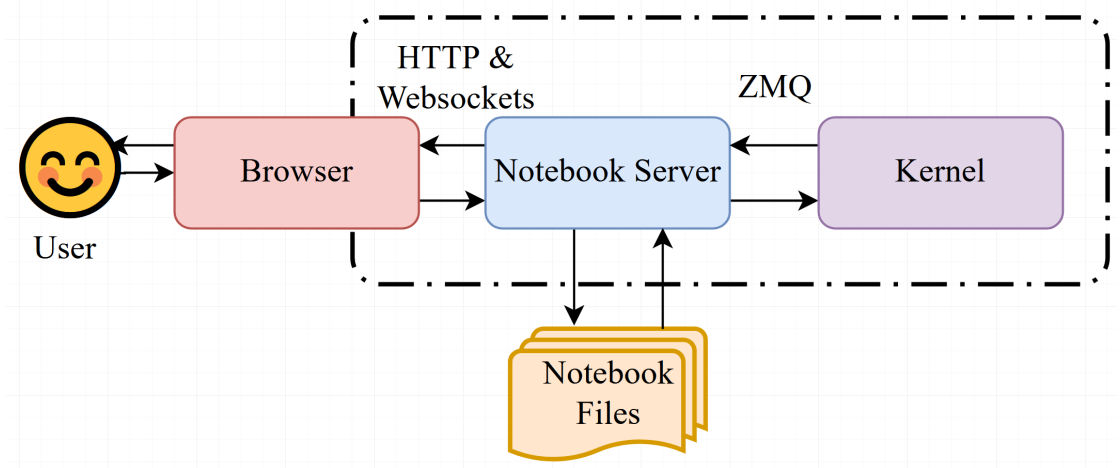
- IPython -> IPYthon NoteBook -> `*.ipynb`
- community began adding other languages (“kernels”) to the notebooks, starting with Julia and R
- JUlia, PYThon, and R -> Jupyter
- “Jupyter” is also a homage to Galileo’s notebooks which recorded the discovery of Jupiter’s moons

IP[y]: IPython

IP[y]: Notebook

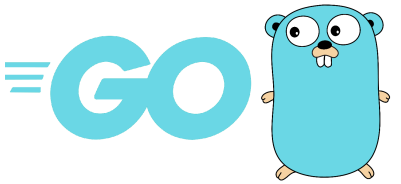
jupyter

* How does it work?

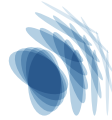


- Starting jupyter-notebook starts the notebook server
- Opening a notebook starts up the “kernel”
- The kernel communicates with the notebook server via ZMQ
- The notebook server shows content via the browser
- JavaScript used in Browser

* Supported Kernels



- 50+ languages supported.
- More complete list at
- https://github.com/jupyter/jupyter/wiki/Jupyter-kernels



Use case: data analysis

- Explorative data analysis
- Convenient combination of processing, results and interpretation
- Complete capture of all computational steps
- Good record for reproducibility
- Share with collaborators & supervisors (html, latex, pdf)

X-ray Absorption Spectroscopy

Step 1: Load data and align them by train id and pulse id

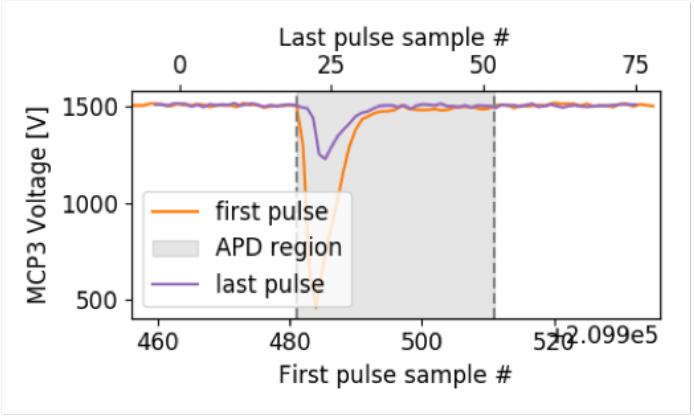
```
In [4]: proposalNB = 900074
        semesterNB = 201930
        runNB = 487
        topic = 'SCS'
        fields = ["SCS_photonFlux", "SCS_XGM", "MCP3apd", "nrj"]
        run = tb.load(fields, runNB, proposalNB, semesterNB, topic,
                      validate=True, display=False)
        nrun = tb.matchXgmTimPulseId(run)
```

```
Checking run directory: /gpfs/exfel/exp/SCS/201930/p900074/raw
/r0487/
No problems found
```

Step 2: check the pulse integration window

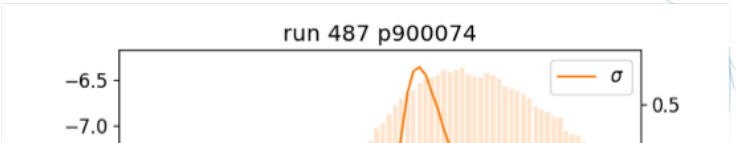
```
In [5]: tb.checkTimApdWindow(nrun, mcp=3)
```

```
no raw data for MCP3. Loading trace from MCP3
```



Step 3: bin the data and plot the XAS spectrum

```
In [6]: nrj = np.linspace(nrun.nrj.min(), nrun.nrj.max(), 80)
        xas = tb.xas(nrun, nrj, plot=True)
```



Use case: recipes

- For recurring tasks, pre-populate notebook with cells to carry out a particular type of data analysis
- Create collection of such notebook recipes
- Compromise between static recipe (=script) and interactive exploration

X-ray Absorption Spectroscopy

Step 1: Load data and align them by train id and pulse id

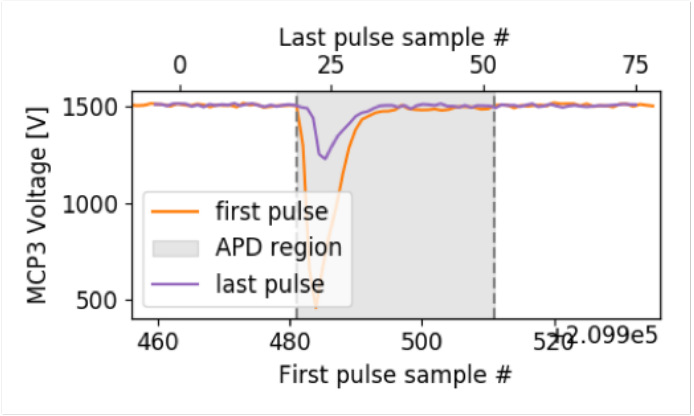
```
In [4]: proposalNB = 900074
        semesterNB = 201930
        runNB = 487
        topic = 'SCS'
        fields = ["SCS_photonFlux", "SCS_XGM", "MCP3apd", "nrj"]
        run = tb.load(fields, runNB, proposalNB, semesterNB, topic,
                      validate=True, display=False)
        nrun = tb.matchXgmTimPulseId(run)
```

```
Checking run directory: /gpfs/exfel/exp/SCS/201930/p900074/raw
/r0487/
No problems found
```

Step 2: check the pulse integration window

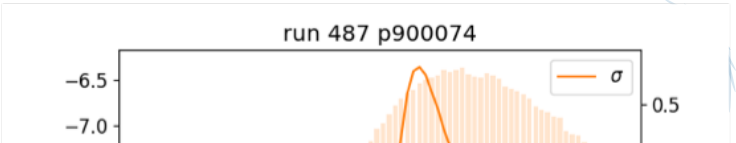
```
In [5]: tb.checkTimApdWindow(nrun, mcp=3)
```

```
no raw data for MCP3. Loading trace from MCP3
```



Step 3: bin the data and plot the XAS spectrum

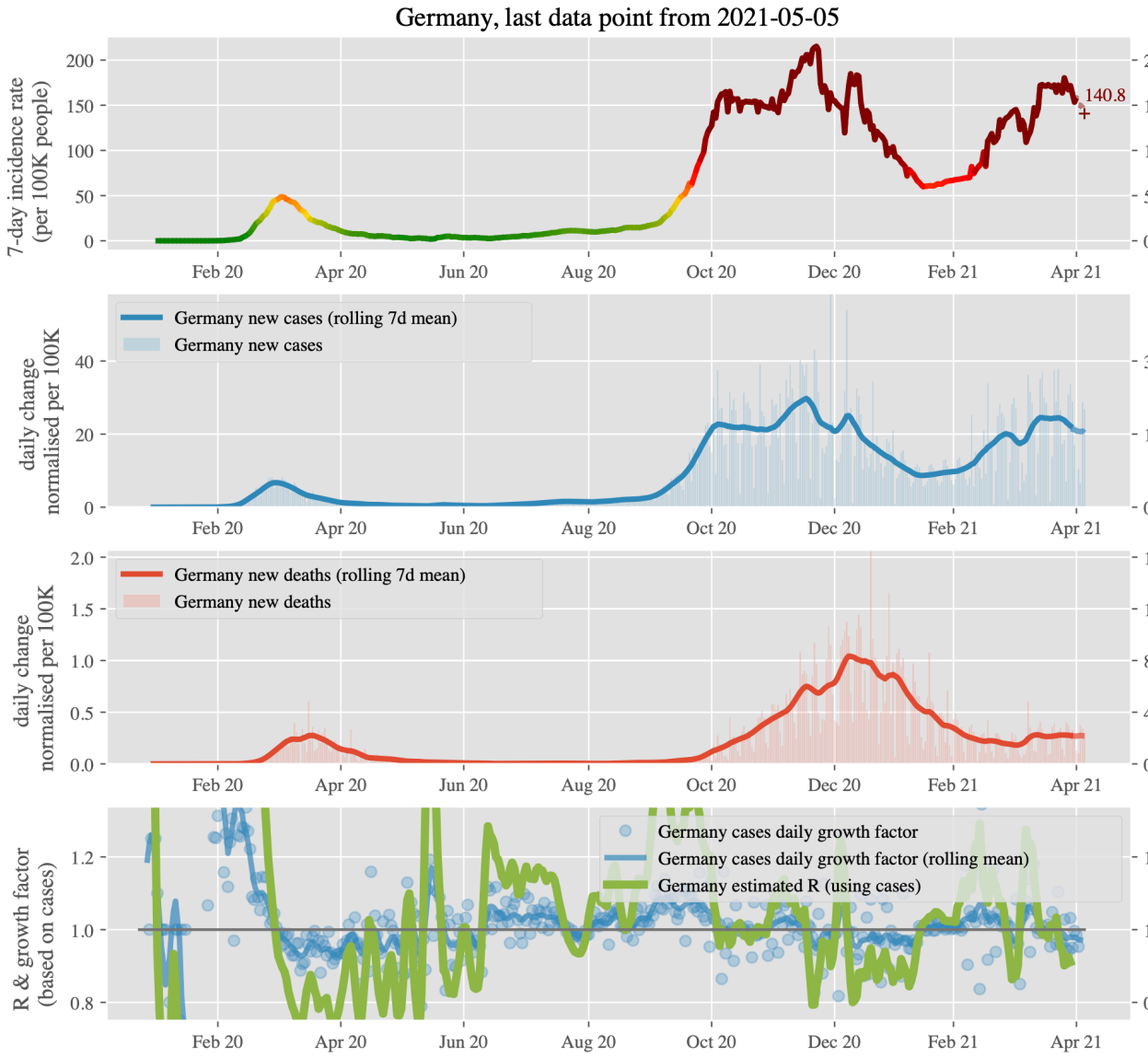
```
In [6]: nrj = np.linspace(nrun.nrj.min(), nrun.nrj.max(), 80)
        xas = tb.xas(nrun, nrj, plot=True)
```



Use case: report generator

- Use Jupyter Notebook as a report generator
  - *execute using **`nbconvert`** & save resulting notebook or create html/pdf
  - *Can modify parameters in beginning of notebook (**`nbparametrize`** or **`papermill`**)
  - Error messages embedded in output
- Use cases:
  - Open Science COVID Analysis https://oscovida.github.io https://oscovida.github.io/html/Germany.html

