FAIR_bioinfo : Open Science and FAIR principles in a bioinformatics project

How to make a bioinformatics project more reproducible

C. Hernandez[1] T. Denecker[2] J. Seiler[2] G. Le Corguillé[2]
C. Toffano-Nioche[1]

[1]Institute for Integrative Biology of the Cell (I2BC)
UMR 9198, Université Paris-Sud, CNRS, CEA
91190 - Gif-sur-Yvette, France

[2]IFB Core Cluster taskforce

June 2021

General information

Practical information:

- Dates: June 28th - 30th
- Location: Institut des Systèmes Complexes, 113 rue Nationale, 75013-Paris
- Courses: 9:00 to 17:30
- Meal: 12:30-14:00
- Pauses: 10:30-11:00 + 15:30-16:00
- 2 days of courses + 1 day of course building

Round table:

- Teachers
- Learners

Ressources:

- 

- GitLab
- LaTeX

Training schedule

Day 1:

- Introduction to reproducibility
- History management (3 Practical Sessions, git, GitHub)
- Control your development environment (1 PS, CONDA)
- Encapsulation (2 PS, docker)

Day 2:

- Workflow (2 PS, SNAKEMAKE)
- Traceability with notebooks (2 PS, jupyter,)
- IFB resources (2 PS, slurm,)
- Sharing and disseminating (GitHub, zenodo)
- Conclusion

Day 3:

- Empowerment and improvement of resources

Table of contents

Literate programming

Introduction

What is literate programming ?
Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to humans what we want the computer to do.
— Donald E. Knuth, Literate Programming, 1984

Introduction

What is literate programming ?

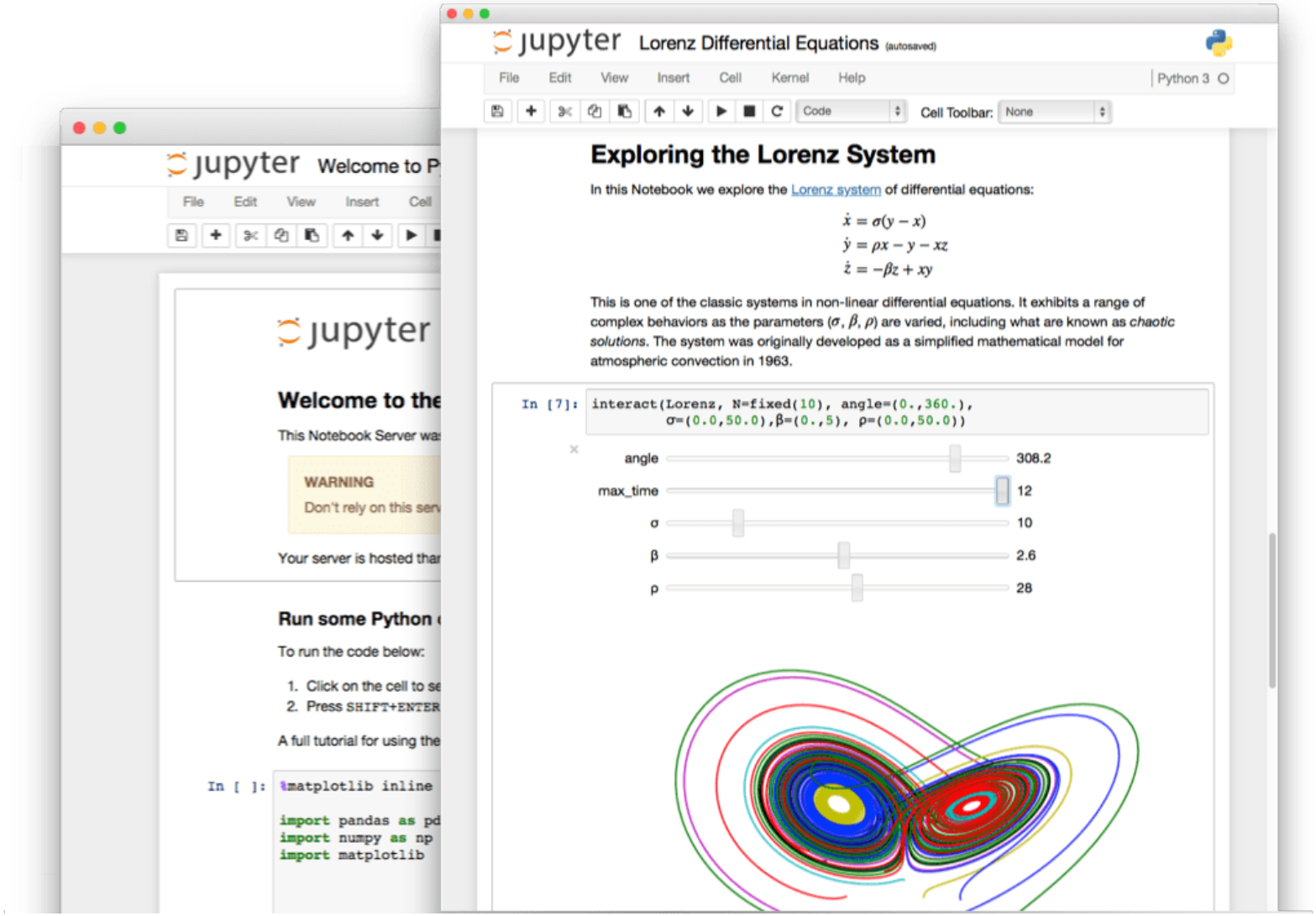
Definition

"Literate programming is a programming paradigm introduced by Donald Knuth in which a computer program is given an explanation of its logic in a natural language, such as English, interspersed with snippets of macros and traditional source code, from which compilable source code can be generated." Donald Knuth, 1984.

