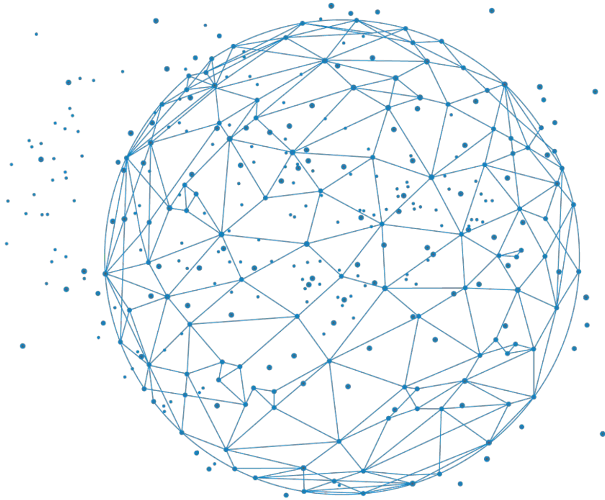


2025 June 5th



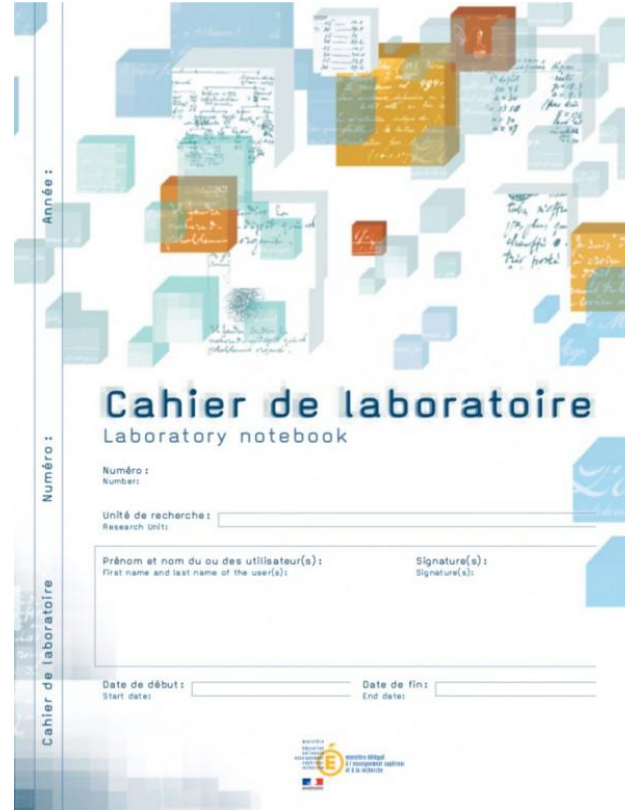
FAIR principles applied to bioinformatics

Thomas Denecker

CNRS - IFB

<https://orcid.org/0000-0003-1421-7641>

What is this?



Année : _____

Numéro : _____

Cahier de laboratoire

Laboratory notebook

Numéro : _____
Number: _____

Unité de recherche : _____
Research Unit: _____

Prénom et nom du ou des utilisateur(s) : _____
First name and last name of the user(s): _____

Signature(s) : _____
Signature(s): _____

Date de début : _____ Date de fin : _____
Start date: _____ End date: _____

Ministère de l'Enseignement Supérieur et de la Recherche
Institut Français de Bioinformatique
IFB



The laboratory notebook allows :

- day-to-day recording of the details of the work
- to report on the progress and scientific experimentation, from the idea to the conclusion
- to keep knowledge in a lab

Also very useful for drafting a patent or for proving anteriority.



A legal tool :

- Each notebook and the pages are numbered.
- On the cover page, we find on each notebook the mentions of the owner of the results.
- Each page has two parts at the bottom intended to be dated and signed: two signatures = two people, user and third party (witness),
 - ideally a third party not involved in the research work but capable of understanding it

<https://www.curie.asso.fr/-Cahier-de-laboratoire-national-.html>

<https://slideplayer.fr/slide/3817405/>



For all those who carry out research work :

- researchers,
- engineers,
- technicians,
- doctoral students,
- trainees,
- etc.

Are you using it ?



Cahier de laboratoire
Laboratory notebook

Année : _____

Numéro : _____

Unité de recherche : _____
Research Unit:

Prénom et nom du ou des utilisateur(s) : _____
First name and last name of the user(s):

Signature(s) : _____
Signature(s):

Date de début : _____ Date de fin : _____
Start date: _____ End date: _____

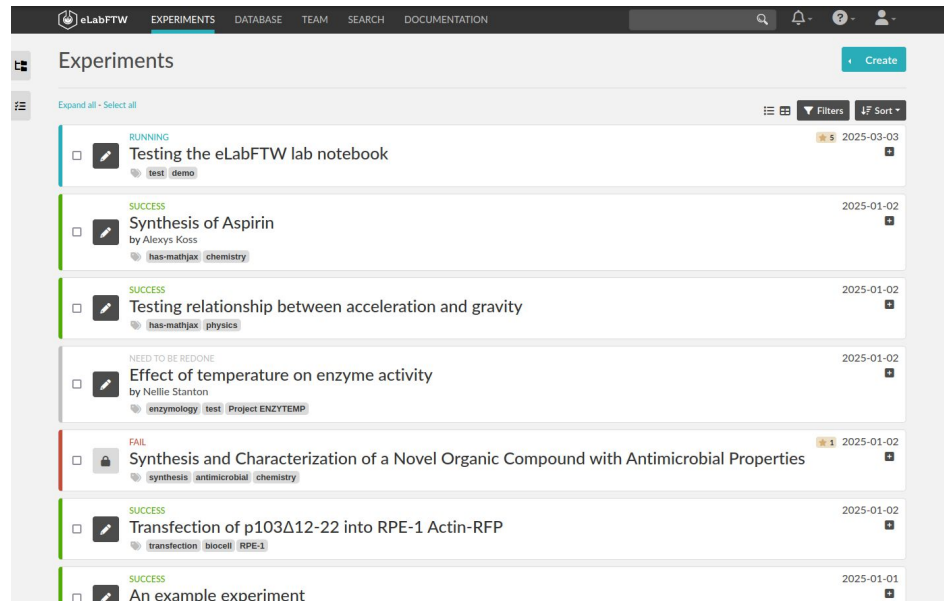
Ministère de l'Enseignement Supérieur et de la Recherche
Ministère de la Santé
Ministère de l'Industrie

Modern LN since 2009 (C.U.R.I.E.
Network)

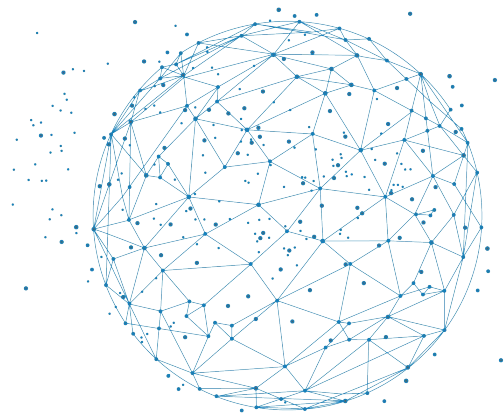
But less and less adapted to recent
evolutions of our work

- Increased data quantity
- Change in the nature of data
- Dematerialization
- Security

We need an electronic tool for
individual traceability.



What about programming?





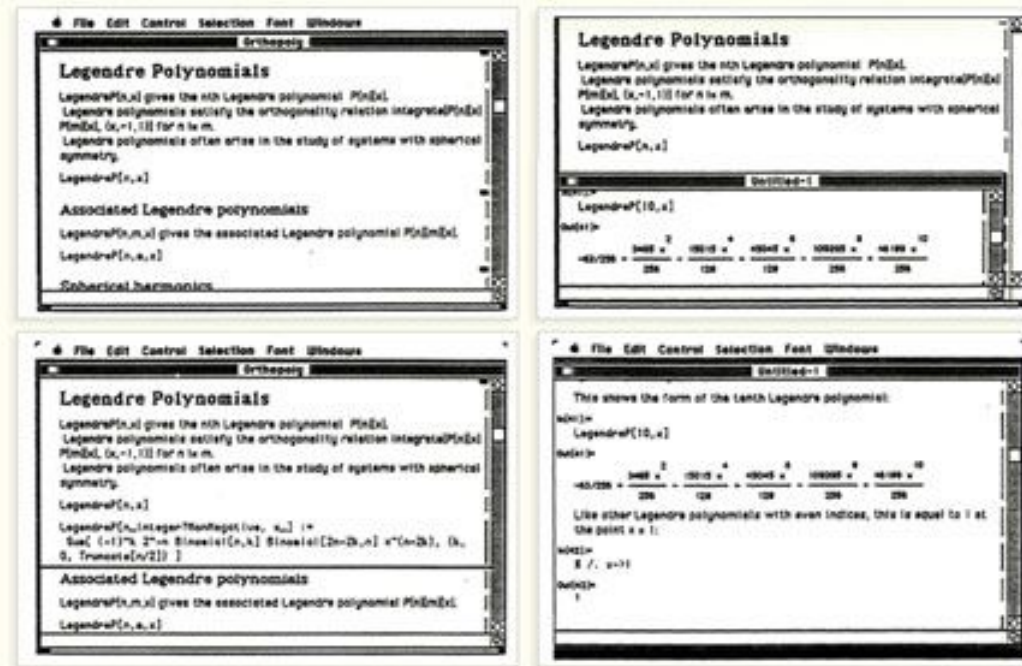
Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.