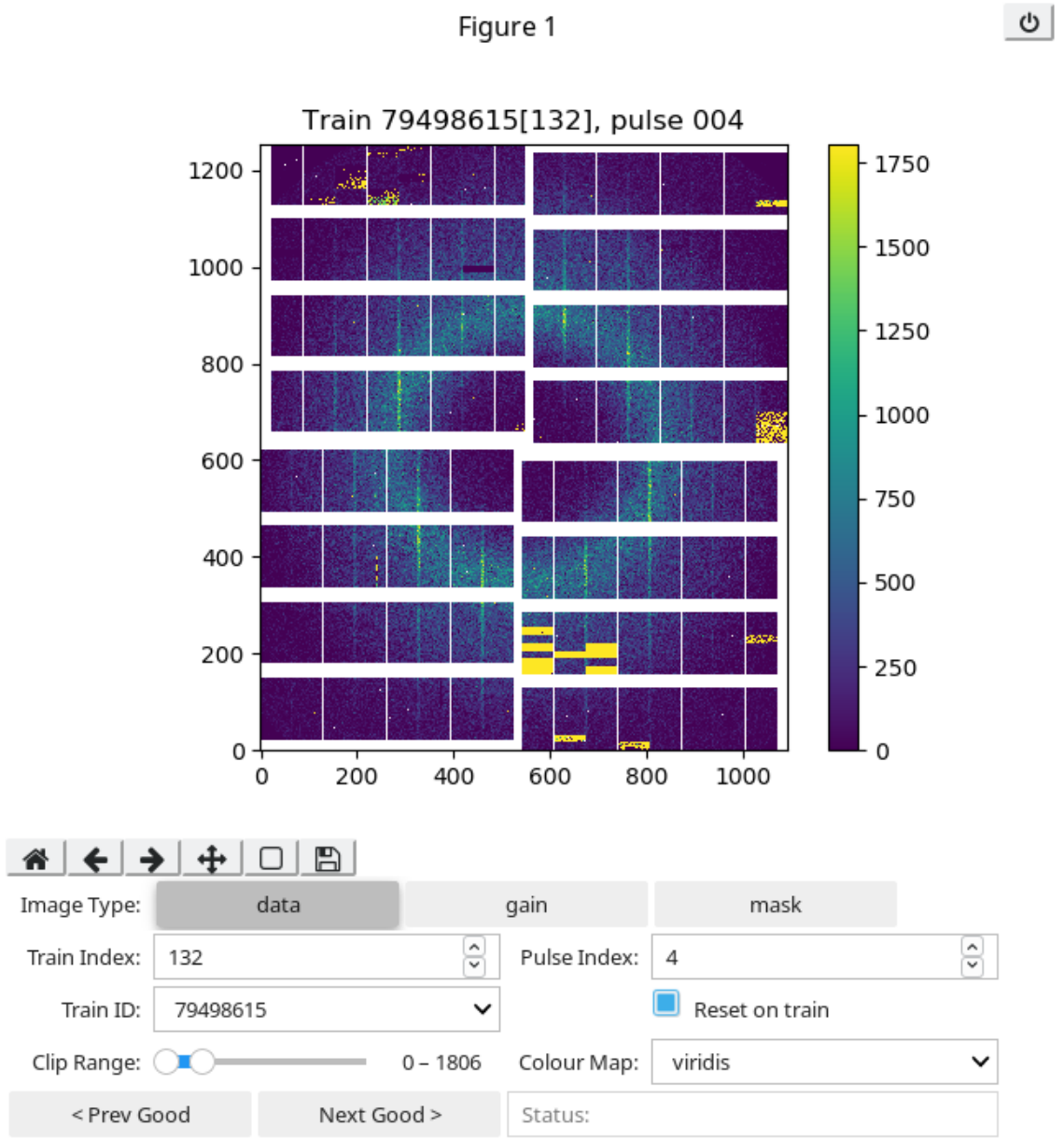
```
[4]: overview("Germany");
```



Use case: blending GUI and script

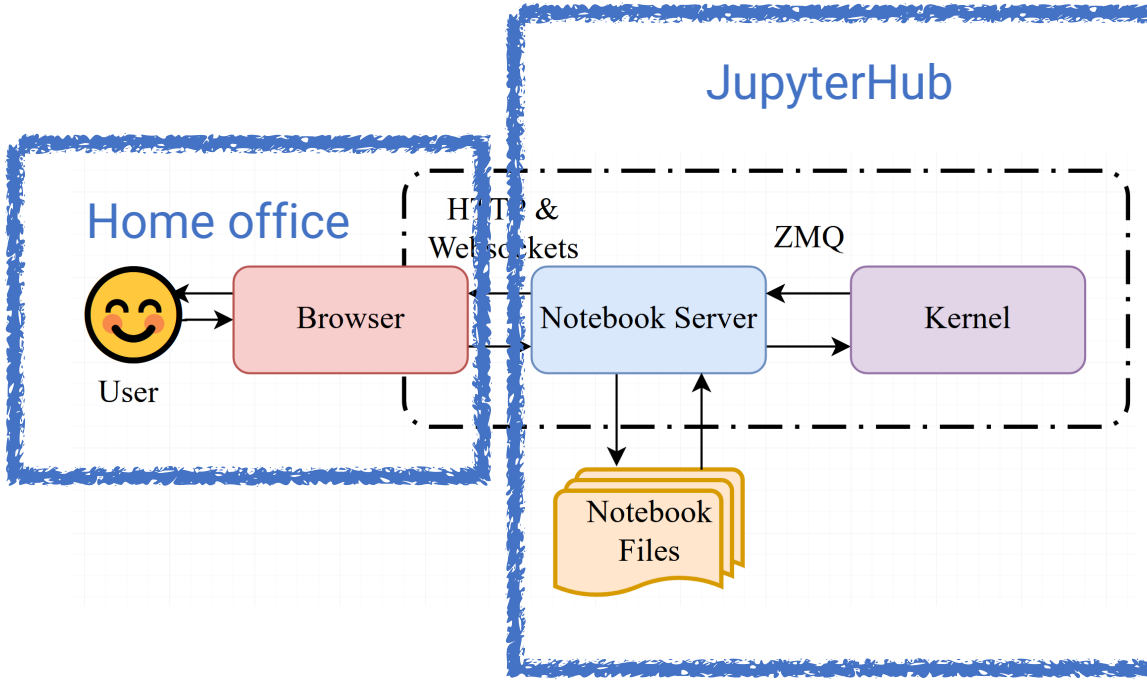
- "Widgets" provide graphical control elements in notebooks
- Buttons, sliders etc trigger code execution / plotting
- Discussion
  - Reproducibility is difficult at the moment

```
[6]: demo = InteractiveGeom(geom, run)
[7]: demo.interactive()
```



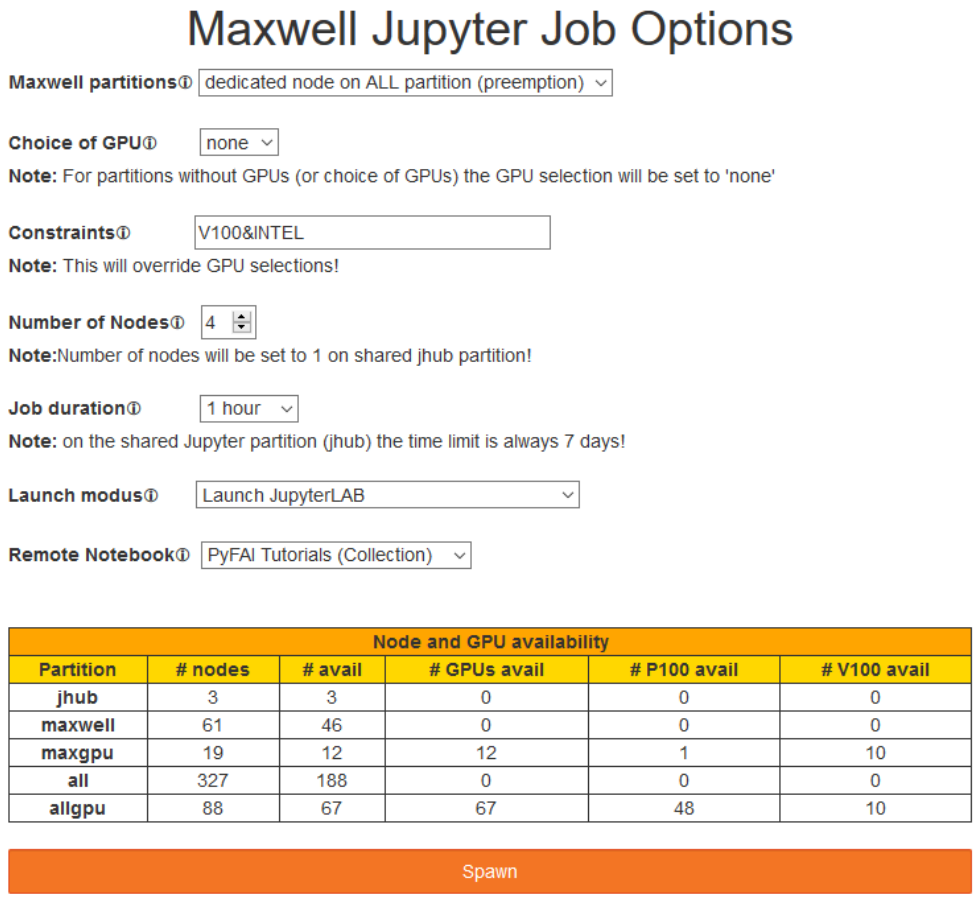
*JupyterHub: use remote compute environment

- JupyterHub allows
  - users to connect through browser to High Performance Computing (HPC) resources
  - Use storage of HPC systems for notebooks and data
- Example: JupyterHub @ GWDG: https://jupyter-cloud.gwdg.de



JupyterHub example EuXFEL / DESY

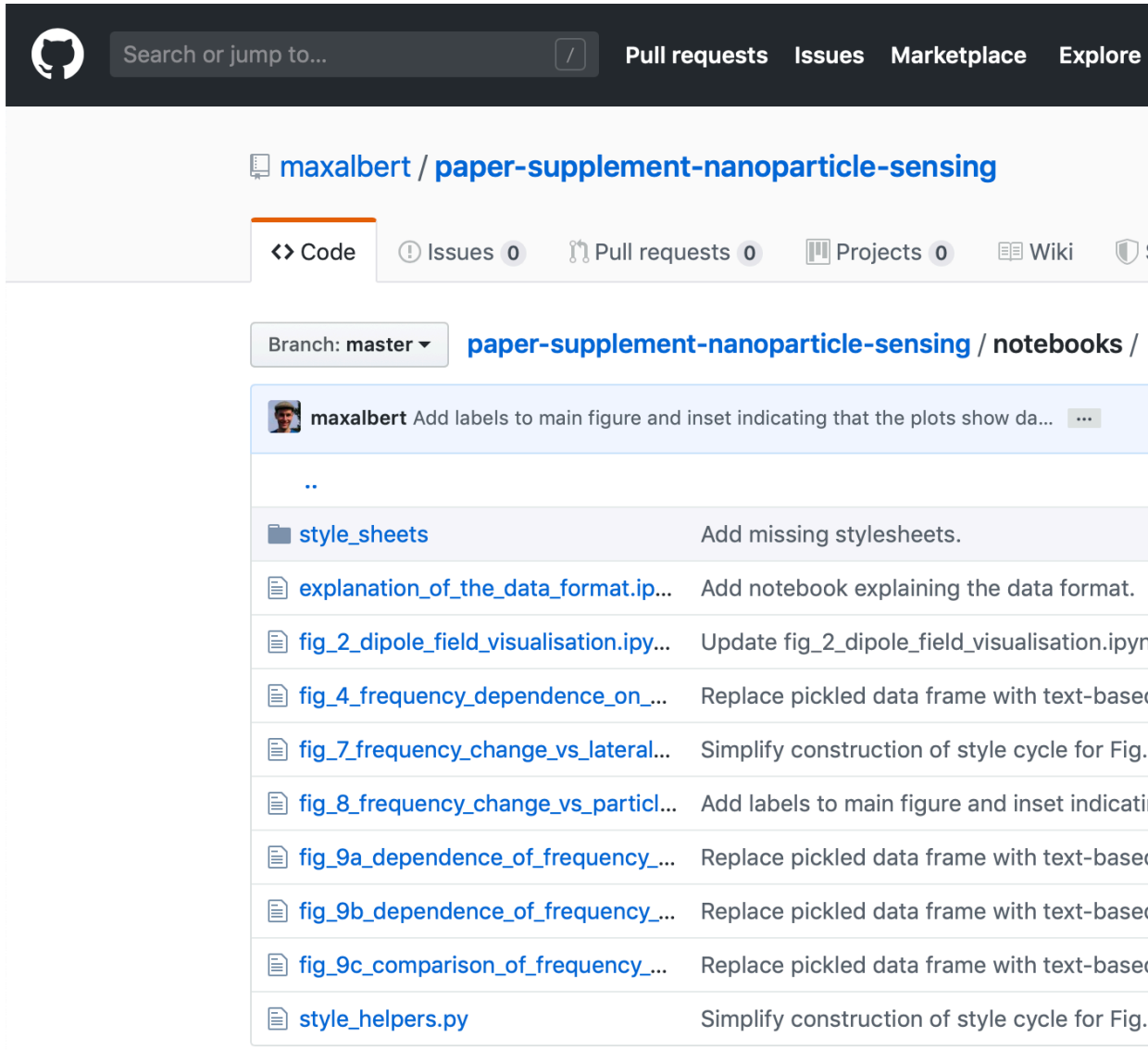
- Access to ~30PB of data
- Many thousand compute cores
- Select appropriate hardware before launching



Data exploration and analysis with Jupyter notebooks, Proceedings of the 17th International Conference on Accelerator and Large Experimental Physics Control Systems ICALEPCS2019, TUCPR02 http://accelconf.web.cern.ch/icalepcs2019/doi/JACoW-ICALEPCS2019-TUCPR02.html, pdf (2020)

Use case: reproducible publication

- Create github repository to complement publication, containing
- one notebook per figure / main result
- Zenodo for long term preservation