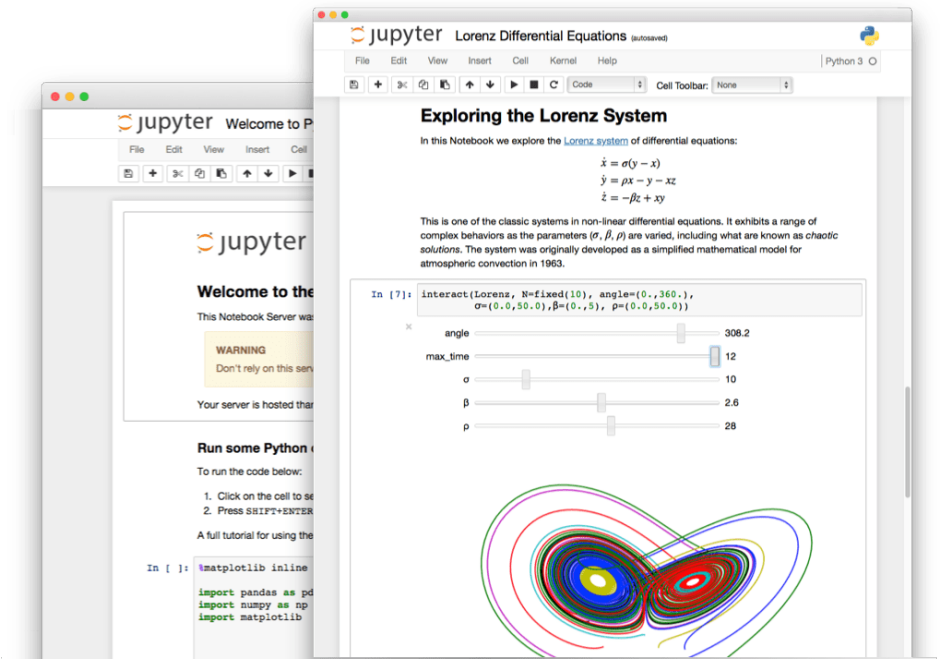
Wikipedia, 18/08/2020
`https://en.wikipedia.org/wiki/Literate_programming#Workflow`

Introduction

What does it look like ?



Introduction



Interactive programming interface allowing to combine both natural and computer languages.

In one file:

- Explanations
- Code
- Results
- Graphs and plots

Introduction

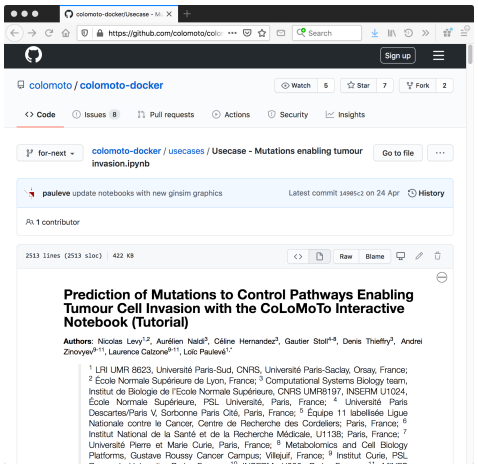
Why using literate programming frameworks ?

Use cases:

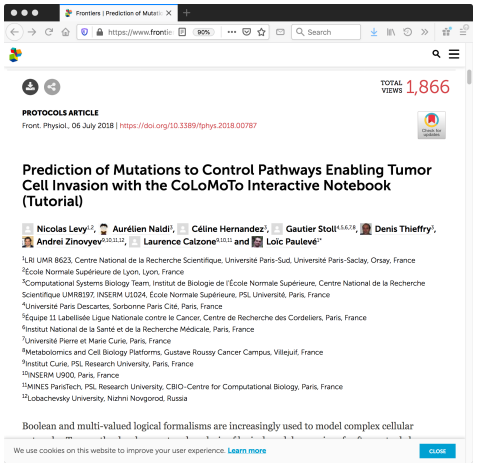
- Day to day analyses
- Analysis reports
- Writing scientific articles

Example of an article entirely written using a notebook

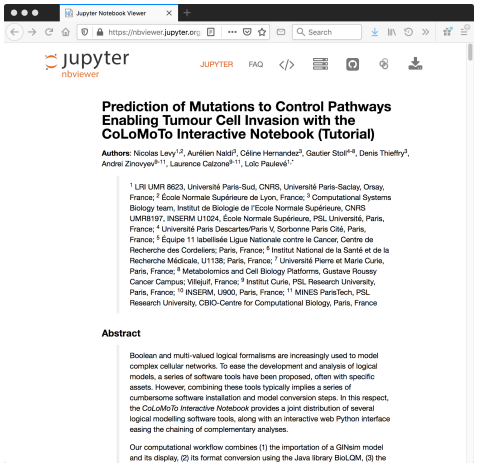
File (on a repository)



Published article



Executable file



Literate programming

This session :

- Markdown
- Rmarkdown / RStudio
- Jupyter

Markup and markdown

Definition

A markup language uses tags to define elements within a document.