— Donald E. Knuth, Literate Programming, 1984



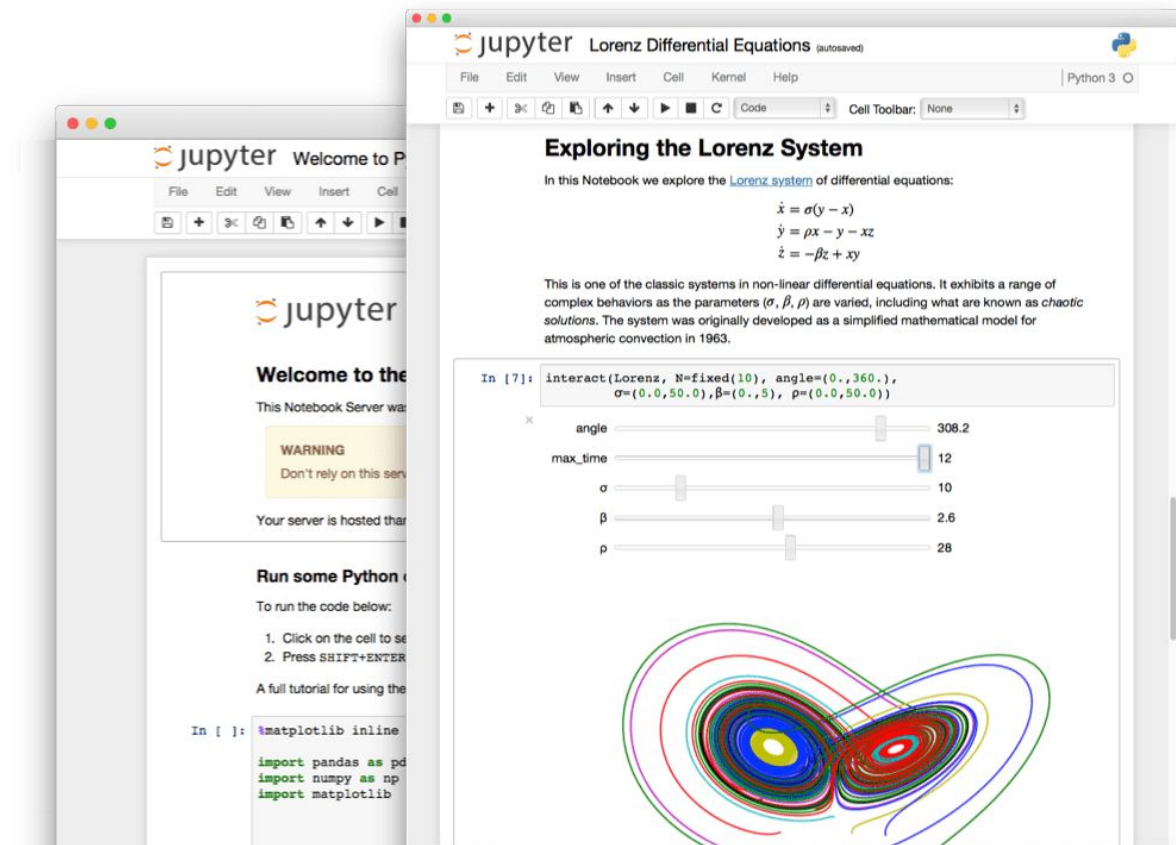
A literate computing environment is one that allows users not only to execute commands interactively, but also to store in a literate document the results of these commands along with figures and free-form text.

- Millman KJ and Perez F (2014)



Wolfram Mathematica notebook (1987)

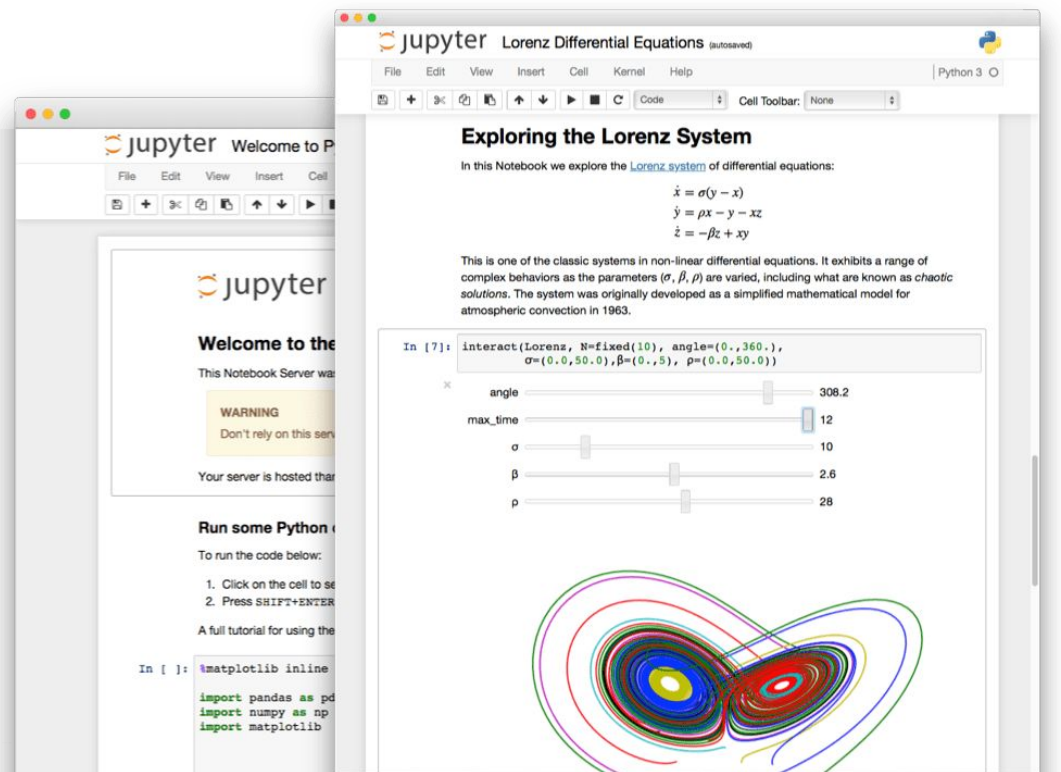
What does it look like
now ?



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

In one file:

- Explanations
- Code
- Results
- Graphs and plots





Why using literate programming frameworks ?

Use cases:

- Labbook
- Day to day analyses
- Analysis reports
- Writing scientific article

Example of an article entirely written using a notebook



File (on a repository)

colomoto / colomoto-docker

Code Issues Pull requests Actions Security Insights

for-next - colomoto-docker / usecases / Usecase - Mutations enabling tumour invasion.ipynb

Go to file

pauleve update notebooks with new ginsim graphics Latest commit 14985c2 on 24 Apr History

253 lines (2513 sloc) 422 KB

Prediction of Mutations to Control Pathways Enabling Tumour Cell Invasion with the CoLoMoTo Interactive Notebook (Tutorial)

Authors: Nicolas Levy^{1,2}, Aurélien Naldi³, Céline Hernandez³, Gautier Stoll^{4,5}, Denis Thieffry³, Andrei Zinovyev^{6,11}, Laurence Calzone^{6,11}, Loïc Paulevé^{1,7}

¹ LRI UMR 8623, Université Paris-Sud, CNRS, Université Paris-Saclay, Orsay, France; ² École Normale Supérieure de Lyon, France; ³ Computational Systems Biology team, Institut de Biologie de l'École Normale Supérieure, CNRS UMR8197, INSERM U1024, École Normale Supérieure, PSL Université, Paris, France; ⁴ Université Paris Descartes/Paris V, Sorbonne Paris Cité, Paris, France; ⁵ Équipe 11 labellisée Ligue Nationale contre le Cancer, Centre de Recherche des Cordeliers, Paris, France; ⁶ Institut National de la Santé et de la Recherche Médicale, U1138; Paris, France; ⁷ Université Pierre et Marie Curie, Paris, France; ⁸ Metabolomics and Cell Biology Platforms, Gustave Roussy Cancer Campus, Villejuif, France; ⁹ Institut Curie, PSL Research University, Paris, France; ¹⁰ INSERM U900, Paris, France; ¹¹ MINS ParisTech, PSL Research University, CIBO-Centre for Computational Biology, Paris, France

Published article

Frontiers | Prediction of Mutations to Control Pathways Enabling Tumor Cell Invasion with the CoLoMoTo Interactive Notebook (Tutorial)