Example:
https://github.com/maxalbert/paper-supplement-nanoparticle-sensing

Nanotechnology **27** (2016) 455502

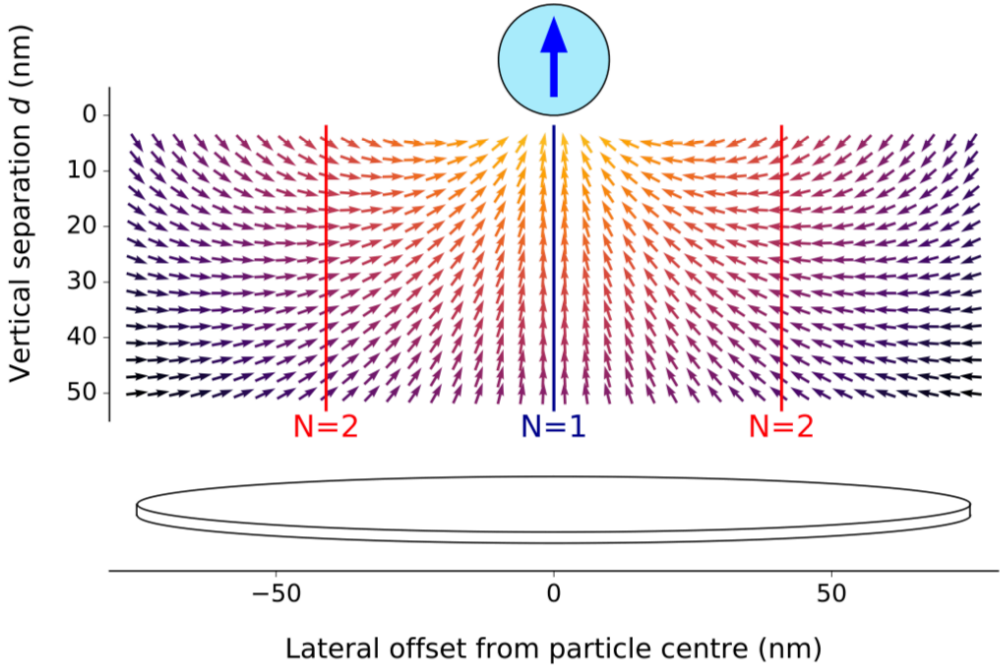
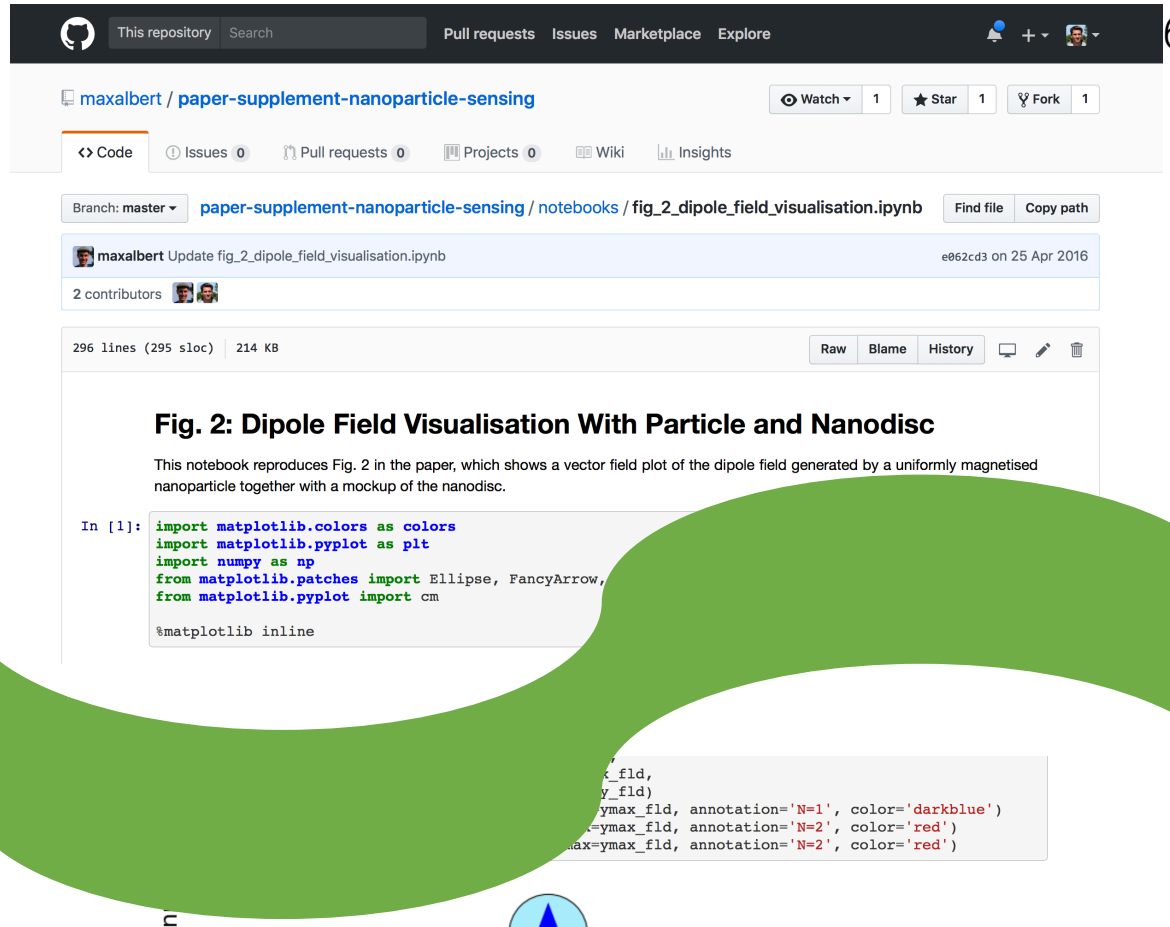
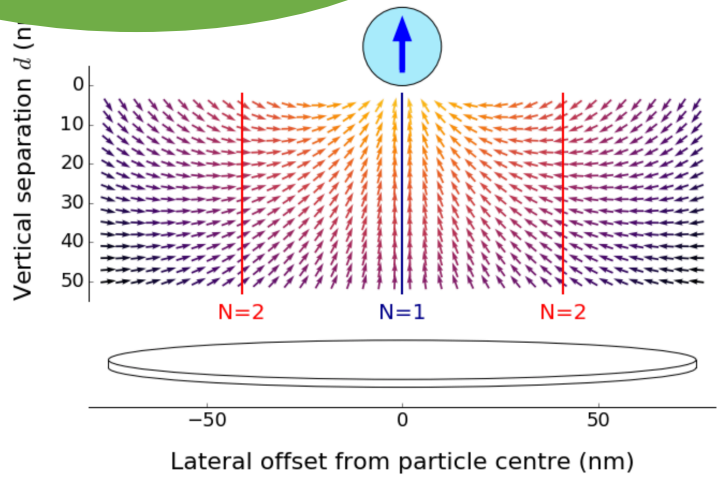


Figure 2. Vector field plot of the dipole field generated by a uniformly $+z$-magnetised MNP. The vectors are scaled to uniform length with their colour indicating the field strength (orange is high and violet/black is low). The vertical lines correspond to the x-values where modes 1 and 2 have maxima in their spin precession amplitude (see figures 3(a)–(b)). A schematic of the nanodisc is shown at the bottom.

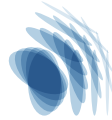




[Maximilian Albert et al 2016 Nanotechnology 27 455502]

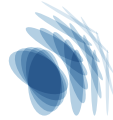
One-study one-document

- *One-study one-document*:
  Notebook combines (and logs) equations, code, results, interpretation
- Much easier than manually tracking post-processing, analysis, plot-creation scripts, figure files, and LaTeX / Word files.
- *Data exploration and analysis with Jupyter notebooks,* 10.18429/JACoW-ICALEPCS2019-TUCPR02 (2020)



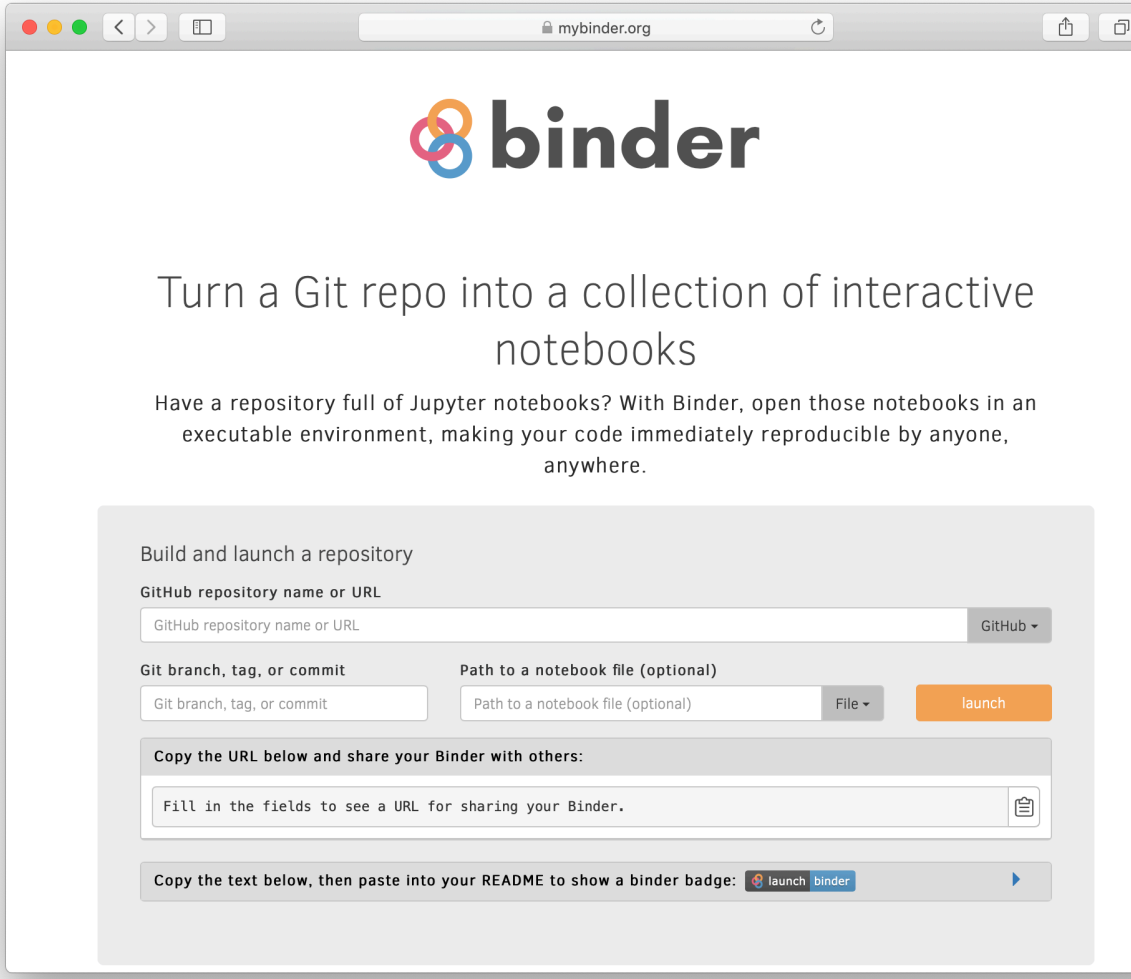
Jupyter for Reproducible Science

- One-study one-document: research driven from notebook
- Archive notebooks (and software environment) with data
- Publish notebooks with manuscript
  - For example one notebook per key statement / figure / table
- Strategy:
  Using Jupyter for reproducible scientific workflows. Computing in Science & Engineering, 10.1109/MCSE.2021.3052101 (2021)