Three different types and usage :

- Presentational (used by traditional word-processing systems)
  - Markup is invisible
- Procedural, provides instructions to process the text (e.g. TeX, PostScript)
  - Markup is visible and can be directly manipulated by the author.
- Descriptive, to label documents parts (e.g. LaTeX, HTML, XML...)
  - Emphasizes the document structure.

Markdown language

Markdown is a Lightweight markup language.
Designed to be :

- easy to write using any generic text editor (plain-text-formatting syntax)
- easy to read in its raw form

Markdown language

You've probably see it already on GitHub (README), Wikipedia...

```
# Heading

## Sub-heading

### Another deeper heading

A [link](http://example.com).

Text attributes _italic_, *italic*, **bold**, `monospace`.

Bullet list:

  * apples
  * oranges
  * pears
```

Github guide :
urlhttps://guides.github.com/features/mastering-markdown/

Literate programming

But how is this useful for literate programming?

When you want to weave both code (to be interpreted) and formatting information, you precisely need a lightweight language for the formatting part.

The challengers

No need to hide, there are currently two main frameworks used in bioinformatics:
RMarkdown and Jupyter

RMarkdown

RMarkdown

At the beginning, there was nothing.

RMarkdown

At the beginning, there was nothing.

Then came Sweave.
Leisch, Friedrich (2002). "Sweave, Part I: Mixing R and LaTeX: A short introduction to the Sweave file format and corresponding R functions"

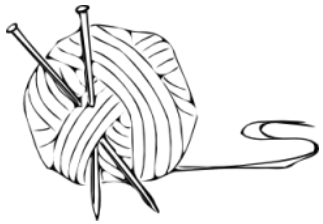
RMarkdown

At the beginning, there was nothing.

Then came Sweave.
Leisch, Friedrich (2002). "Sweave, Part I: Mixing R and LaTeX: A short introduction to the Sweave file format and corresponding R functions"
And people saw that the path would be long...

RMarkdown

knitR (2011)



"The knitr package was designed to be a transparent engine for dynamic report generation with R, solve some long-standing problems in Sweave, and combine features in other add-on packages into one package"
`https://yihui.org/knitr/`