Front. Physiol. 06 July 2018 | <https://doi.org/10.3389/fphys.2018.00787>

Prediction of Mutations to Control Pathways Enabling Tumor Cell Invasion with the CoLoMoTo Interactive Notebook (Tutorial)

Nicolas Levy^{1,2}, Aurélien Naldi³, Céline Hernandez³, Gautier Stoll^{4,5,6,7,8}, Denis Thieffry³, Andrei Zinovyev^{9,10,11,12}, Laurence Calzone^{9,10,11} and Loïc Paulevé¹

¹LRI UMR 8623, Centre National de la Recherche Scientifique, Université Paris-Sud, Université Paris-Saclay, Orsay, France
²École Normale Supérieure de Lyon, Lyon, France
³Computational Systems Biology Team, Institut de Biologie de l'École Normale Supérieure, Centre National de la Recherche Scientifique UMR8197, INSERM U1024, École Normale Supérieure, PSL Université, Paris, France
⁴Université Paris Descartes, Sorbonne Paris Cité, Paris, France
⁵Équipe 11 Labellisée Ligue Nationale contre le Cancer, Centre de Recherche des Cordeliers, Paris, France
⁶Institut National de la Santé et de la Recherche Médicale, Paris, France
⁷Université Pierre et Marie Curie, Paris, France
⁸Metabolomics and Cell Biology Platforms, Gustave Roussy Cancer Campus, Villejuif, France
⁹Institut Curie, PSL Research University, Paris, France
¹⁰INSERM U900, Paris, France
¹¹MINS ParisTech, PSL Research University, CIBO-Centre for Computational Biology, Paris, France
¹²Lobachevsky University, Nizhni Novgorod, Russia

Executable file

Jupyter Notebook Viewer

Prediction of Mutations to Control Pathways Enabling Tumour Cell Invasion with the CoLoMoTo Interactive Notebook (Tutorial)

Authors: Nicolas Levy^{1,2}, Aurélien Naldi³, Céline Hernandez³, Gautier Stoll^{4,5}, Denis Thieffry³, Andrei Zinovyev^{6,11}, Laurence Calzone^{6,11}, Loïc Paulevé^{1,7}

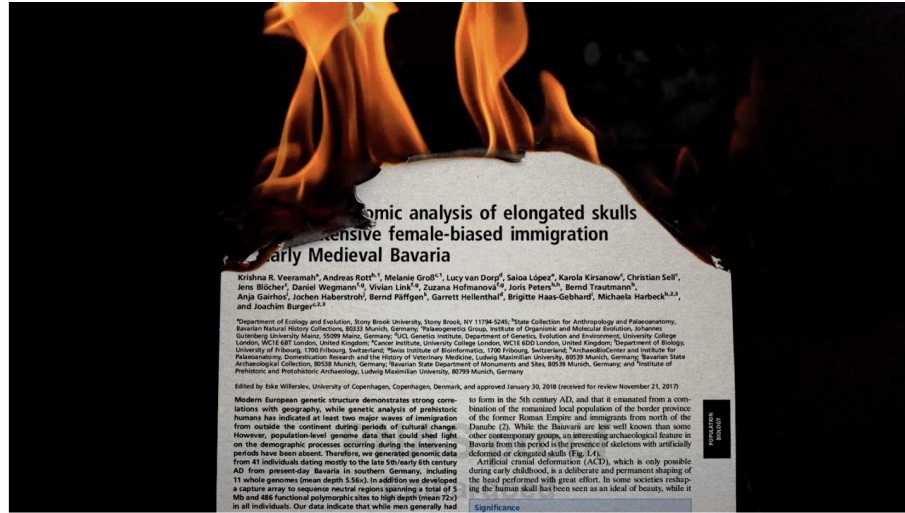
¹ LRI UMR 8623, Université Paris-Sud, CNRS, Université Paris-Saclay, Orsay, France; ² École Normale Supérieure de Lyon, France; ³ Computational Systems Biology team, Institut de Biologie de l'École Normale Supérieure, CNRS UMR8197, INSERM U1024, École Normale Supérieure, PSL Université, Paris, France; ⁴ Université Paris Descartes/Paris V, Sorbonne Paris Cité, Paris, France; ⁵ Équipe 11 labellisée Ligue Nationale contre le Cancer, Centre de Recherche des Cordeliers, Paris, France; ⁶ Institut National de la Santé et de la Recherche Médicale, U1138; Paris, France; ⁷ Université Pierre et Marie Curie, Paris, France; ⁸ Metabolomics and Cell Biology Platforms, Gustave Roussy Cancer Campus, Villejuif, France; ⁹ Institut Curie, PSL Research University, Paris, France; ¹⁰ INSERM U900, Paris, France; ¹¹ MINS ParisTech, PSL Research University, CIBO-Centre for Computational Biology, Paris, France

Abstract

Boolean and multi-valued logical formalisms are increasingly used to model complex cellular networks. To ease the development and analysis of logical models, a series of software tools have been proposed, often with specific assets. However, combining these tools typically implies a series of cumbersome software installation and model conversion steps. In this respect, the CoLoMoTo Interactive Notebook provides a joint distribution of several logical modelling software tools, along with an interactive web Python interface easing the chaining of complementary analyses.

Our computational workflow combines (1) the importation of a Ginsim model and its display, (2) its format conversion using the Java library BioLOM, (3) the

DOI:10.3389/fphys.2018.00787



PNAS / Richard Georg / Getty / The Atlantic

SCIENCE

THE SCIENTIFIC PAPER IS OBSOLETE

Here's what's next.

By James Somers

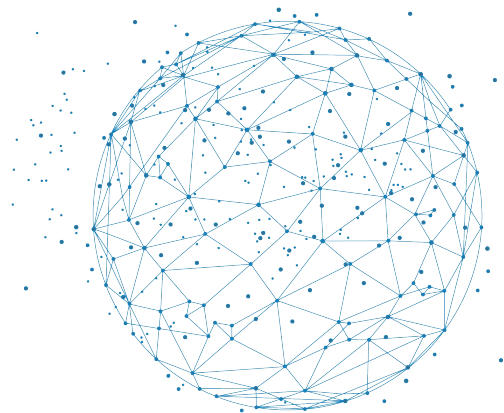
APRIL 5, 2018

SHARE   

THE SCIENTIFIC paper—the actual form of it—was one of the enabling inventions of modernity. Before it was developed in the 1600s, results

We need to be careful

Markup / 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.

Example in HTML

```
<h1>Heading</h1>
<h2>Sub-heading</h2>
<a href="www.webpage.com">Link</a>
<ul>
  <li>List-item1</li>
  <li>List-item2</li>
  <li>List-item3</li>
</ul>
```

Heading

Sub-heading

Link

- List-item1
- List-item2
- List-item3

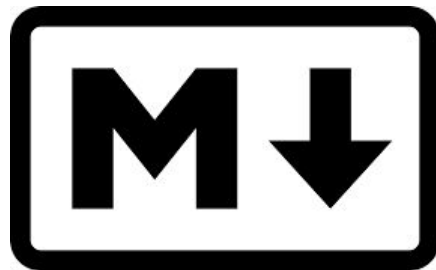
Markdown is a Lightweight markup language

Designed to be :

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

From GitHub's help page

<https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax>



Example in markdown

```
# Heading
## Sub-heading
### Another deeper heading

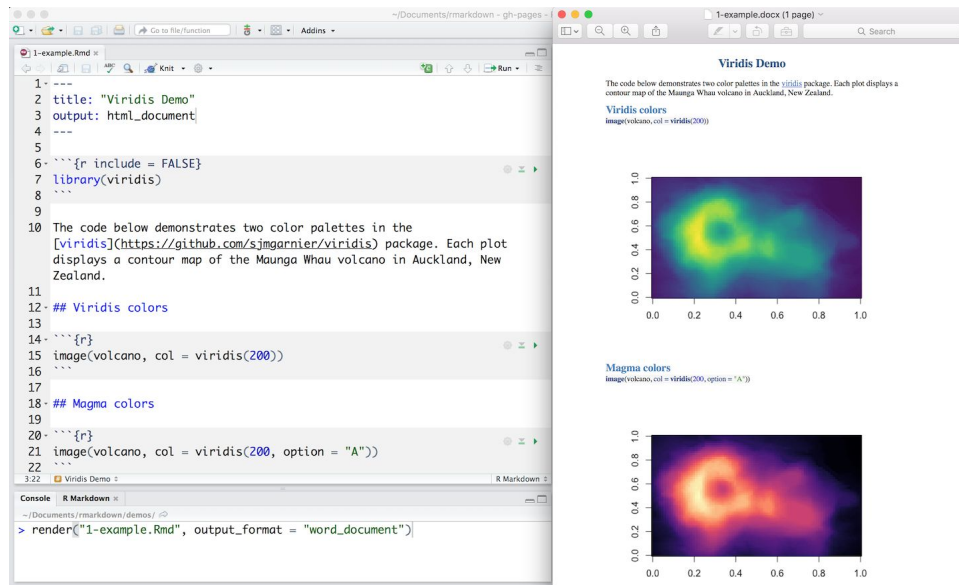
A [link](http://example.com).

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

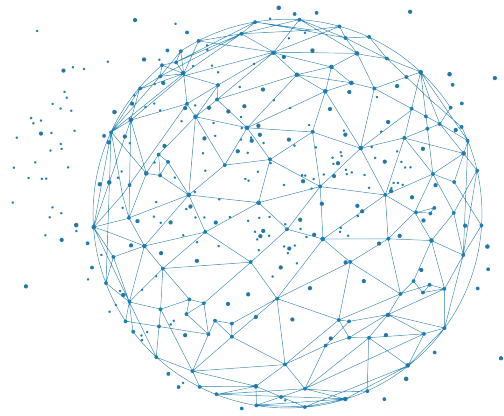
Bullet list:
* apples
* oranges
* pears
```

But how is this useful for literate programming?