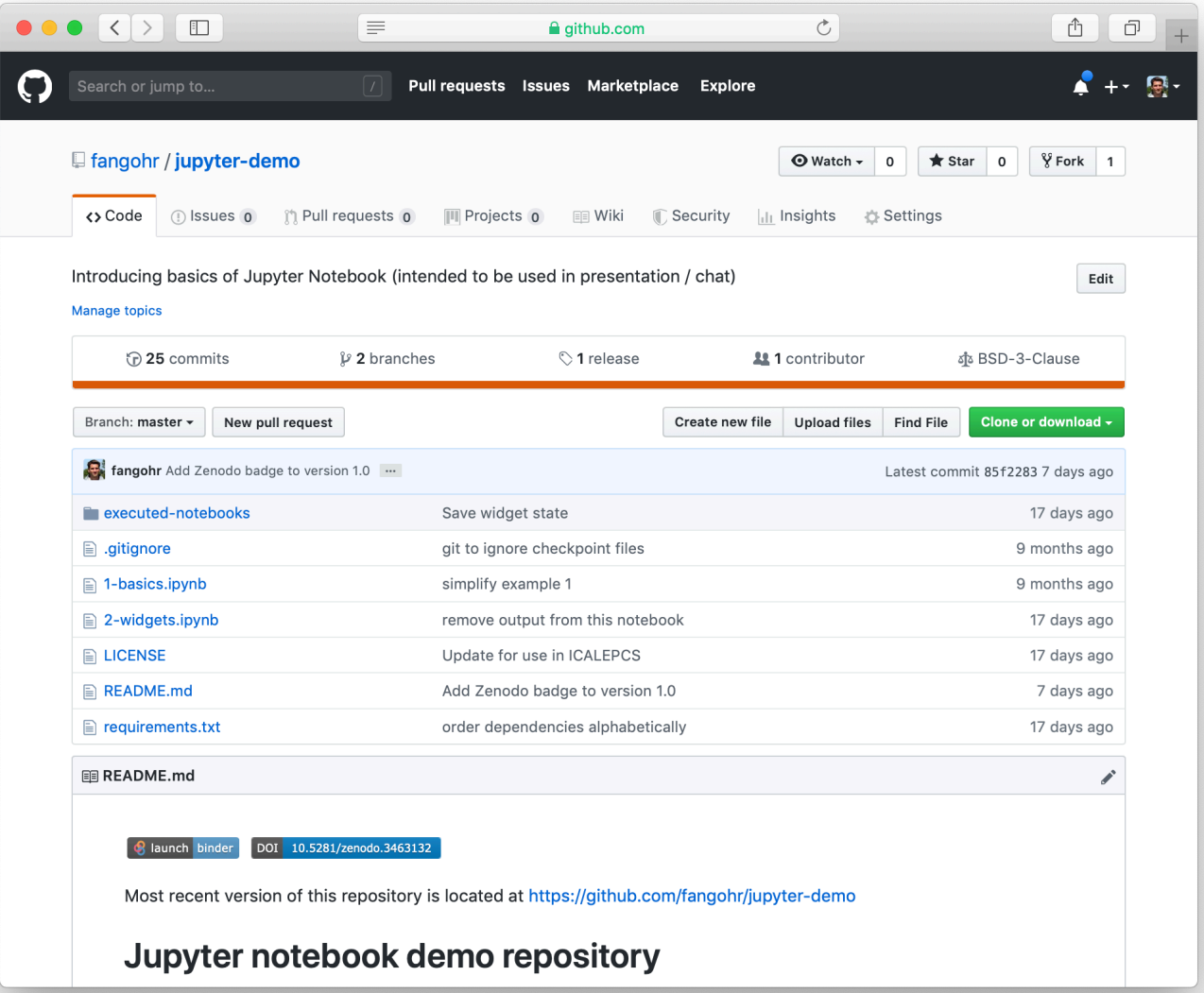


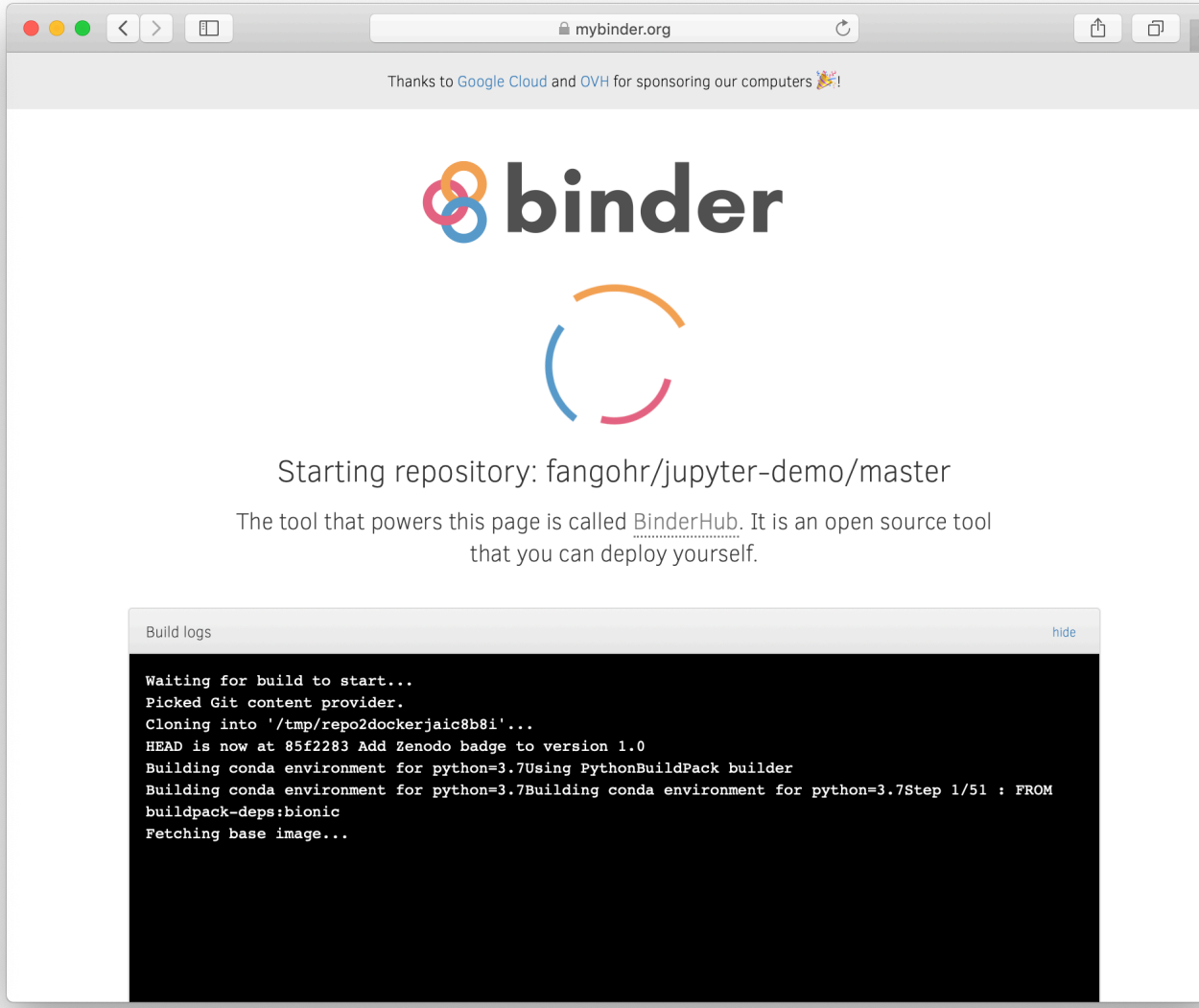
*Binder project

- Given a (Github) repository with
  - Jupyter notebooks
  - Software requirements (Dockerfile, requirements.txt, environment.yml)
- Binder service
  - builds a container with the required software (`repo2docker`)
  - starts Jupyter notebook server in that container offering the notebooks
- Alternative to Google Colab
- Public Binder service: https://mybinder.org

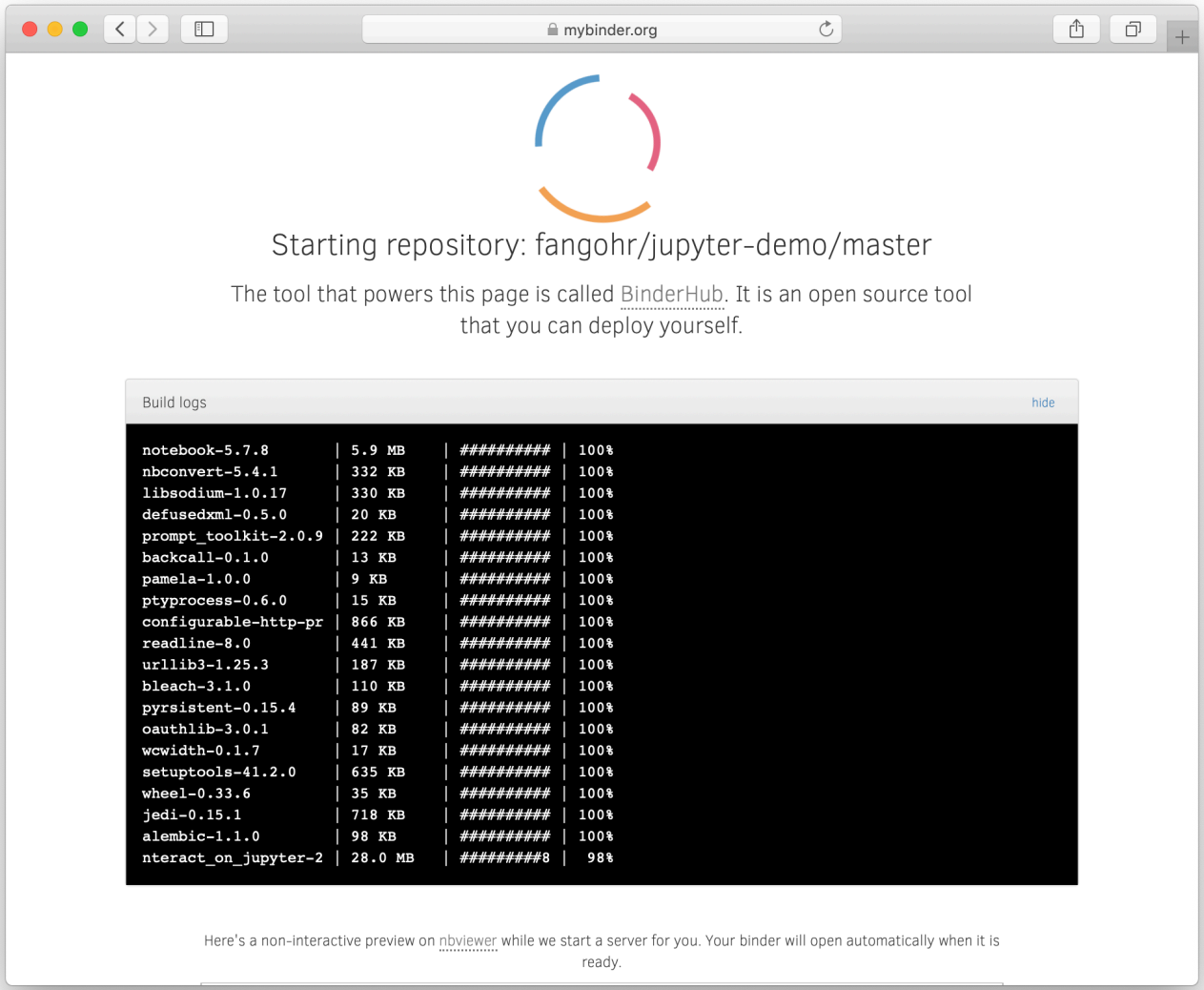


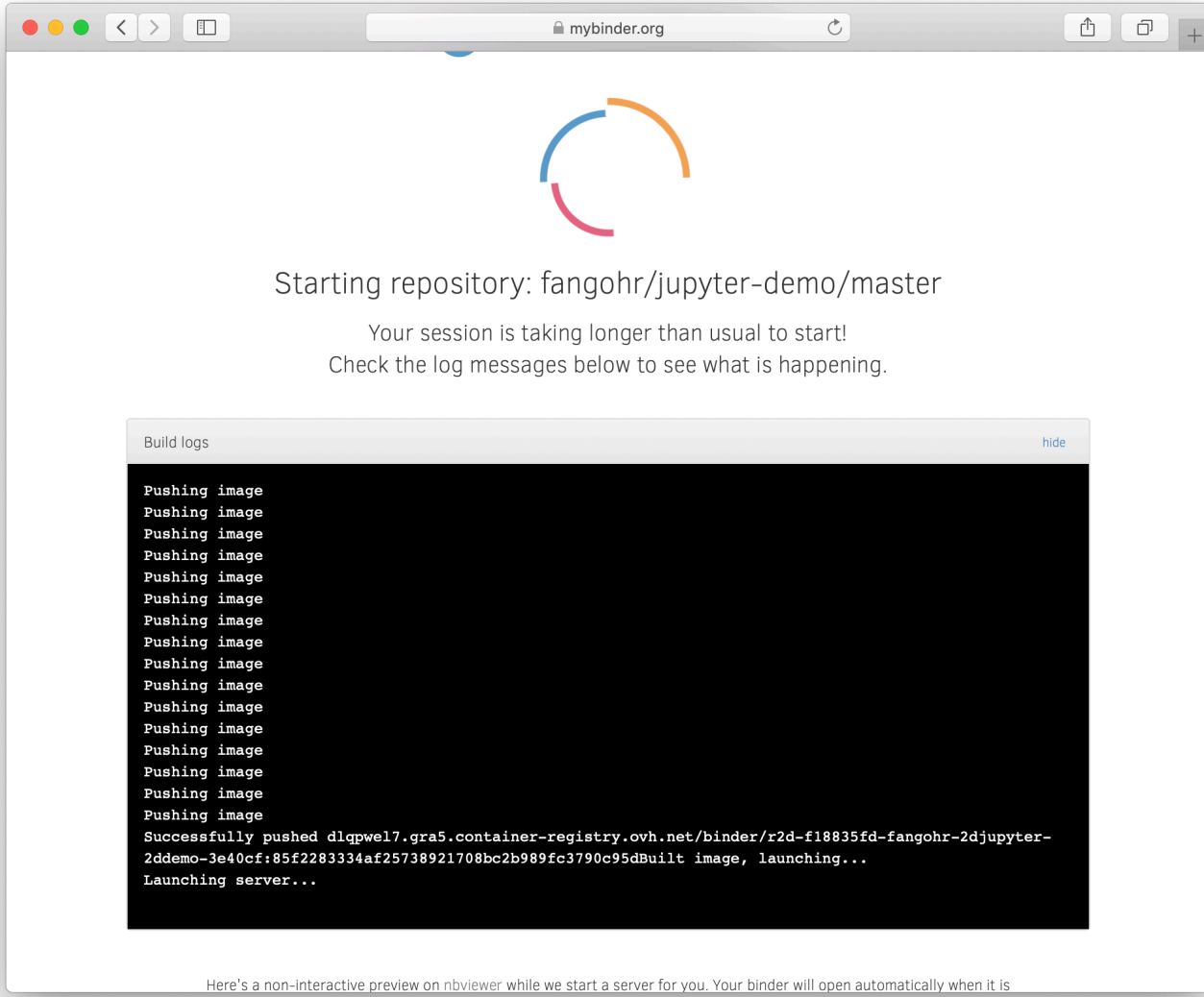
Example: https://github.com/fangohr/jupyter-demo