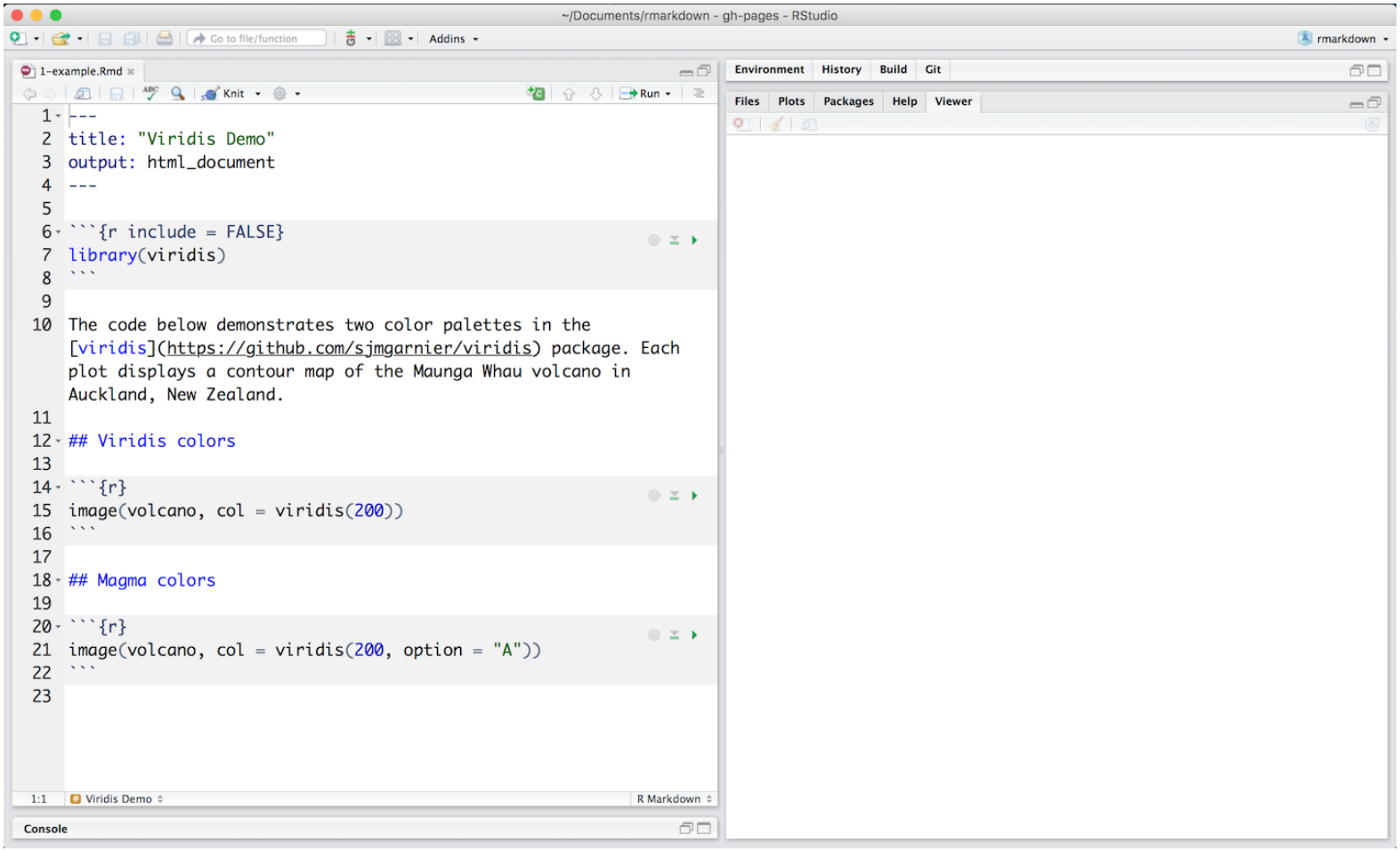
RMarkdown

RMarkdown

"When you run render, R Markdown feeds the .Rmd file to knitr, which executes all of the code chunks and creates a new markdown (.md) document which includes the code and its output.
The markdown file generated by knitr is then processed by pandoc which is responsible for creating the finished format."
`https://rmarkdown.rstudio.com`

RMarkdown

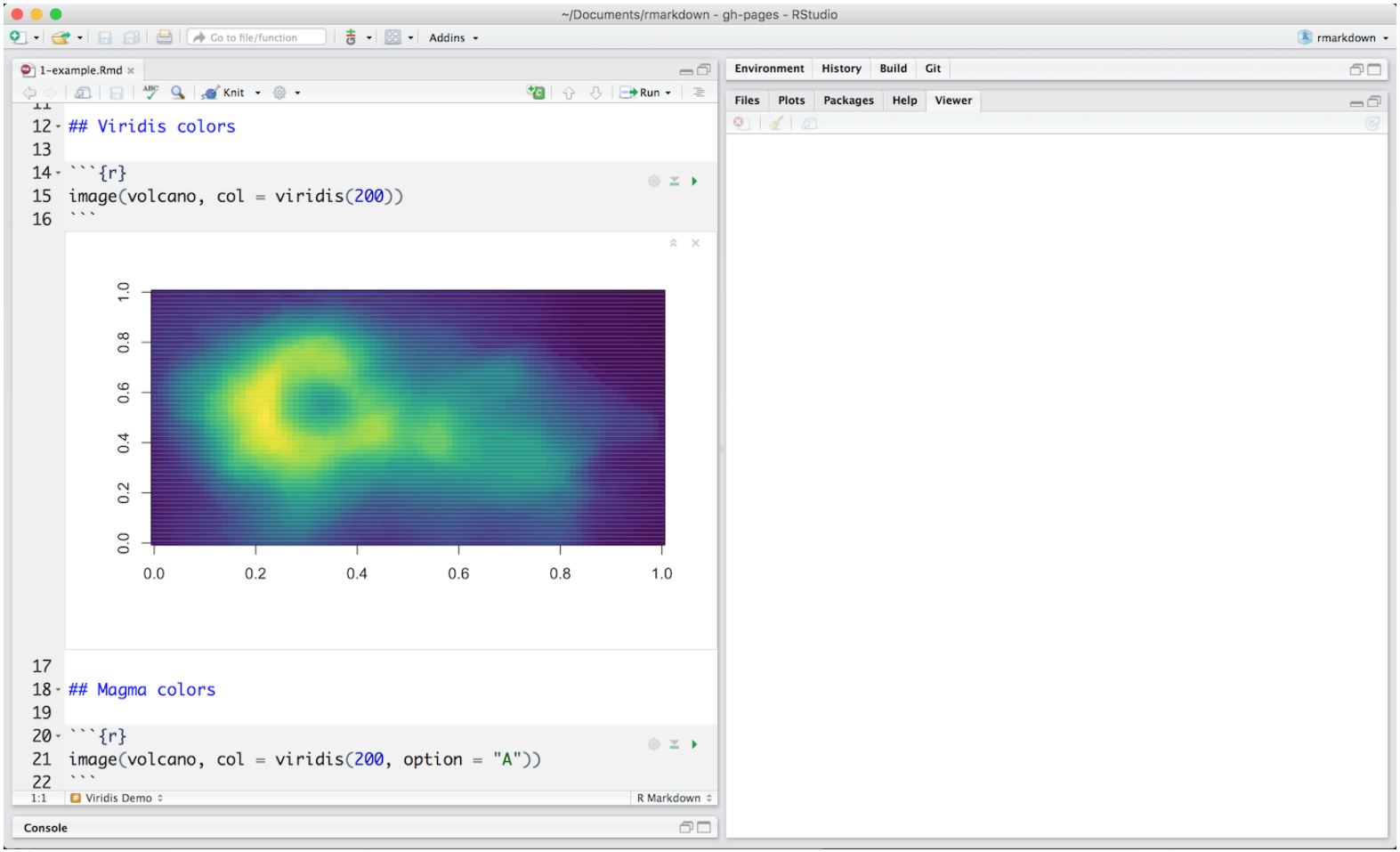
RMarkdown



Integrated into RStudio, IDE for R.