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



Best practices



Notebook and Open science : toward more FAIR play

Mariannig Le Béché^{1*}, Célya Gruson Daniel^{2,3}, Clémence Lascombes³, Émilien Schultz⁴

1 CREM, Université de Lorraine

2 Inno³

3 COSTECH – Université de Technologie de Compiègne

4 CREST – GENES/IPP

*Corresponding author: Mariannig Le Béché mariannig.le-beche@lorraine.fr

Abstract

Notebooks are now commonly used in digital research practices. Despite their increasing ubiquity, the characteristics, roles, and uses associated with notebooks have seldom been studied from a social science perspective. In this article, we present an overview of the available empirical work on notebooks in order to describe existing practices, typologies crafted to grasp their diversity, and their limitations when used in data analysis workflows. Following this review, which highlights a focus of studies on interactive computational notebooks specifically within data science rather than research practices in academic contexts, we discuss the role of notebooks as a vector and lever for the FAIR (Findable, Accessible, Interoperable, Reusable) principles associated with open science.

keywords

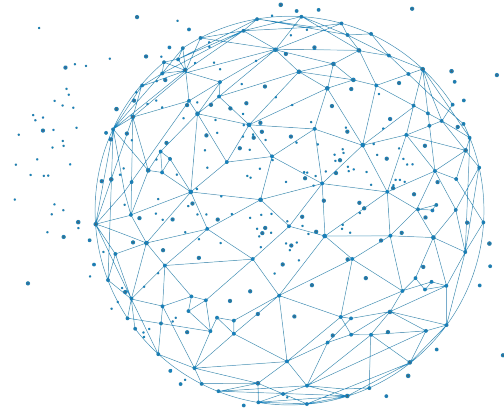
notebook, literate programming, jupyter, open science, FAIR

<https://hal.science/hal-04549986/document>

General categories	Best practices
Make your analysis traceable and reproducible	• Use a version control system to manage project dependency • Manage project dependencies • Provide applications without third-party dependencies • Put imports at the beginning of the file • Ensure that the entire code functions correctly, not just the modified part
Write quality code (i.e. code that can be easily shared and reused)	• Structure your code into modules (abstract the code into functions and place them in a dedicated module; place dependencies at the beginning of the notebook) • Test your code • Name your notebooks consistently • Respect standards • Use relative paths • Define requirements
Exploit the paradigm of literate programming	• Document your code for yourself and others • Use Markdown headings to structure your notebook
Keep your notebook clear and concise	• Keep your notebook clear • Keep your notebook concise
Differentiate between artefacts produced during development and production	• Differentiate between artefacts produced during development and production
Adopt open distribution	• Make your notebook available • Make your data available

Table 1: Catalogue of good practices specific to the use of notebooks extracted from the literature review

Limites





In this article, the authors highlight several limitations:

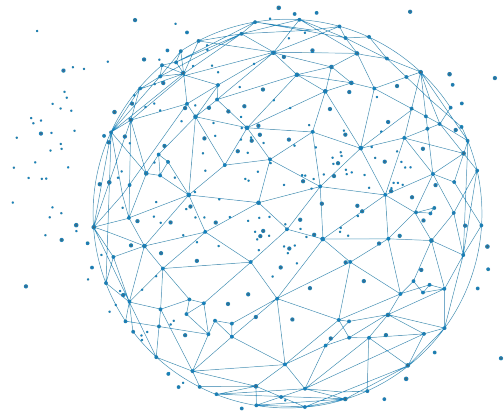
Reproducibility

- Lack of documentation
- Lack of consistency in cell execution
- Version tracking
- The quality of the code is often low

Interface dependent

Limited for large data sets

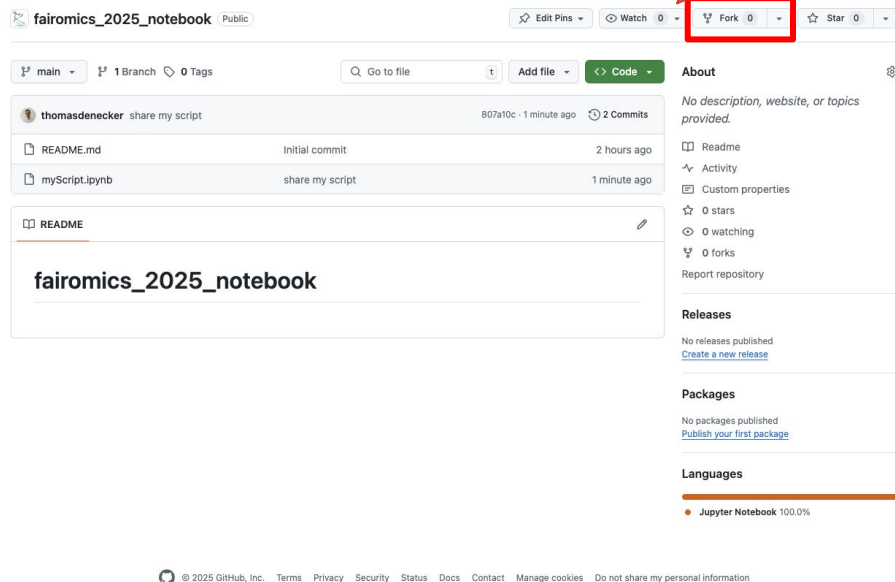
Let's try



Step 1 - Fork the project

A fork is a personal copy of someone else's repository on your GitHub/Gitlab account.

It allows you to freely experiment with changes without affecting the original project.

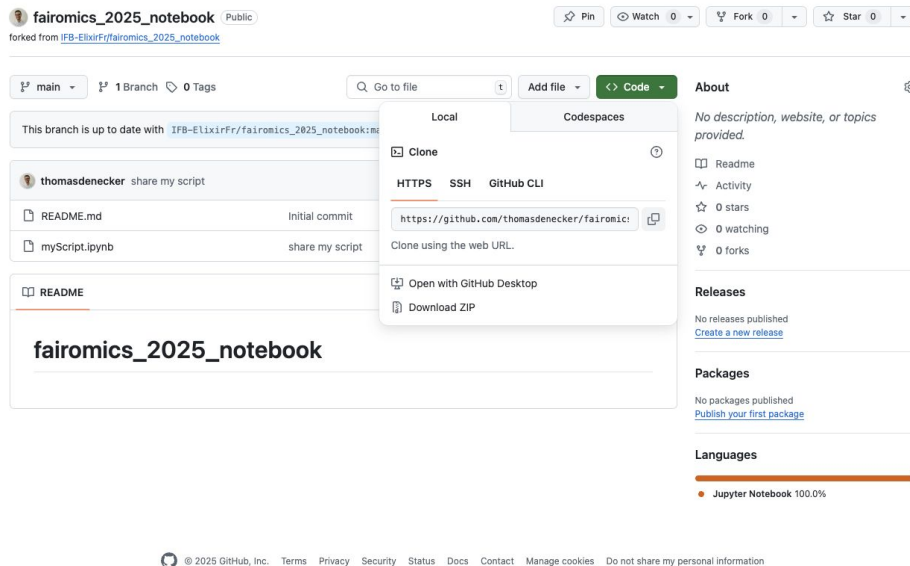


https://github.com/IFB-ElixirFr/fairomics_2025_notebook

Step 2 - Clone the project

`git clone` is a command used to download a remote Git repository to your local machine.