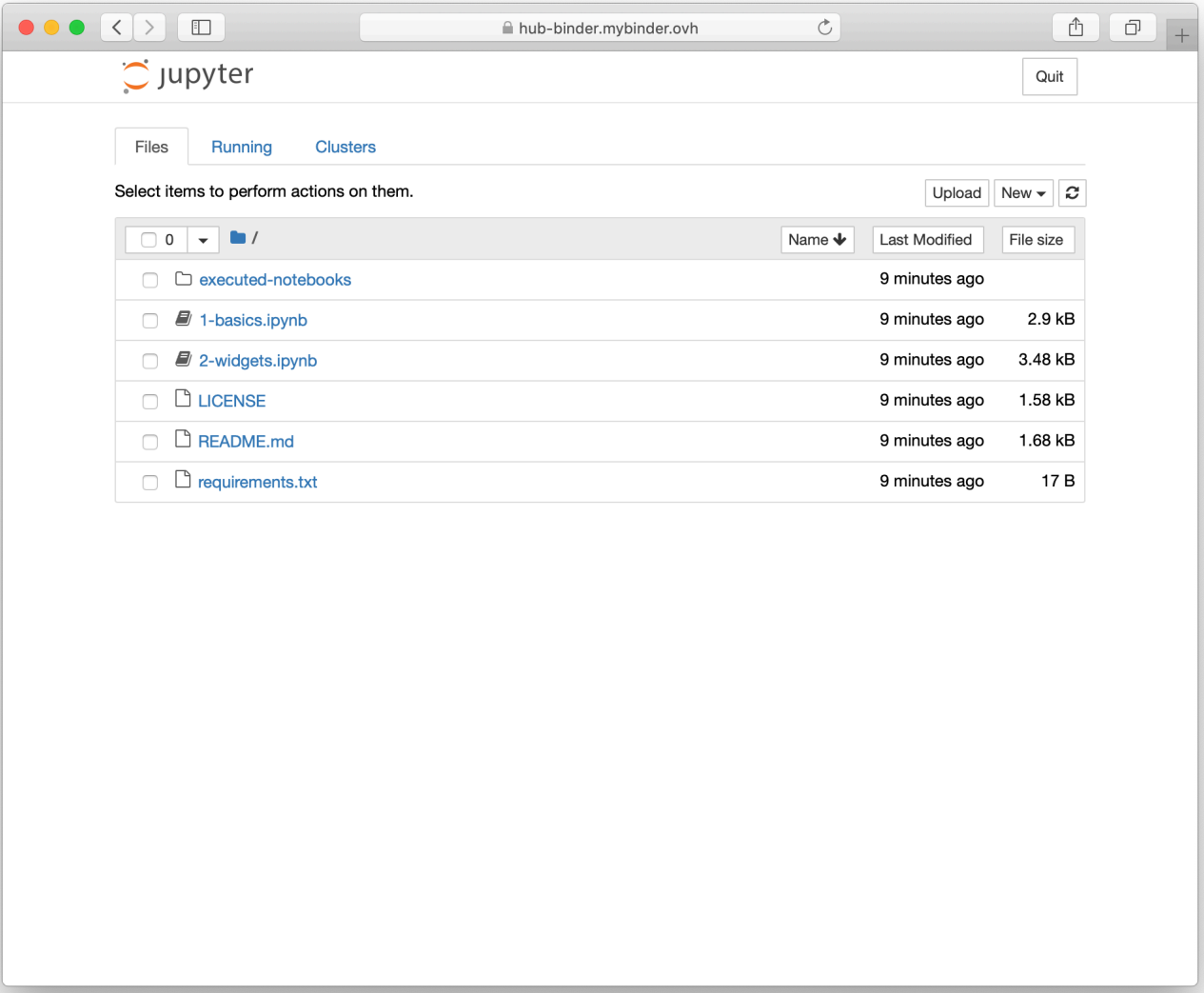


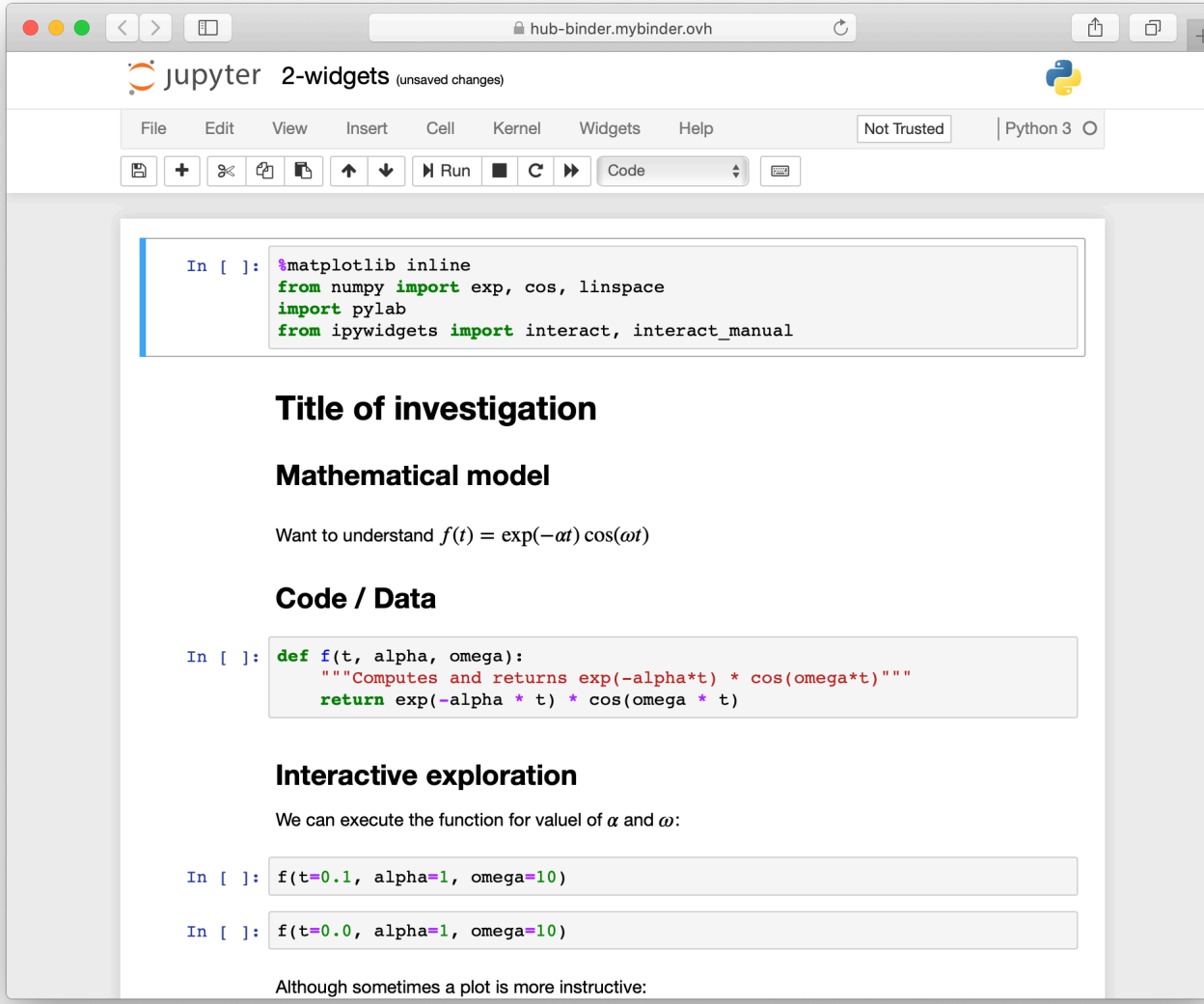
http://github.com/fangohr/jupyter-demo





Example: https://github.com/fangohr/jupyter-demo





Example: https://github.com/fangohr/jupyter-demo

Nanotechnology **27** (2016) 455502

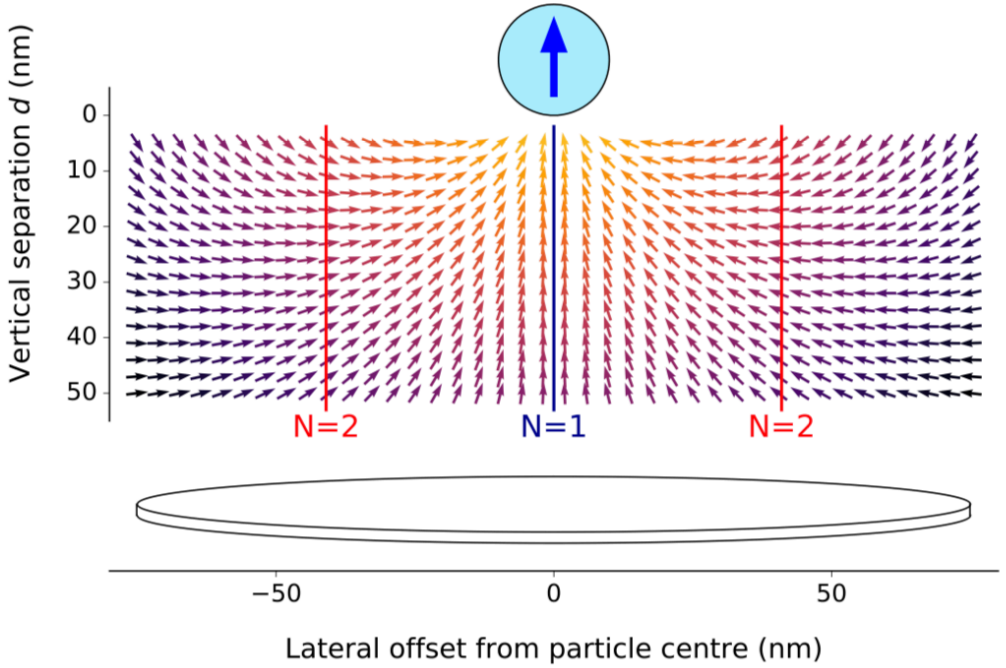
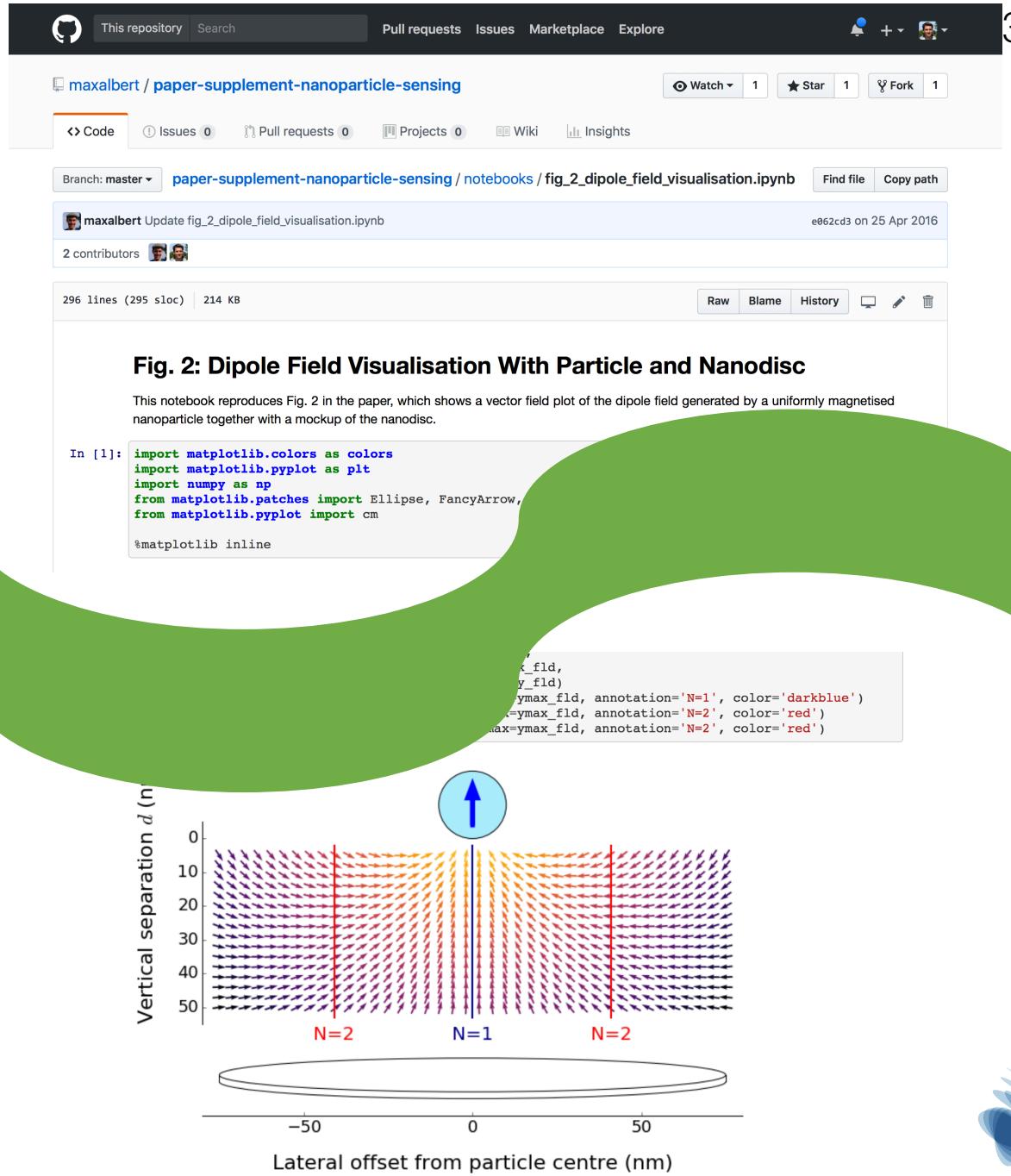


Figure 2. Vector field plot of the dipole field generated by a uniformly $+z$-magnetised MNP. The vectors are scaled to uniform length with their colour indicating the field strength (orange is high and violet/black is low). The vertical lines correspond to the x-values where modes 1 and 2 have maxima in their spin precession amplitude (see figures 3(a)–(b)). A schematic of the nanodisc is shown at the bottom.