RMarkdown

R Notebooks

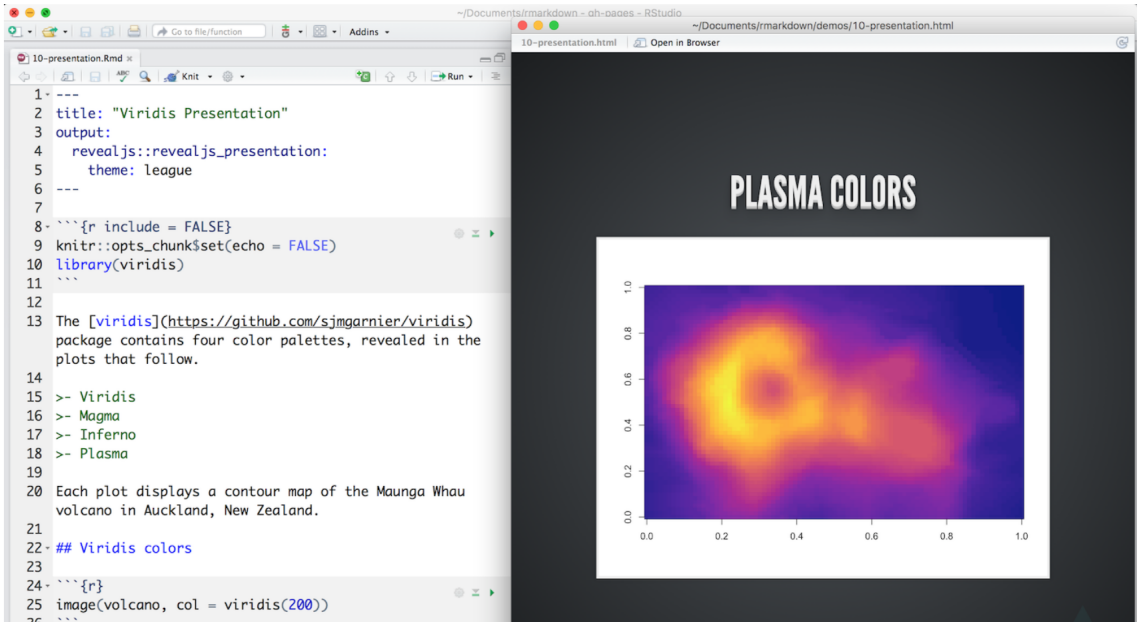


RMarkdown

R Notebooks and more...

- Markdown Basics
- Output Formats
- Notebooks
- Slide Presentations
- Dashboards
- Websites
- Interactive Documents
- Cheatsheets

file below, which is available here on RStudio Cloud.



Jupyter

Jupyter

A bit of history...

- 2011 : IPython (interactive Python shell) with notebook functionalities
- 2014 : Spin-off project called Project Jupyter
- a non-profit, open-source project maintained by a strong Community
- "Jupyter will always be 100% open-source software, free for all to use and released under the liberal terms of the modified BSD license"
- A reference to the three core programming languages supported by Jupyter (Julia, Python and R)

`https://jupyter.org/`

Jupyter

What can it do?

Jupyter

What can it do?
Everything (excepted coffee)

Jupyter

But what is it exactly ?

Jupyter

But what is it exactly ?
Web-based interactive computational environment.

Jupyter

But what is it exactly ?
Web-based interactive computational environment.

- Web-based : client/server

Jupyter

But what is it exactly ?
Web-based interactive computational environment.

- Web-based : client/server
- Interactive : notebook system

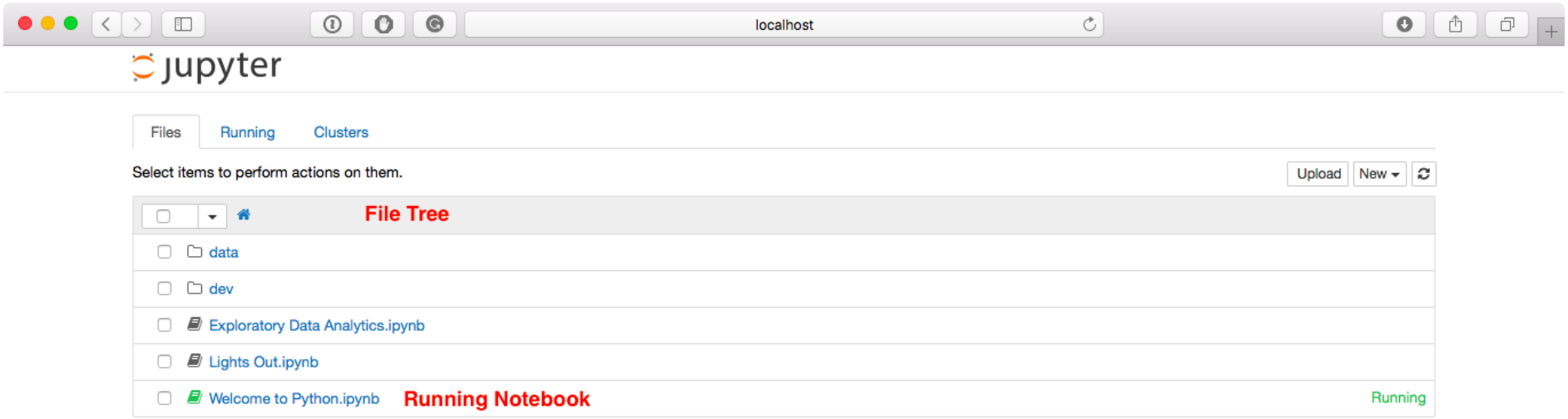
Jupyter

But what is it exactly ?
Web-based interactive computational environment.