- It:
- Copies the entire project history
 - Creates a local working directory
 - Sets up a link to the remote (usually called origin)



Let's improve this notebook

```
myScript.ipynb
+
Code
git
Bash

[ ]: ORDERS=("order1" "order2" "order3")

mkdir -p fries burgers served

for order in "${ORDERS[@]"; do
    echo "=== Processing $order ==="

    echo "Cutting potatoes for $order..."
    sleep 1
    echo "Potatoes cut for $order" > "fries/cut_$order.txt"

    echo "Frying fries for $order..."
    sleep 2
    echo "Fries cooked for $order" > "fries/fried_$order.txt"

    echo "Shaping burger patty for $order..."
    sleep 1
    echo "Raw patty shaped for $order" > "burgers/raw_patty_$order.txt"

    echo "Cooking patty for $order..."
    sleep 2
    echo "Patty cooked for $order" > "burgers/cooked_patty_$order.txt"

    echo "Assembling burger for $order..."
    sleep 1
    echo "Burger assembled for $order" > "burgers/burger_$order.txt"

    echo "Serving burger and fries for $order..."
    sleep 1
    echo "Burger and fries served for $order" > "served/$order.txt"

    echo "=== $order is ready! ==="
    echo
done

echo "All orders are complete."
```



Don't forget to push your modifications ;)

In case of fire



1. git commit

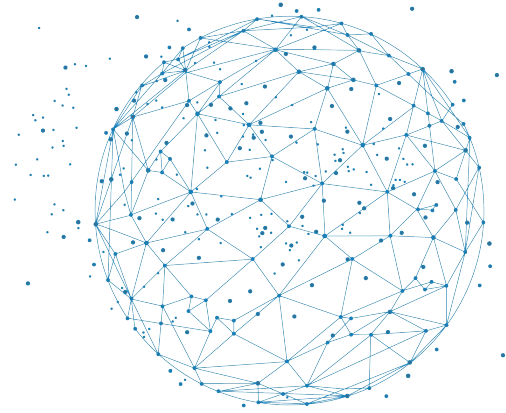


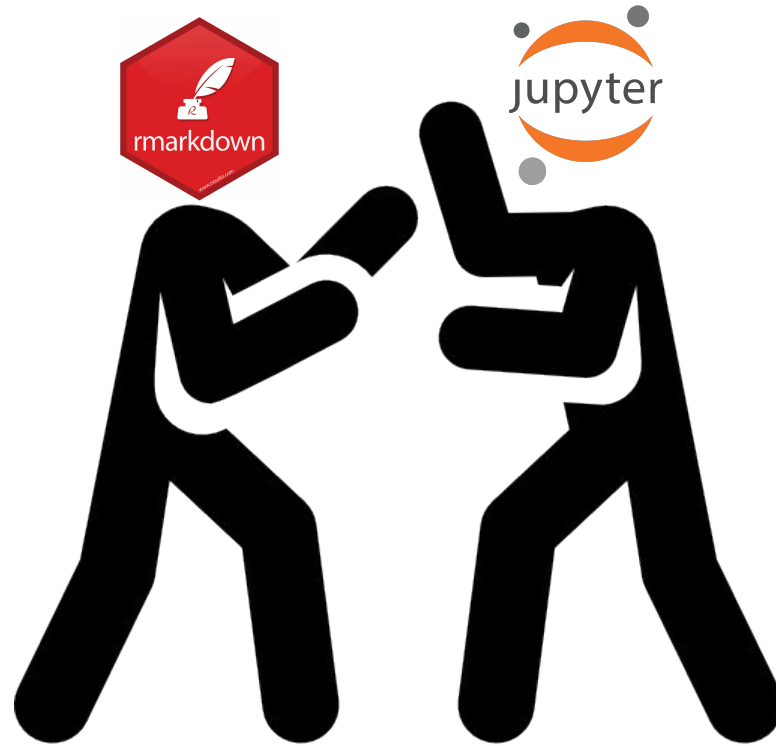
2. git push

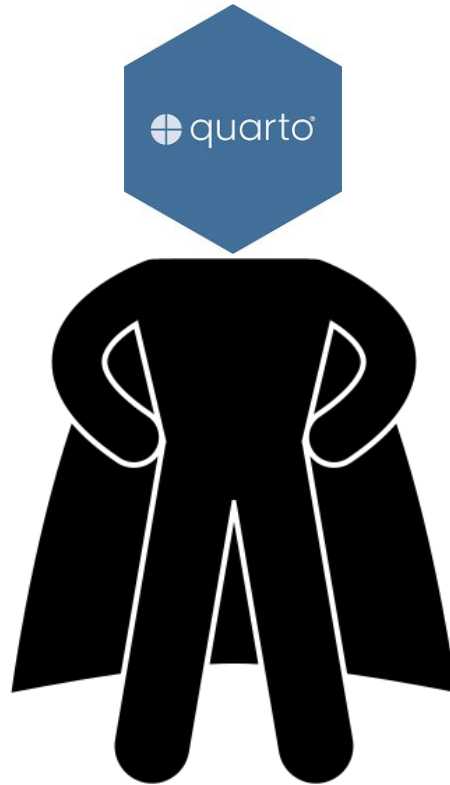


3. leave building

Alternative to Jupyter notebooks







At the beginning, there was nothing. Then came Sweave.

Vol. 2/3, December 2002

28

Sweave, Part I: Mixing R and L^AT_EX

A short introduction to the Sweave file format and corresponding R functions

by Friedrich Leisch

This is the first article in a two part mini series on Sweave (Leisch, 2002), a tool that allows to embed the R code for complete data analyses in L^AT_EX documents. In this issue we will introduce the Sweave file format and R functions to process it, and demonstrate how to use Sweave as a reporting tool for literate statistical practice (Rossini, 2001). The companion article scheduled for the next issue of R News will concentrate on how to use files in Sweave format to write primers or manuals for R packages that can be automatically checked for syntax errors in the code or inconsistencies between examples and implementation.

The traditional way of writing a report as part of a statistical data analysis project uses two separate steps: First, the data are analyzed, and afterwards the results of the analysis (numbers, graphs, ...) are used as the basis for a written report. In larger projects the two steps may be repeated alternately, but the basic procedure remains the same. R supports this in a number of ways: graphs can be saved as EPS, PDF, or WMF which in turn can be included in L^AT_EX or Word documents. L^AT_EX tables can be created by specifying the columns and row separators in `write.table()` or using the package `stable`. The basic paradigm is to write the report around the results of the analysis.

The purpose of Sweave is to create dynamic reports, which can be updated automatically if data or analysis change. Instead of inserting a preformatted graph or table into the report, the master document contains the R code necessary to obtain it. When run through R, all data analysis output (tables, graphs, ...) is created on the fly and inserted into a final L^AT_EX document. The report can be automatically updated if data or analysis change, which allows for truly reproducible research.

A small example

Sweave source files are regular noweb files (Ramos, 1998) with some additional syntax that allows control over the final output. Noweb is a simple literate programming tool which allows to combine program source code and the corresponding documentation into a single file. These consist of a sequence of code and documentation segments, called chunks. Different command line programs are used to extract the code ("tangle") or typeset documentation to

gether with the code ("noweb"). A small Sweave file is shown in Figure 1, which contains four code chunks embedded in a simple L^AT_EX document. "`<<->>`" at the beginning of a line marks the start of a code chunk, while a "`@`" at the beginning of a line marks the start of a documentation chunk. Sweave translates this into a regular L^AT_EX document, which in turn can be compiled by `latex` as Figure 2.

The code chunks

The main work of Sweave is done on the code chunks. All code chunks are evaluated by R in the order they appear in the document¹. Within the double angle brackets we can specify options that control how the code and the corresponding output are rendered in the final document. The first code chunk (lines 5-8 in Figure 1) declares that neither the R code (`<<echo=false`) nor output (`results=hide`) shall be included. The purpose of this chunk is to initialize R by loading packages and data, we want to hide these technical details from the reader.

Let us skip the text in lines 10-19 for the moment and go directly to the next code chunk in lines 20-22. It uses the default settings for all options (nothing is specified within the double angle brackets): both input and output are shown to the user (see Figure 2); the chunk is rendered such that it emulates the R console when the code is typed at the prompt. All input and output are automatically encapsulated in verbatim-like environments.

The next code chunk can be found at lines 30-31. It uses the package `stable` to pretty-print the coefficient matrix of the linear regression model. By specifying `result=latex` we tell Sweave that the output of this code chunk is regular L^AT_EX code and hence needs no protection by a verbatim environment. The last code chunk in lines 36-38 is marked as a figure chunk (`fig=true`) such that Sweave creates EPS and PDF files corresponding to the plot created by the commands in the chunk. Furthermore, an `\includegraphics()` statement is inserted into the L^AT_EX file. Options `width` and `height` are passed to R's graphics device and determine the size of the figure in the EPS and PDF files.