Binder link available on:
https://github.com/maxalbert/paper-supplement-nanoparticle-sensing

Frequency-based nanoparticle sensing over large field ranges using the ferromagnetic resonances of a magnetic nanodisc: supplementary material

DOI 10.5281/zenodo.60605 preprint arxiv:1604.07277 launch binder license MIT

This repository accompanies the paper *"Frequency-based nanoparticle sensing over large field ranges using the ferromagnetic resonances of a magnetic nanodisc"*, published in *Nanotechnology*, Volume 27, Number 45. It provides the data underlying the figures in the paper, as well as Jupyter notebooks to reproduce those figures.

The latest version of this repository can be found at https://github.com/maxalbert/paper-supplement-nanoparticle-sensing

---

Authors: Maximilian Albert, Marijan Beg, Dmitri Chernyshenko, Marc-Antonio Bisotti, Rebecca L. Carey, Hans Fangohr and Peter Metaxas.

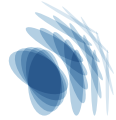
Contents

The directory `notebooks/` contains Jupyter notebooks for the relevant figures in the paper. On Github you can view them directly in the browser:

- Fig. 2: Dipole field visualisation
- Fig. 4: Frequency dependence on external field strength
- Fig. 7: Frequency change vs. lateral particle position

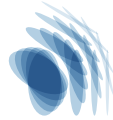
Summary – Jupyter notebook use cases

- Data analysis, One-study one-document
- Provision of recipes, report generator
- Remote data analysis (JupyterHub)
- Documentation of Software (also publishing of books)
- Reproducibility

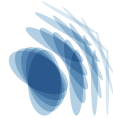
Additional remarks

- A wide ecosystem of tools is emerging for Project Jupyter
  - JupyterLab - a new interface to Jupyter Notebooks (Window manager in browser)
  - Many discipline specific libraries; rapid development; can be difficult to track
- Observed issues
  - Multiplication of code and notebooks, for example during intense data analysis or experiment period
  - State of notebook may be tricky: Good practice to restart and re-run to check

Summary Jupyter for Reproducible Science

- “One-study one-notebook” concept is powerful
- Some open issues:
  - Jupyter Notebook does not work for (i) all applications and (ii) all people
  - Archiving software environment
  - Studies with large data and large compute
- Binder shows interesting perspectives (NFDI, EOSC, GWDG, MPCDF, …?)

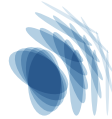


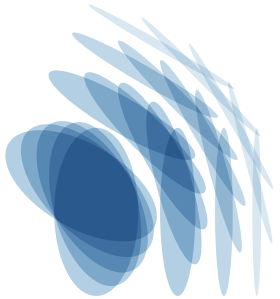
Acknowledgements

- Marijan Beg, Thomas Kluyver, Robert Rosca
- OpenDreamKit Horizon 2020, European Research Infrastructures project (No 676541), http://opendreamkit.org
- PaNOSC: Photon and Neutron Open Science Cloud, European Union's Horizon 2020 research and innovation programme under grant agreement (No 823852), http://panosc.eu

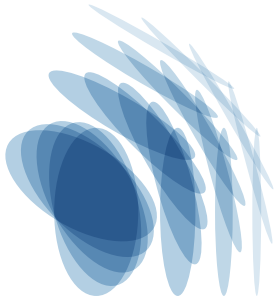
References

- Hans Fangohr et al, *Data exploration and analysis with Jupyter notebooks*, Proceedings of the 17th International Conference on Accelerator and Large Experimental Physics Control Systems ICALEPCS2019, TUCPR02, 10.18429/JACoW-ICALEPCS2019-TUCPR02 (2020)
- Marijan Beg et al, *Using Jupyter for reproducible scientific workflows*, Computing in Science & Engineering, vol. 23, no. 2, pp. 36-46, 10.1109/MCSE.2021.3052101 (2021)





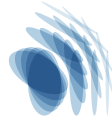
Tutorial



Appendix

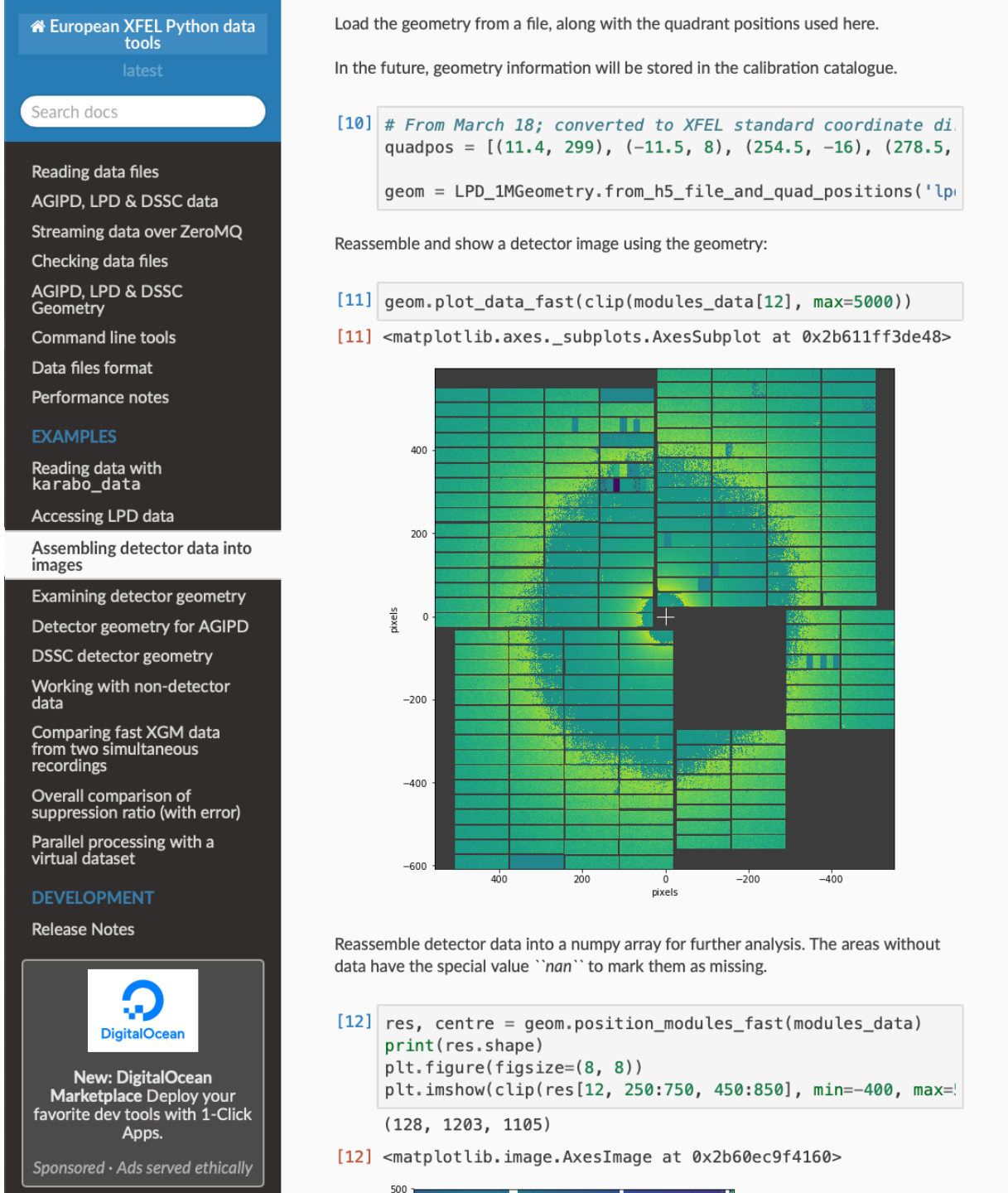
Example text books using Jupyter

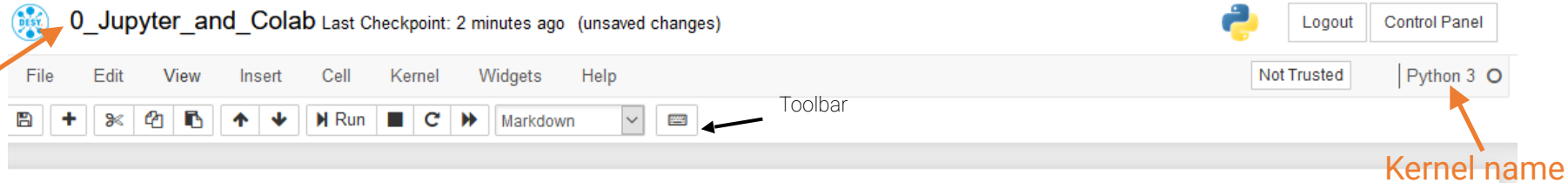
- François Chollet: *Deep Learning with Python*, https://github.com/fchollet/deep-learning-with-python-notebooks
- Wes McKinney: *Python for Data Analysis*, https://github.com/wesm/pydata-book
- Aurélien Géron: *Hands-on Machine Learning with Scikit-Learn, Keras and TensorFlow*, *https://github.com/ageron/handson-ml2*
- Hans Fangohr: *Introduction to Python for Computational Science and Engineering*, https://github.com/fangohr/introduction-to-python-for-computational-science-and-engineering/blob/master/Readme.md

Use case: documenting software library

- Use notebook as chapter in documentation
  - Supported by sphinx —> html, pdf
- Documentation easy to create:
  - enter commands in notebook
  - ouput is produced automatically
  - updating docs means re-running notebook
- Can run regression test on documentation notebooks using NoteBook VALidate (`nbval`)





Markdown cells

Jupyter Notebooks and Colab

Jupyter Notebooks are a widely used tool for data science. They contain both code and rich-text elements like paragraphs, equations and charts. Notebooks are both (a) human-readable documents used to present analysis and results; and (b) executable documents that your computer can be run.

Take 5 minutes to explore!

Currently selected cell

Important shortcuts

Action	Colab Shortcut
Executes current cell	`<CTRL-ENTER>`
Executes current cell and moves to next cell	`<SHIFT-ENTER>`
Insert cell above	`<CTRL-M> <A>`
Append cell below	`<CTRL-M> <B>`
Convert cell to code	`<CTRL-M> <Y>`
Convert cell to Markdown	`<CTRL-M> <M>`
Autocomplete	`<TAB>`
Goes from edit to "command" mode	`<ESC>`
Goes from "command" to edit mode	`<ENTER>`

Note: On OS X use `<COMMAND>` instead of `<CTRL>`

Types of cell

Notebooks are made up of a series of cells. There are two types:

- **Code cells.** These contain executable commands in Python.
- **Text ('markdown') cells.** These include plain text, or you can use markdown conventions to add formatting.

Code cell

```
In [6]:
1 print('Hello, I am a code cell')
2 print('Running me executes the code and renders the output below.')
```

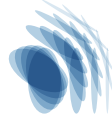
Code output

```
Hello, I am a code cell
Running me executes the code and renders the output below.
```

Sharing and exporting notebooks

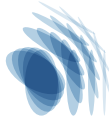
- Share `*.ipynb` files
  - Can be displayed using `jupyter-notebook`
  - Code can be re-executed on collaborator's machine
    - Only if software is available, and data is available, and collaborators know how to start notebook
- "Static sharing" through html and email (for example)
  - Often sufficient – does not require installation or additional skills
    - Effective to communicate with supervisors, line managers etc
  - Convert using menu "`File -> Download as -> html`"
  - Or using `nbconvert`:

```
$ jupyter-nbconvert --to html 2-widgets.ipynb
[NbConvertApp] Converting notebook 2-widgets.ipynb to html
[NbConvertApp] Writing 382609 bytes to 2-widgets.html
```

Summary – some tools in ecosystem

- Jupyter Lab – next generation Jupyter user interface
- Jupyter Hub – serving notebook from compute facility for multiple users
- NBDIME – DIffing and MErging tools
- NBVAL – VALidation tool; use each cell as a test
- NBCONVERT – conversion of notebooks to other formats & execution
- NBParametrize and Papermill – inject parameters into notebook files
- IPyWidgets – GUI like elements in notebook
- Binder – Cloud hosted execution of notebooks from github repositories
- There is much more.