- Web-based : client/server
- Interactive : notebook system
- Computational environment : console, many kernels available...

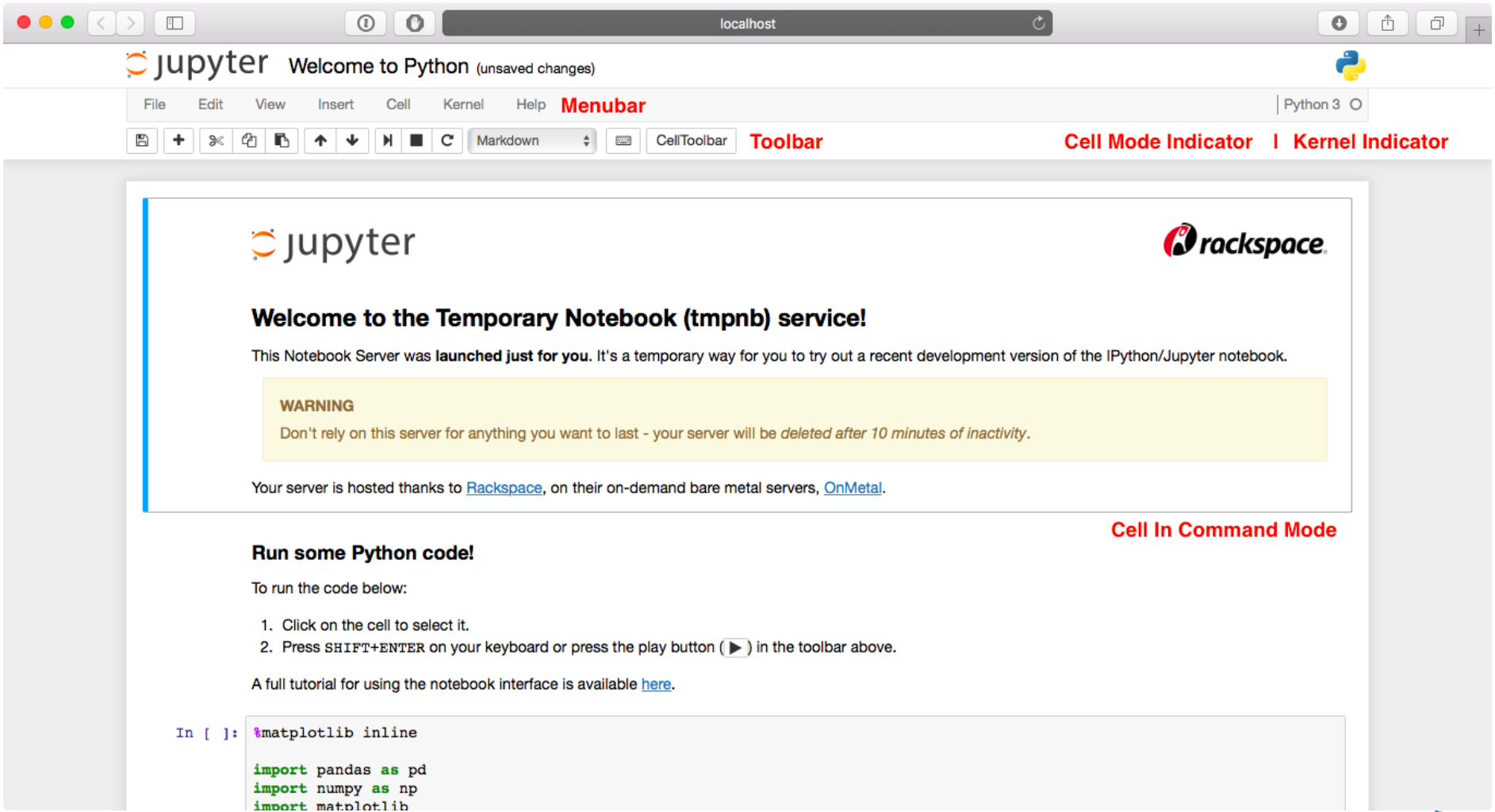
Jupyter

Dashboard



Jupyter

Notebook editor



Jupyter

Project Jupyter

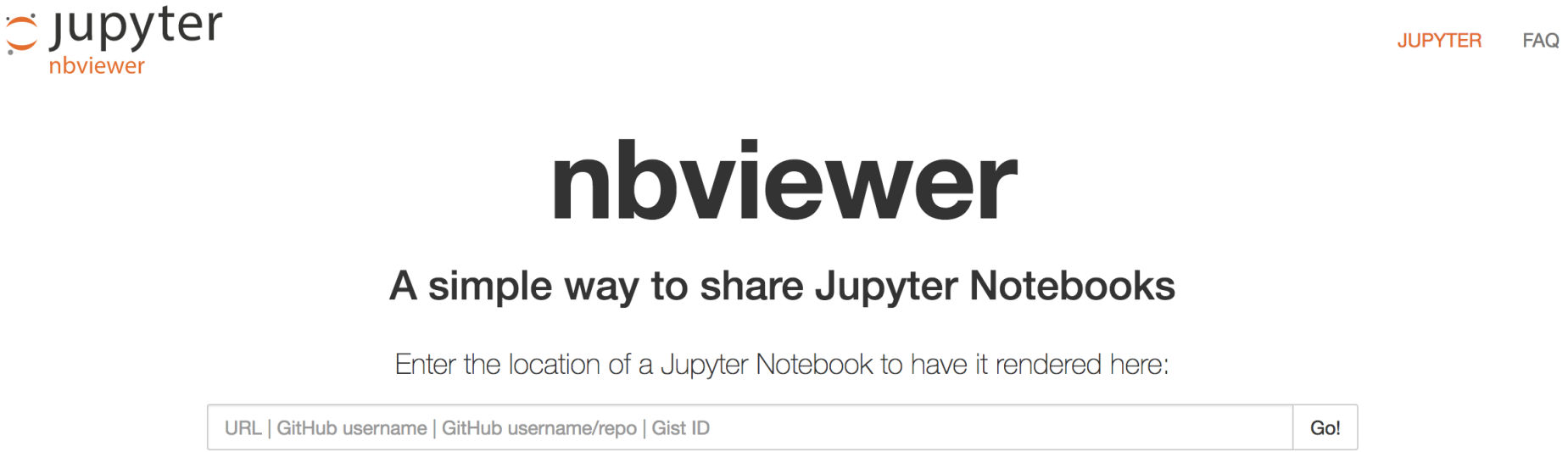
- A non-profit, open-source project maintained by a strong Community