In line 28 we use `\SweaveOpts(echo=false)` to modify the default for option `echo` to the value of `false` for all code chunks following, hence the code for the last two chunks is not shown in Figure 2. It has exactly the same effect as if we had included `<<echo=false` within the double angle brackets of the two chunks.

```
\documentclass[a4paper]{article}

\begin{document}

5 <<echo=false,results=hide>>
  library(lattice)
  library(table)
  data(cats, package="MASS")
  @
10 \section*{The Cats Data}

  Consider the \texttt{cats} regression example from Venables \& Ripley
  (1997). The data frame contains measurements of heart and body weight
15 of \Sexpr{ncol(cats)} cats (\Sexpr{sum(cats$sex=="F")} female,
  \Sexpr{sum(cats$sex=="M")} male).

  A linear regression model of heart weight by sex and gender can be
  fitted in R using the command
20 <<>>
  lm1 = lm(Hwt~Bwt*Sex, data=cats)
  lm1
  @
  Tests for significance of the coefficients are shown in
  Table \ref{tab:coef}, a scatter plot including the regression lines is
  shown in Figure \ref{fig:cats}.

  \SweaveOpts(echo=false)

30 <<results=latex>>
  xtable(lm1, caption="Linear regression model for cats data.", label="tab:coef")
  @

\begin{figure}
  \centering
  <<fig=true,width=12,height=6>>
  \set{col.whitebg()}
  print(xplot(Hwt~Bwt|Sex, data=cats, type=c("p", "r"))
40 @
  \caption{The cats data from package MASS.}
  \label{fig:cats}
\end{figure}

\end{document}
```

Figure 1: A minimal Sweave file: `example.Snw`.

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	2.9813	1.8428	1.62	0.1080
Bwt	2.6364	0.7759	3.40	0.0009
SexM	-4.1654	2.0618	-2.02	0.0453
Bwt:SexM	1.6763	0.8373	2.00	0.0472

Table 1: Linear regression model for cats data.

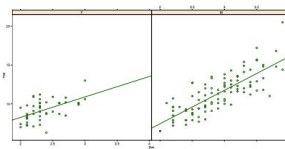


Figure 1: The cats data from package MASS.

The Cats Data

Consider the `cats` regression example from Venables & Ripley (1997). The data frame contains measurements of heart and body weight of 144 cats (47 female, 97 male).

A linear regression model of heart weight by sex and gender can be fitted in R using the command

```
> lm1 = lm(Hwt ~ Bwt * Sex, data = cats)
> lm1
```

Call:

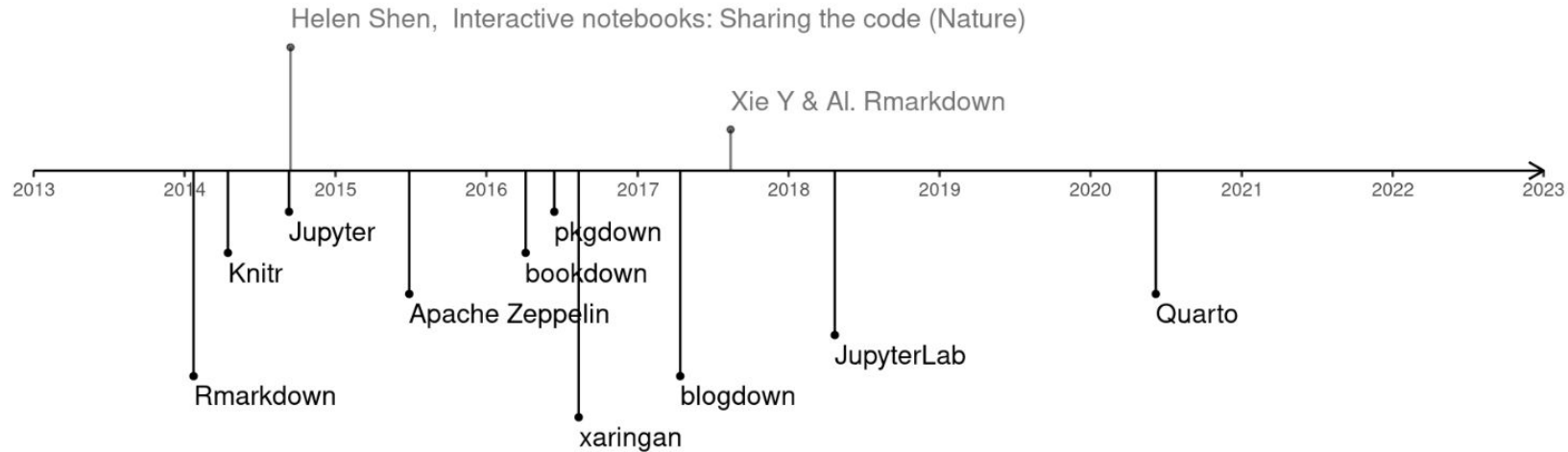
```
lm(formula = Hwt ~ Bwt * Sex, data = cats)
```

```
Coefficients:
(Intercept)      Bwt      SexM      Bwt:SexM
      2.981      2.636      -4.165      1.676
```

Tests for significance of the coefficients are shown in Table 1, a scatter plot including the regression lines is shown in Figure 1.

Figure 2: The final document is created by running `latex` on the intermediate file `'example.tex'` created by Sweave (`'example.Snw'`).

And people saw that the path would be long...