* Executing a notebook from the command line (nbconvert)

- Convert from ipynb to html:

```
$ jupyter-nbconvert --to html 2-widgets.ipynb
```

- Can optionally set output name

```
$ jupyter-nbconvert --to html --output myout.html 2-widgets.ipynb
```

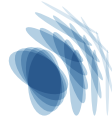
- Can execute notebook before conversion:

```
$ jupyter-nbconvert --execute --to html --output myout.html 2-widgets.ipynb
```

- Execute notebook and save results as notebook:

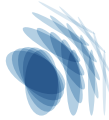
```
$ jupyter-nbconvert --execute --to ipynb --output myout.ipynb 2-widgets.ipynb
```

```
[NbConvertApp] Converting notebook 2-widgets.ipynb to ipynb
[NbConvertApp] Executing notebook with kernel: python3
[NbConvertApp] Writing 103398 bytes to myout.ipynb
```



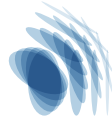
nbconvert

- `nbconvert` is a tool which can *execute and convert* notebooks to various other formats such as:
  - HTML, LaTeX, PDF, Markdown, ReStructuredText, Python, …
- Static HTML pages can be created as documentation or tutorial
  - `Nbsphinx` -> Jupyter notebook provides section in sphinx documentation
- Homepage: https://github.com/jupyter/nbconvert
- Notebook as a script approach
  - Can change parameters inside notebook before execution (`nbparametrize` or `papermill`)



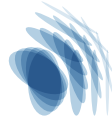
nbval

- py.test is a popular Python testing framework
- nbval is a py.test plugin which lets py.test recognise and collect Jupyter notebooks
- In each notebook, each cell is a test:
  - The test passes if execution of the input creates the stored output
  - The test fails otherwise
- There is a variety of configuration parameters
- Home page: https://github.com/computationalmodelling/nbval

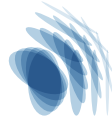
nbval example

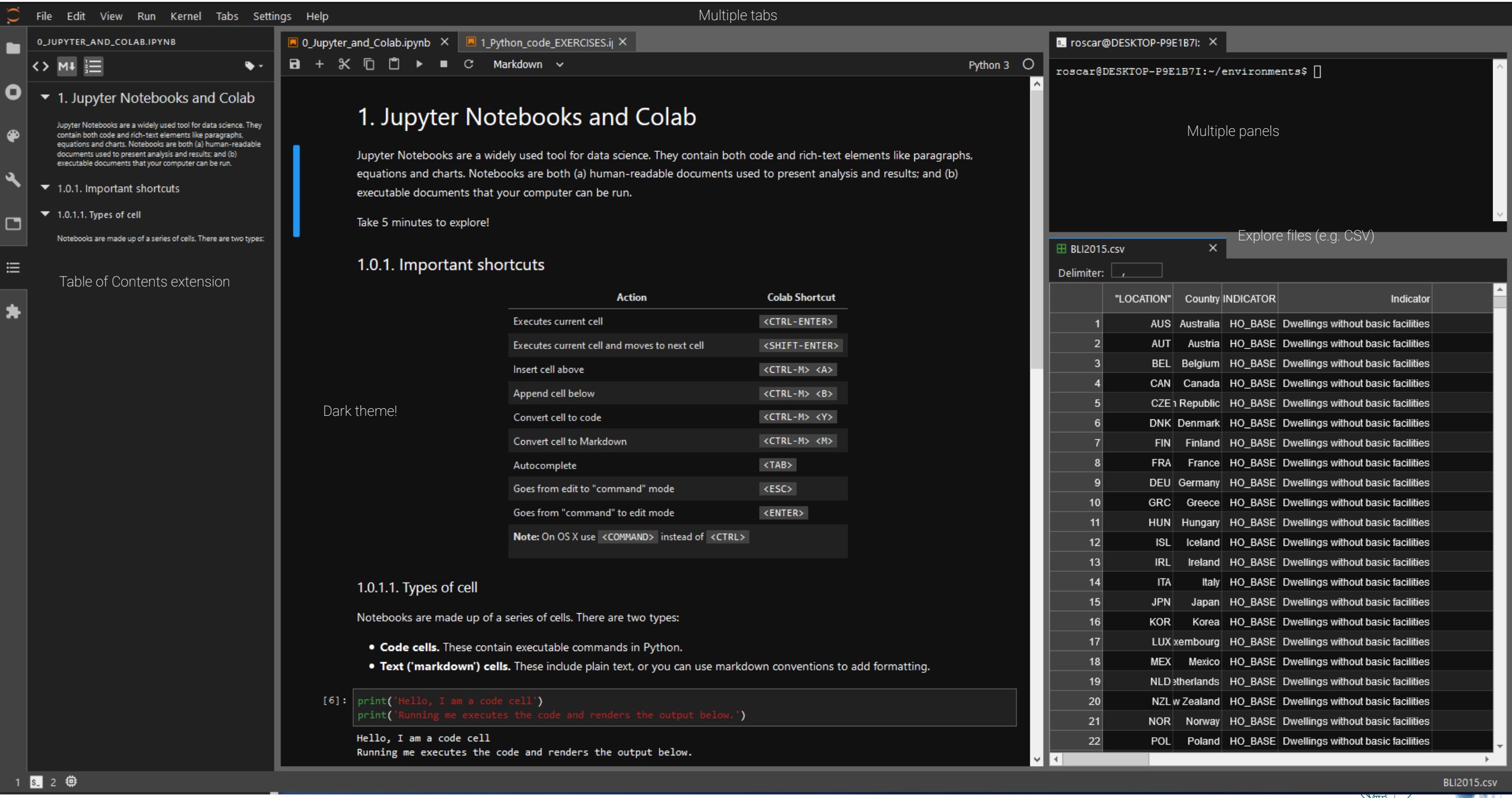
```
$ py.test --verbose --nbval 2-widgets.ipynb
============================= test session starts ==============================
platform darwin -- Python 3.6.8, pytest-3.10.0, py-1.7.0, pluggy-0.8.0 -- /Users/fangohr/anaconda3/
bin/python
rootdir: /Users/fangohr/Desktop/jupyter-demo, inifile:
plugins: remotedata-0.3.1, openfiles-0.3.0, doctestplus-0.1.3, arraydiff-0.2, nbval-0.9.1
collected 9 items

2-widgets::ipynb::Cell 0 PASSED                                          [ 11%]
2-widgets::ipynb::Cell 1 PASSED                                          [ 22%]
2-widgets::ipynb::Cell 2 PASSED                                          [ 33%]
2-widgets::ipynb::Cell 3 PASSED                                          [ 44%]
2-widgets::ipynb::Cell 4 PASSED                                          [ 55%]
2-widgets::ipynb::Cell 5 PASSED                                          [ 66%]
2-widgets::ipynb::Cell 6 PASSED                                          [ 77%]
2-widgets::ipynb::Cell 7 PASSED                                          [ 88%]
2-widgets::ipynb::Cell 8 PASSED                                          [100%]

==================== 9 passed, 1 warning in 2.11 seconds =====================
(base) [20:49:09] fangohr:jupyter-demo git:(master*) $
```



Jupyter Lab

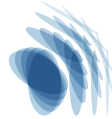
- Next generation Jupyter notebook interface
- Window manager embedded in browser
- “classic notebook“ in one window
- Additional features in other windows
  - File browser
  - Extensions
  - CSV viewer.




File Edit View Run Kernel Tabs Settings Help
Multiple tabs
0_JUPYTER_AND_COLAB.IPYNB
1. Jupyter Notebooks and Colab
Jupyter Notebooks are a widely used tool for data science. They contain both code and rich-text elements like paragraphs, equations and charts. Notebooks are both (a) human-readable documents used to present analysis and results; and (b) executable documents that your computer can be run.
1.0.1. Important shortcuts
1.0.1.1. Types of cell
Notebooks are made up of a series of cells. There are two types:
Table of Contents extension
0_Jupyter_and_Colab.ipynb
1_Python_code_EXERCISES.i
Markdown
Python 3
1. Jupyter Notebooks and Colab
Jupyter Notebooks are a widely used tool for data science. They contain both code and rich-text elements like paragraphs, equations and charts. Notebooks are both (a) human-readable documents used to present analysis and results; and (b) executable documents that your computer can be run.
Take 5 minutes to explore!
1.0.1. Important shortcuts
Dark theme!
Action
Colab Shortcut
Executes current cell
<CTRL-ENTER>
Executes current cell and moves to next cell
<SHIFT-ENTER>
Insert cell above
<CTRL-M> <A>
Append cell below
<CTRL-M> <B>
Convert cell to code
<CTRL-M> <Y>
Convert cell to Markdown
<CTRL-M> <M>
Autocomplete
<TAB>
Goes from edit to "command" mode
<ESC>
Goes from "command" to edit mode
<ENTER>
Note: On OS X use <COMMAND> instead of <CTRL>
1.0.1.1. Types of cell
Notebooks are made up of a series of cells. There are two types:
Code cells. These contain executable commands in Python.
Text ('markdown') cells. These include plain text, or you can use markdown conventions to add formatting.
[6]:
print('Hello, I am a code cell')
print('Running me executes the code and renders the output below.')
Hello, I am a code cell
Running me executes the code and renders the output below.
roscar@DESKTOP-P9E1B7I:
roscar@DESKTOP-P9E1B7I:~/environments$
Multiple panels
Explore files (e.g. CSV)
BLI2015.csv
Delimiter: ,
"LOCATION" Country INDICATOR Indicator
1 AUS Australia HO_BASE Dwellings without basic facilities
2 AUT Austria HO_BASE Dwellings without basic facilities
3 BEL Belgium HO_BASE Dwellings without basic facilities
4 CAN Canada HO_BASE Dwellings without basic facilities
5 CZE ı Republic HO_BASE Dwellings without basic facilities
6 DNK Denmark HO_BASE Dwellings without basic facilities
7 FIN Finland HO_BASE Dwellings without basic facilities
8 FRA France HO_BASE Dwellings without basic facilities
9 DEU Germany HO_BASE Dwellings without basic facilities
10 GRC Greece HO_BASE Dwellings without basic facilities
11 HUN Hungary HO_BASE Dwellings without basic facilities
12 ISL Iceland HO_BASE Dwellings without basic facilities
13 IRL Ireland HO_BASE Dwellings without basic facilities
14 ITA Italy HO_BASE Dwellings without basic facilities
15 JPN Japan HO_BASE Dwellings without basic facilities
16 KOR Korea HO_BASE Dwellings without basic facilities
17 LUX xembourg HO_BASE Dwellings without basic facilities
18 MEX Mexico HO_BASE Dwellings without basic facilities
19 NLD etherlands HO_BASE Dwellings without basic facilities
20 NZL w Zealand HO_BASE Dwellings without basic facilities
21 NOR Norway HO_BASE Dwellings without basic facilities
22 POL Poland HO_BASE Dwellings without basic facilities
1 2
BLI2015.csv

Jupyter Notebook vs. JupyterLab

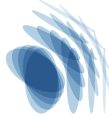
- JupyterLab is the newer interface to the notebooks
- Window manager in browser
- It provides a more flexible interface closer to a modern IDE
- Flexible layouts and panes
- Console
- Drag/Drop/Expand/Collapse cells
- Themes
- Extensions

Nbdime – NoteBook DIff and MErge

- Jupyter notebooks are rich media documents, stored as plain text JSON files
- Basic diff and merge tools (such as that used by git) do not handle this format well
    - Small changes to text/plots result in unreadable diffs
- `nbdime` provides multiple tools to help with "content-aware" diffing and merging for Jupyter notebook files
- Homepage: -https://github.com/jupyter/nbdime

- `nbdiff` compare notebooks in a terminal-friendly way
- `nbmerge` three-way merge of notebooks with automatic conflict resolution
- `nbdiff-web` shows you a rich rendered diff of notebooks
- `nbmerge-web` gives you a web-based three-way merge tool for notebooks
- `nbshow` present a single notebook in a terminal-friendly way

notebook-v1.ipynb | notebook-v2.ipynb

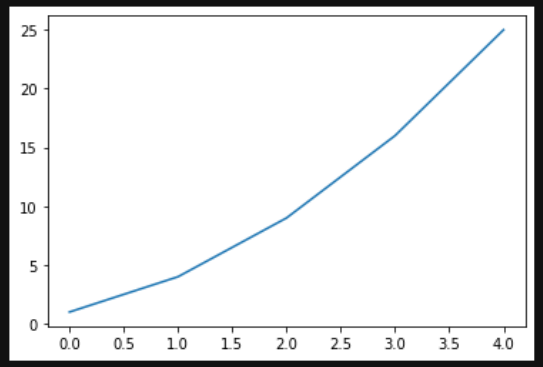
Markdown | Python 3

1. This is a placeholder title!

Here's some description text, maybe has a tpyo in it...