- Adopted by the biggest in the Cloud industry (Google, Microsoft, Amazon...)
- And financed by the biggest (Google, Microsoft, EU Horizon 2020 program, Alfred P. Sloan Foundation...)

Inside the Python community (snakemake, conda...)

Integration with GitHub since 2015 (renderer)

Jupyter

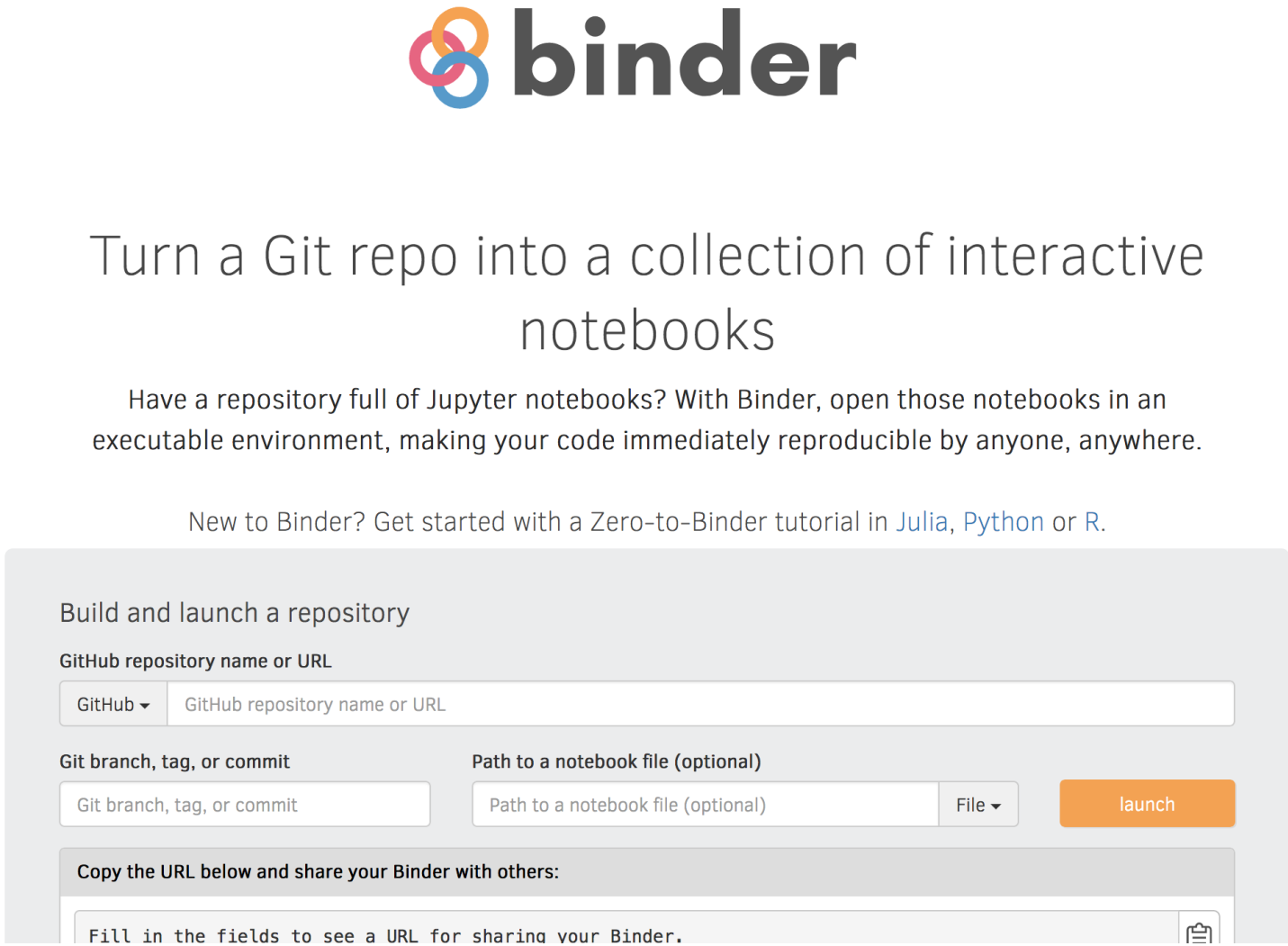
Nbviewer : a static renderer for Jupyter notebooks