<https://camembr.quarto.pub/hello-quarto/#/les-packages>



”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/>

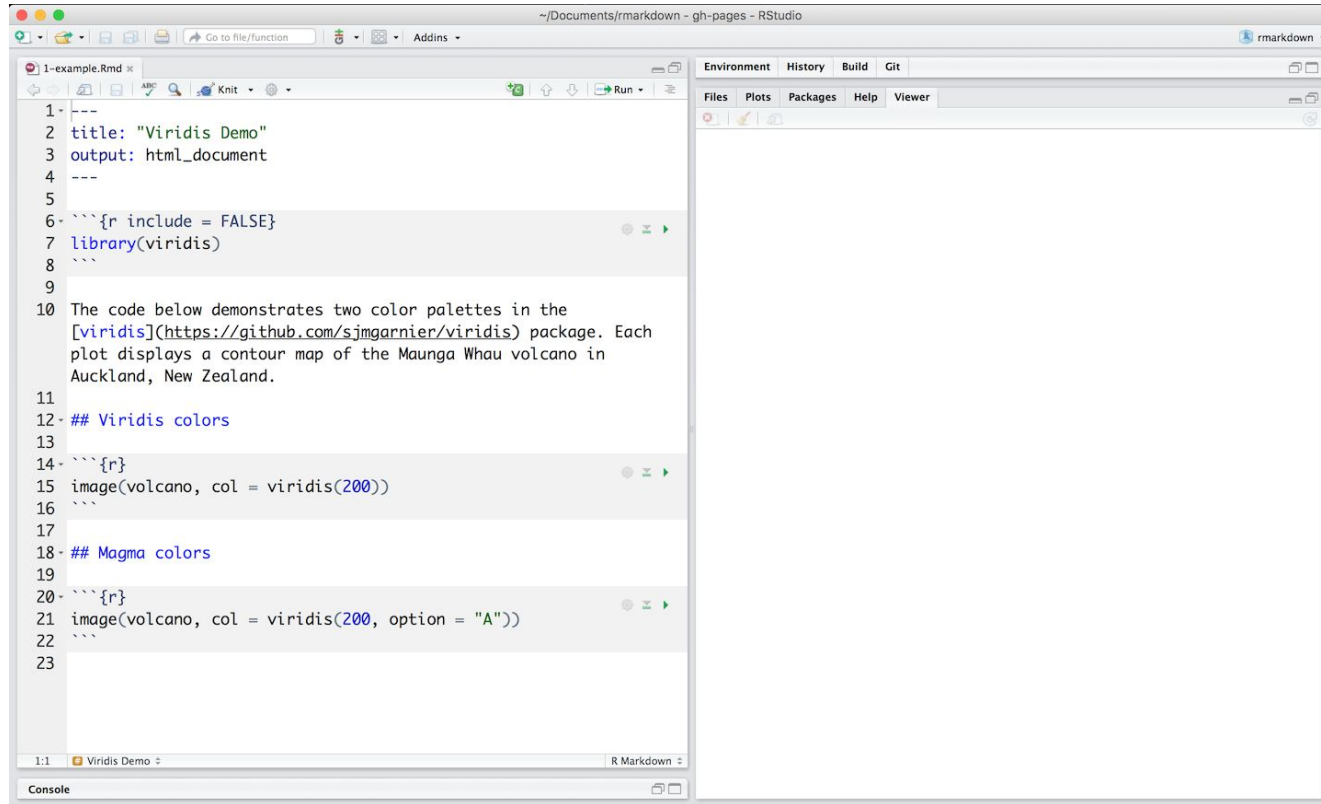


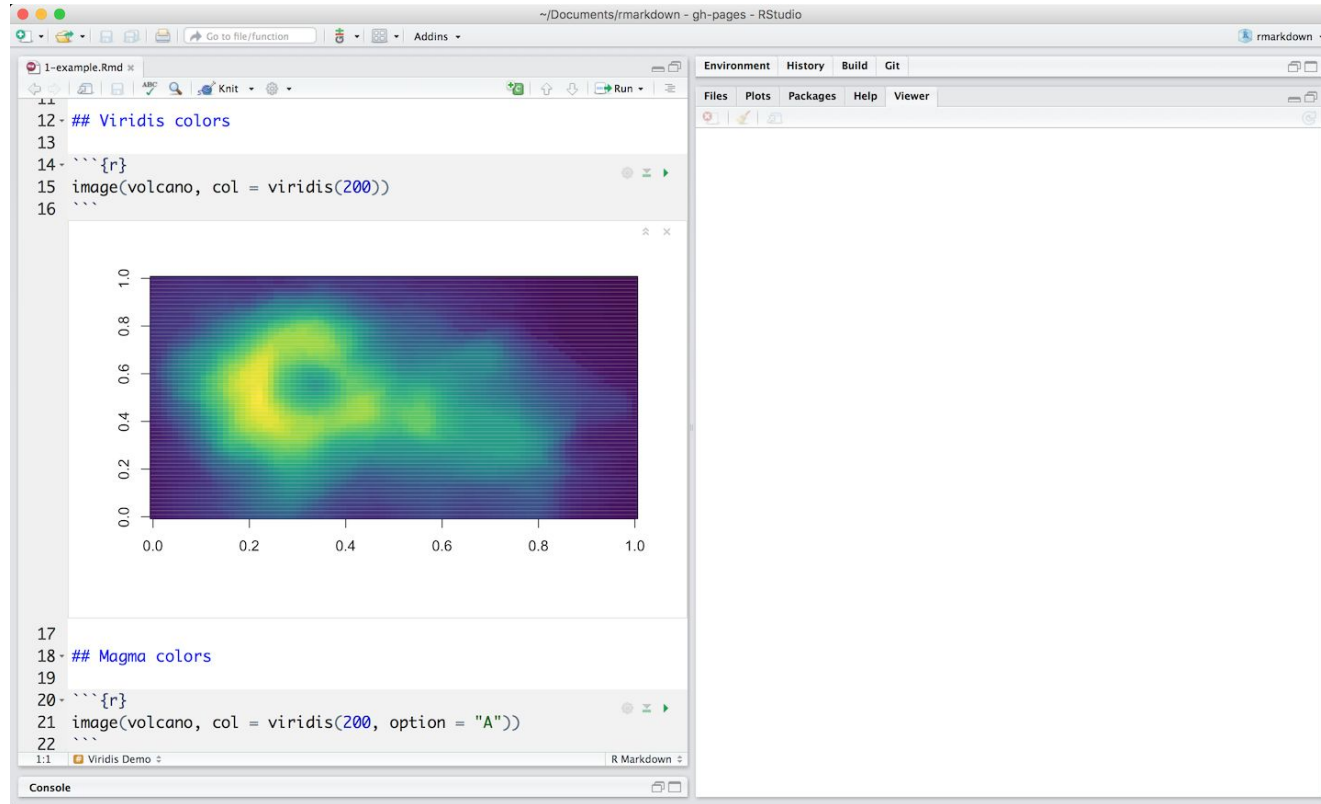


“When you run render, R Markdown feeds the .Rmd file to knitr, which executes all of the code chunks and creates a new markdown (.md) document which includes the code and its output.

The markdown file generated by knitr is then processed by pandoc which is responsible for creating the finished format.”

<https://rmarkdown.rstudio.com>







Markdown Basics

Output Formats

Notebooks

Slide Presentations

Dashboards

Websites

Interactive Documents

Cheatsheets

file below, which is available [here](#) on RStudio Cloud.

The screenshot displays the RStudio interface. On the left, the R Markdown source file '10-presentation.Rmd' is open, showing the following code:

```
1 ---
2 title: "Viridis Presentation"
3 output:
4   revealjs::revealjs_presentation:
5     theme: league
6 ---
7
8 ```{r include = FALSE}
9 knitr::opts_chunk$set(echo = FALSE)
10 library(viridis)
11 ```
12
13 The [viridis](https://github.com/sjmgarnier/viridis)
14 package contains four color palettes, revealed in the
15 plots that follow.
16
17 >- Viridis
18 >- Magma
19 >- Inferno
20 >- Plasma
21
22 Each plot displays a contour map of the Maunga Whau
23 volcano in Auckland, New Zealand.
24
25 ## Viridis colors
26
27 ```{r}
28 image(volcano, col = viridis(200))
29 ```
```

On the right, the rendered HTML output '10-presentation.html' is shown in a browser window. It features a dark background with the title 'PLASMA COLORS' in white. Below the title is a contour plot of the Maunga Whau volcano, colored using the 'plasma' color palette. The plot has axes ranging from 0.0 to 1.0.

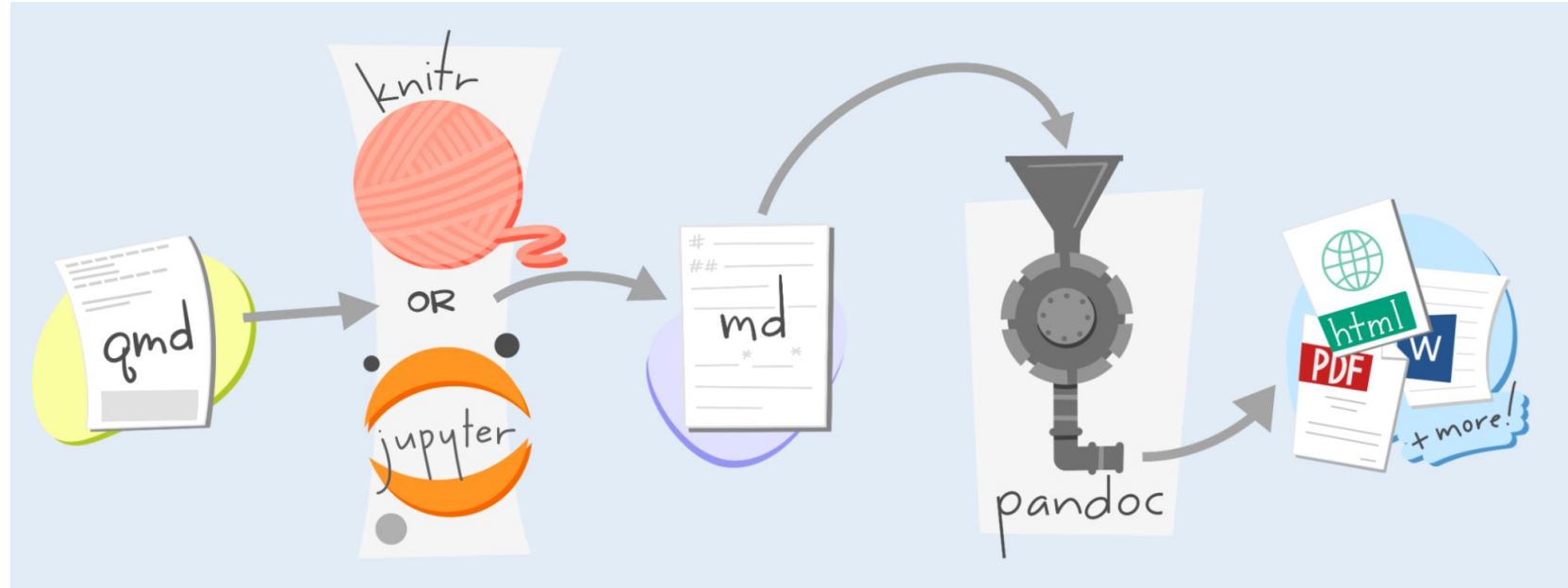




Quarto is an open-source scientific and technical publishing system where authors :

- Can use Jupyter notebooks or with plain text markdown in your favorite editor.
- Create dynamic content with Python, R, Julia, and Observable.
- Publish reproducible, production quality articles, presentations, websites, blogs, and books in HTML, PDF, MS Word, ePub, and more.
- Share results in a lot of publishing systems like GitHub.





With R

```

title: "ggplot2 demo"
author: "Norah Jones"
date: "5/22/2021"
format:
  html:
    fig-width: 8
    fig-height: 4
    code-fold: true

```

Air Quality

@fig-airquality further explores the impact of temperature on ozone level.

```

{r}
| label: fig-airquality
| fig-cap: "Temperature and ozone level."
| warning: false

library(ggplot2)

ggplot(airquality, aes(Temp, Ozone)) +
  geom_point() +
  geom_smooth(method = "loess")

```

ggplot2 demo

Norah Jones

May 22nd, 2021

Air Quality

[Figure 1](#) further explores the impact of temperature on ozone level.

► Code

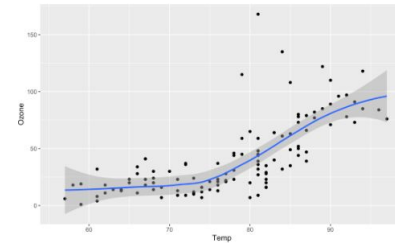


Figure 1: Temperature and ozone level.