Here's some text that won't change.

```
[1]: import matplotlib.pyplot as plt

[2]: amazing_math = [1, 2, 3, 4, 5]
     amazing_math = [x**2 for x in amazing_math]

[3]: plt.plot(amazing_math)

[3]: [<matplotlib.lines.Line2D at 0x7f30ab9b86d8>]
```



TODO:

- [ ] Better title
- [ ] Fix typo
- [ ] Section headings
- [ ] Go up to 6 squared!

Mode: Command | Ln 1, Col 1 | notebook-v1.ipynb

notebook-v1.ipynb | notebook-v2.ipynb

Markdown | Python 3

1. Example of Notebook Diffing!

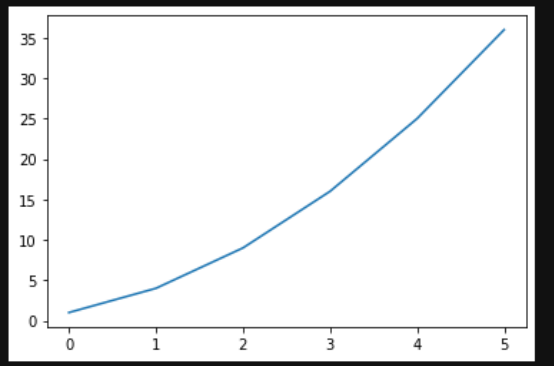
Here's some description text, now typo free!

Here's some text that won't change.

```
[1]: import matplotlib.pyplot as plt

[2]: amazing_math = [1, 2, 3, 4, 5, 6]
     amazing_math = [x**2 for x in amazing_math]
```

1.1. A Plot

```
[3]: plt.plot(amazing_math)

[3]: [<matplotlib.lines.Line2D at 0x7f7cbc8395f8>]
```



TODO:

- [x] Better title
- [x] Fix typo
- [x] Section headings
- [x] Go up to 6 squared!

Mode: Command | Ln 1, Col 1 | notebook-v2.ipynb

```
diff notebook-v1.ipynb notebook-v2.ipynb
7c7
<     "# This is a placeholder title!"
---
>     "# Example of Notebook Diffing!"
14c14
<     "Here's some description text, maybe has a tpyo in it...\n",
---
>     "Here's some description text, now typo free!\n",
34c34
<     "amazing_math = [1, 2, 3, 4, 5]\n",
---
>     "amazing_math = [1, 2, 3, 4, 5, 6]\n",
39a40,46
>    "cell_type": "markdown",
>    "metadata": {},
>    "source": [
>     "## A Plot"
>    ]
>   },
>   {
47c54
<        "[<matplotlib.lines.Line2D at 0x7f30ab9b86d8>]"
---
>        "[<matplotlib.lines.Line2D at 0x7f7cbc8395f8>]"
56c63
<       "image/png": "iVBORw0KGgoAAAANSUhEUgAAAXAAAAD4CAYAAAD1jb0+AAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAALEgAACxIB0
```


```
t1+/AAAADh0RVh0U29mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uMy4xLjEsIGh0dHA6Ly9tYXRwbG90bGliLm9yZy8QZhcZAAAgAElEQVR4nO3d
eXwU9eH/8dcHEgjhCFdCwhHCHe4rYERBFKWgVqpWBZTiUelatbb9ae2E961UFRFRO7zsggopYOCaIDcu5/fH9n6o5RASHZ3dpY
38/HIg83MJPN2zLwzmZ35jLHWIiIiwaeO0wFERKRqVOAiIkFKBS4EqRU4CIiQUoFLiISpEL8ubKWLVvauLg4f65SRCTorV27NttaG3nqdL8WeF
xcHGvWrPHnKkVEgp4xZu/ppusUiohIkFKBi4gEKRW4iEiQUoGLiAQpFbiISJA6a4EbY9oZY74xxmw2xmwytrmf6EMeagMWa95+Ny38cVEZH/q
MxlhGXAg9baZGNMY2CtMWaxZ97z1tpnfBdPREQqctYjcGtturU22pIwnPt1EbmGp17/3OZ0DN8bEAf2BlZ5JdxtjNhpj
ZhpjmlXwNZONMWuMMWuysrKqFVZEJJgcLy7j5pmrefvHPazde9Tr37/SBW6MaQR8AXxncGzmoCzp/s6a+00a22CtYHhMvJ/7gQVEam
RcgtLmfDGStbuO8aL4/pzSXwr6+jUrfSG2NCKS/vOdbaDwGstYdPmj8d+Nzr6UREglBOQQm/mrmKFLel5vDJ+AKN6RftkPZW5CsUAbwBbrLXPnT
Q95qTFrgZSvR9PRCS4HD1RwvjpLOlLz+e1mwb6rLyhckfgFwAueA
uDW2uWAOc2sL70fR0QkOB3OK2L89BUcyinizZsHMaRzS5+v06/DyYqI1ESHcgoZP30FWfnFvHXrYAZ3a06XarARUSqYf/RAsZNX0FuQSlv33Ye
A9uf9opqn1CBi4hU0Z7sE4yfvolITJS7mTDqPPm2b+nX9KnARkZN8sTGde+evo3evjFIgXX6UiVogIXkWtrX/v4FFPN3FJfBSv3jiAsNDgKG9QgYtILTZ
RkXOw+VAeN72xkpA6hrmTzqdzVCPHsmg8cBGRSko5kMu46SuoH1HBXc4W96gI3ARkUpJ3neMiTNXEdEglHmTEmnXPNzpSDoCFxE5m9V7jjJhxk
qaN6zHgjvOD4jyBh2Bi4ic0b93ZnPbrDxENmaNvmaVMmMBVd6gI3ARkdNamnaYO99JplUFxVpgwlJtWdzy5mraNmvA
eo7Hel/qMBFRE6xaFMGd89Nj66Ce/cNpim4fWcjhRaKnARkZN8sTGde+evo3bxAHjVg4dcc76L/d/2ZtsNBlqbMVIEVTpNUHgWBCo
LiICwLtr9vPIBxtJ7NCCGRMTaFg/8Osx8BOKiPjY3JX7eOyjFIZ2acm0CQk0qfFX6UiVogIXkVrtrX/v4fFPN3FJfBSv3jiAsNDgKG9QgYtILTZ
92S7++uUWRvZoxcvjBi1AvJLjeFlSBi0it9Mo3O3h60Vau6B3DC2P7EVo3uMobVOAiUstYa3nh6+38cl2fTGvNc9c15eQICxvUIGLSC1ireXpRV
t59dud/HJgW566gtg916xinY1WZClxEagVrLX/9Ygszlu9m/Hmx/GVML+oEcXmDClxEagG32/Knzzbx1c9o7uFrxL//VcyIHI//vAfGBHd5gwpcRGo4t9voy
49TmLdqP5OGduCxy7XiPIGFbiI1GAut+WDRzby/toHXqJmqJx4a2a3GlDeowEWkhipzuXwzNwCQzOQcjUzqfn8OLY/4aFzQoUwNCSAhZEk9GJ9wSSdpxbDOt4k
pKTz8KhuTBne2elIPqECF5Eabj1xd1z17F482H+cEV3bh/a0elIPqMCF5Eao6jUxa9nr+WbrVn86aqeTBwS53Qkn1KBi0iNUFjiIvI7a1i+I5u
/Xd2b8efF0h3J51TgIhL0ThSxcdtbq1m5+yj/uLYP1yW0czqSbYxYbY7Z7/m3m+7giIv8tv6iUiTNXsX
rPMV64oV+tKW+o3CPVyoAHrbU9gETgLmNMD+BRY1m0tguwPO5ij5fqbbqyN1rzOB7YAbYAxwFuex
d4CfuGrkCIipzp2ooQbZ6xg06+wKxrX+AfYXE+dwXkpg3ot0WEmhlrU33MoAWlXwNZONMWuMWwuysrRYZcyN7CbYePM21C
AiN7RjsdyRGVLnBjTCPgA+A+a23eyfOstRawp/s6a+00a22CtCtYhMjKyWmFFRLzixg7bQV7Ykx1SqwI0xoZSX9xxr7YeeyTe
NMTGe+TFApm8iioiUy8gtYuzrKziYU8ibNw9maJafVBYmatQDPAGsMVa+9xJsz4FJnezTwQ+8X48EZyB3MKuWhaj2TmF/P2rYM5v1MLpyM5rj
LXgV8ATABSjDHrPdMeA6YC7xpjbjbP8Atf7JqKI1Hb7jhQwbvoK8opKeee2wwDGPnSgSbKYOxmpLL7yUbenkjvt
hFORwoYuhNTRALWjsx8k9f5ZnbMvf2RHq0buJ0pICiAheRgLQ1I58bsUjBYoawmuEd+A1+JJ22f5xGJGD9YWlLubenkjvt
KBv25zB++goahNZeY1AUunEQkYKzde4ybZ66iacNQ5t6sE5SLvm4U5HCmgqcBEJCCt3HeWAwmFBcpIJhHKwJahJGnNvPo3XTBk5HCngqcBFx3A8
7srn9rTW0bGvenGbrJDUJczpSM2vAWp2jGjH79vNo3rCeGMwFAlHNEz8BkwLNPfabvn1Zu
3jGjN3ksq7KnQELiJ+99mQCqVvTqcfi0MIcqVWNxXR1z49CLenmKjXX79Q9Lxap6HlvEwB+t9zIOUK1mq4jtJQKoEyuPl1fkmsvJTq
/gQ+TD3LPJZ154LKukm8/UYGLSJWvudzc/+4GPttwKSCvcQWuXzJXmQ8I4YcnQ2G9p3Cn7qfC
y2Z/PHKHtx2YQenI9VKNAROSeFJS7unL2W77Zl8ecxPZlwfpzTkWotFbiIVNqOzOPcNSeZmcySn6VHmfP82jVWUi1CG
zLplMBd1jXQ6Uq2nAheRMyoqdfHnzzczJ4WT9BUAEWoTcLrLEhIJnsPfICab
fJXBZj1ZOR5JTnPVXqTFmpjEm0xiTetK0J4wTU0Qkuq42u9OMqZXHALXKxiDJMKhz5Hb8xQeA2LfRuzKJiLkSaaHzPAhe
GcQu6ak8z6/TncPCS03310eT/0QPf4sUJ21wK6Yz+3wxcb6PIiJOWxgMM
G6dtYaYiAZ8ds+FKu8gUdUC/xfQCegHpAPPVrSgMwayMWAfe3UCZy/LK4+AFc0SfG6UhyFtV5N+JuY8xGzmWZl5LJ
Mx34/AzLTgOmASQkJNiqrE9EvG/59mzunb+OghIzX9/Ql6v7t3Uk
OJVVw1gI3xswDhgMtjTEHgMeBbK4sMeB4caYfoAF9gB3ePZA/WVkDMr/9XYX0nX
Ze05sbBrXTk3OCnH71itRwbrfl9WW7eOarrcQ2D2fWLYPp0bJ07HEC1TIjXYsRMlPPjeBpamZXJzbjzUckHc5YNnY4llVSlq1CstYf/89oY
8jML+LJMT2ZkNhep0xqGBW4SA1jreXNH/bw96xHX
J5mDOYX8/vLu3D60g06Z1HAqcJEgZ61lzsp9PPn5GzpQHp9zh7UHV4Q+Ahn7
+U+asZXf2CR4a2ZUpwzTTp450dQmKnCRIPT2gP8
```


```
nbdiff notebook-v1.ipynb notebook-v2.ipynb
--- notebook-v1.ipynb  2019-09-24 22:43:55.669188
+++ notebook-v2.ipynb  2019-09-24 22:43:57.525607
## inserted before /cells/0:
+  markdown cell:
+    source:
+      # Example of Notebook Diffing!

## deleted /cells/0:
-  markdown cell:
-    source:
-      # This is a placeholder title!

## modified /cells/1/source:
@@ -1,3 +1,3 @@
-Here's some description text, maybe has a tpyo in it...
+Here's some description text, now typo free!

 Here's some text that won't change.


## modified /cells/3/source:
@@ -1,3 +1,3 @@
-amazing_math = [1, 2, 3, 4, 5]
+amazing_math = [1, 2, 3, 4, 5, 6]

 amazing_math = [x**2 for x in amazing_math]


## inserted before /cells/4:
+  markdown cell:
+    source:
+      ## A Plot

## modified /cells/4/outputs/0/data/text/plain:
-  [<matplotlib.lines.Line2D at 0x7f30ab9b86d8>]
+  [<matplotlib.lines.Line2D at 0x7f7cbc8395f8>]

## inserted before /cells/4/outputs/1:
+  output:
+    output_type: display_data
+    data:
+      image/png: iVBORw0K...<snip base64, md5=f6fa16789fcf2488...>
+      text/plain: <Figure size 432x288 with 1 Axes>
+    metadata (unknown keys):
+      needs_background: light

## deleted /cells/4/outputs/1:
-  output:
-    output_type: display_data
-    data:
-      image/png: iVBORw0K...<snip base64, md5=01fb4236a7c64f48...>
-      text/plain: <Figure size 432x288 with 1 Axes>
-    metadata (unknown keys):
-      needs_background: light

## modified /cells/5/source:
@@ -1,5 +1,5 @@
 TODO:
- - [ ] Better title
- - [ ] Fix typo
- - [ ] Section headings
- - [ ] Go up to 6 squared!

+ - [x] Better title
```