`https://nbviewer.jupyter.org/`

Jupyter

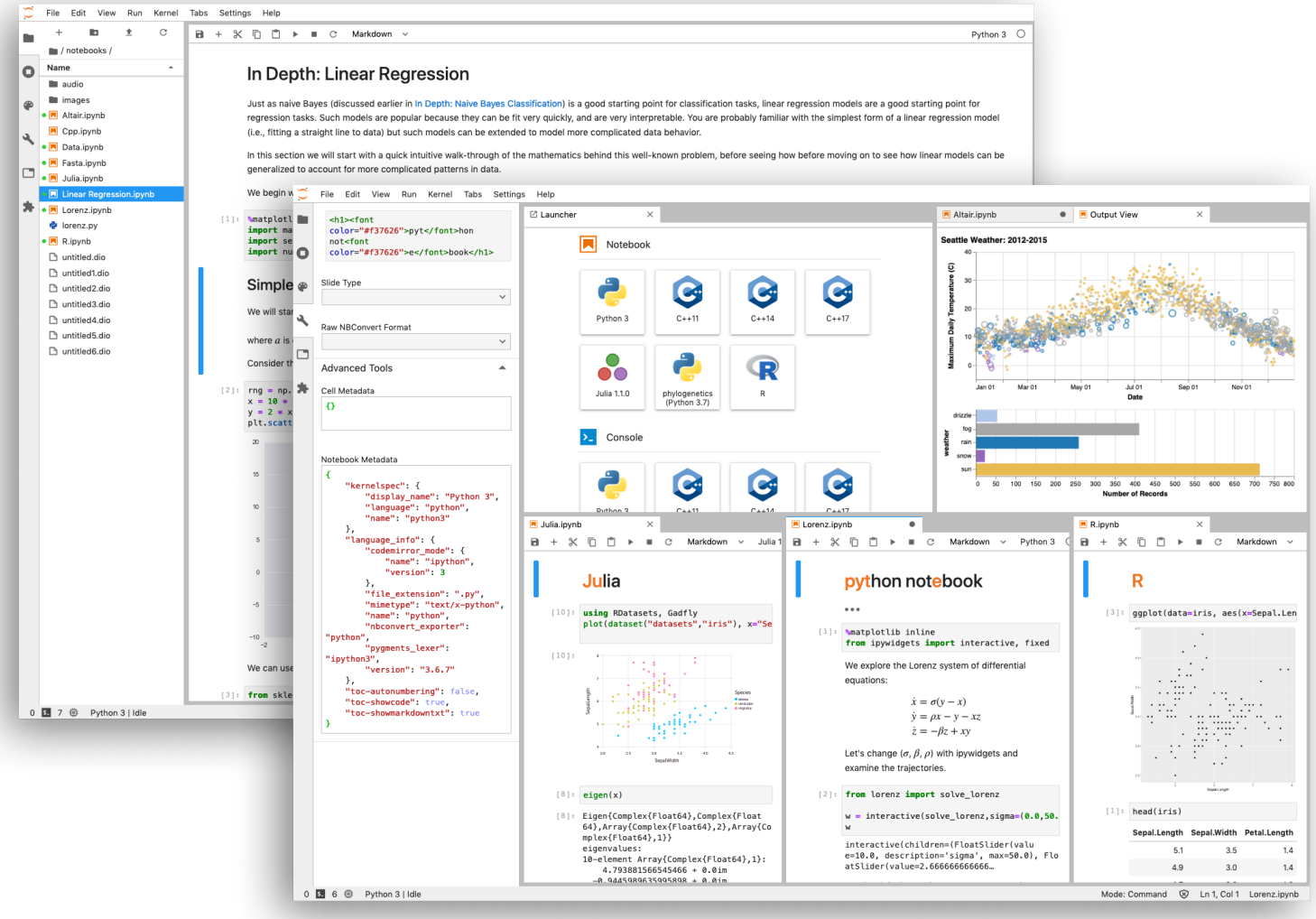
Jupyter + Docker = binder



https://mybinder.org/

Jupyter

Since June 2019 : Jupyter Lab v1.0 (now v3.0.16)



Conclusion ?

Who's the best?

Conclusion ?

Who's the best?

It depends...

Conclusion ?

Who's the best?

It depends...

- R analyses? Go for RMarkdown/RStudio
- R analyses for a publication ? Consider Jupyter with an R kernel

Conclusion ?

Who's the best?

It depends...

- R analyses? Go for RMarkdown/RStudio
- R analyses for a publication ? Consider Jupyter with an R kernel
- Python analyses ? Why do you even ask...

Practical session

Savoir FAIRe

- Markdown
- Learn the structure of an Rmd file
- Turn a script into a notebook
- Extend the notebook with new functionalities
- This afternoon: Jupyter with the IFB cluster