With Jupyter

cell-options.ipynb Python 3 (ipykernel)

Palmer Penguins

author: Norah Jones
format: html
code-tools: true
code-fold: true

```

[3]: #| echo: false
      | import pandas as pd
      | df = pd.read_csv("palmer-penguins.csv")

```

Exploring the Data

See @fig-bill-sizes for an exploration of bill sizes by species.

```

[6]: #| label: fig-bill-sizes
      | #| fig-cap: Bill Sizes by Species
      | import matplotlib.pyplot as plt
      | import seaborn as sns
      | g = sns.FacetGrid(df, hue="species", height=3, aspect=3.5/1.5)
      | g.map(plt.scatter, "bill_length_mm", "bill_depth_mm").add_legend()

[6]: <seaborn.axisgrid.FacetGrid at 0x29467200>

```

A faceted scatter plot showing the relationship between bill length (bill_length_mm) on the x-axis and bill depth (bill_depth_mm) on the y-axis, faceted by species (Adelie, Gentoo, Chinstrap). The x-axis ranges from 35 to 60, and the y-axis ranges from 14 to 20. The legend indicates that blue dots represent Adelie, orange dots represent Gentoo, and green dots represent Chinstrap.

Palmer Penguins

AUTHOR
Norah JonesPUBLISHED
March 12, 2023

Code

Show All Code

Hide All Code

Exploring the Data

See [Figure 1](#) for an exploration of bill sizes by species.

▼ Code

```

import matplotlib.pyplot as plt
import seaborn as sns
g = sns.FacetGrid(df, hue="species", height=3, aspect=3.5/2)
g.map(plt.scatter, "bill_length_mm", "bill_depth_mm").add_legend()

```

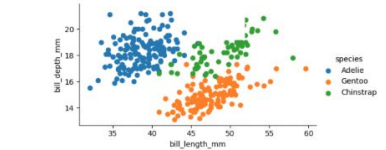


Figure 1: Bill Sizes by Species



Rmarkdown

```
quarto render code/supplementary_material.Rmd --to html  
quarto render code/supplementary_material.Rmd --to docx
```

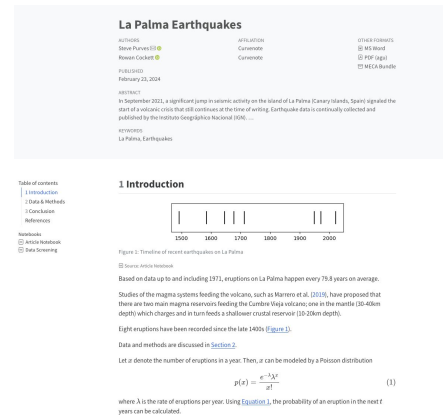
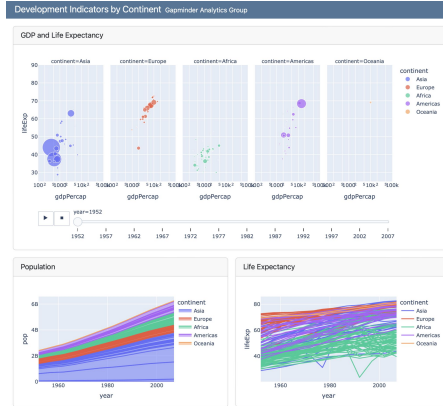


Jupyter

```
quarto render code/supplementary_material.ipynb --to html  
quarto render code/supplementary_material.ipynb --to docx
```



Documents	>
HTML	>
PDF	>
MS Word	>
Typst	>
Markdown	>
All Formats	>
Presentations	>
Dashboards	>
Websites	>
Books	>
Manuscripts	>
Interactivity	>
Publishing	>
Projects	>
Advanced	>



R for Data Science (2e)

- Whole game
- 1 Data visualization
 - 2 Workflow: basics
 - 3 Data transformation
 - 4 Workflow: code style
 - 5 Data tidying
 - 6 Workflow: scripts and projects
 - 7 Data import
 - 8 Workflow: getting help

- Visualize
- 9 Layers
 - 10 Exploratory data analysis
 - 11 Communication

- Transform
- 12 Logical vectors
 - 13 Numbers
 - 14 Strings
 - 15 Regular expressions
 - 16 Factors
 - 17 Dates and times
 - 18 Missing values
 - 19 Joins

- Import
- 20 Spreadsheets
 - 21 Databases
 - 22 Arrow
 - 23 Hierarchical data
 - 24 Web scraping

- Program
- 25 Functions
 - 26 Iteration
 - 27 A field guide to base R

- Communicate
- 28 Quarto
 - 29 Quarto formats

R for Data Science (2e)

Welcome

This is the website for the 2nd edition of “**R for Data Science**”. This book will teach you how to do data science with R: You’ll learn how to get your data into R, get it into the most useful structure, transform it and visualize.

In this book, you will find a practicum of skills for data science. Just as a chemist learns how to clean test tubes and stock a lab, you’ll learn how to clean data and draw plots—and many other things besides. These are the skills that allow data science to happen, and here you will find the best practices for doing each of these things with R. You’ll learn how to use the grammar of graphics, literate programming, and reproducible research to save time. You’ll also learn how to manage cognitive resources to facilitate discoveries when wrangling, visualizing, and exploring data.

This website is and will always be free, licensed under the [CC BY-NC-ND 3.0 License](#). If you’d like a physical copy of the book, you can order it on [Amazon](#). If you appreciate reading the book for free and would like to give back, please make a donation to [Kākāpō Recovery](#): the [kākāpō](#) (which appears on the cover of R4DS) is a critically endangered parrot native to New Zealand; there are only 248 left.

If you speak another language, you might be interested in the freely available translations of the 1st edition:

- [Spanish](#)
- [Italian](#)
- [Turkish](#)

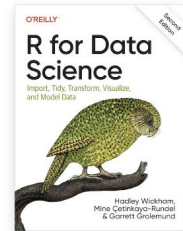
You can find suggested answers to exercises in the book at <https://mine-cetinkaya-rundel.github.io/r4ds-solutions>.

Please note that R4DS uses a [Contributor Code of Conduct](#). By contributing to this book, you agree to abide by its terms.

Acknowledgements

R4DS is hosted by <https://www.netlify.com> as part of their support of open source software and communities.

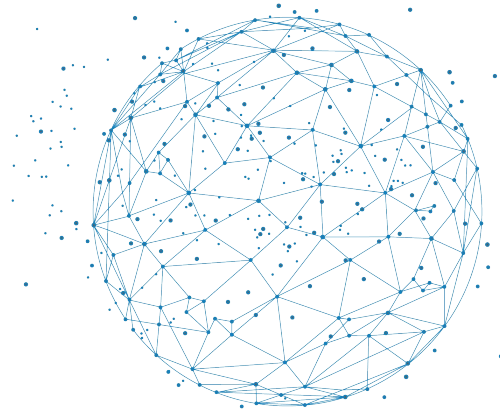
Table of contents
Welcome
Acknowledgements
Edit this page
Report an issue





Method	Jupyter	Rmarkdown	Quarto
IDE	JupyterHub, JupyterLab	R, Rstudio	VS Code, Jupyter, Rstudio, Neovim, TextEditor
Code mixing	Limited	Yes	Yes
Format	ipynb	Rmd	Qmd, Rmd, ipynb
Output	Asciidoc, HTML, LaTeX, MD, PDF, RST, Slide (Reveal)	HTML, PDF, Docx, ODT, RTF, MD, Slides (Powerpoint, Reveal,...), Dashboard, ...	HTML, PDF, Docx, ODT, Epub, RTF, MD, Slides (Powerpoint, Reveal,...), Wiki (MediaWiki, ...), Book, and more !
Reproducibility	Easy	Easy	Yes (if done from the start)

Observable



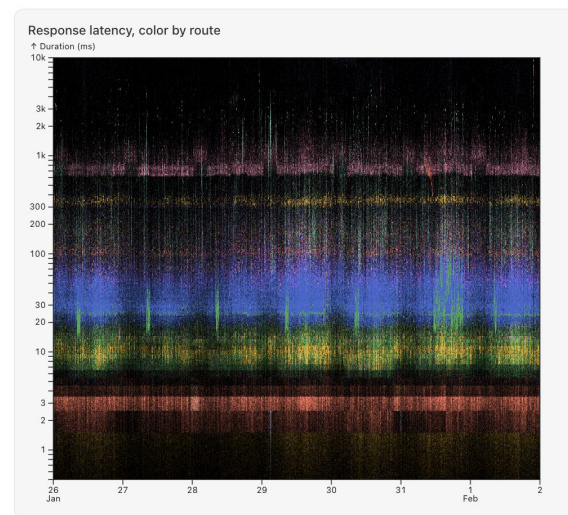
Observable Framework

Observable is an online platform that enables users to create, execute, and share interactive notebooks. These notebooks on Observable are based on JavaScript and allow users to combine code, visualizations, and text within a single interactive document.

Analyzing web logs

Web logs capture traffic metadata, such as the request time and route, how long the server took to respond, the response size, and so on. Analyzing web logs sheds light on both server performance and client behavior. Yet the common practice of summary statistics (e.g., 95th-percentile latency) often hides interesting patterns! This is because performance varies wildly based on the nature of the request, and unusual clients such as bots can easily hide in a sea of “natural” traffic.

What if — instead of summarizing — we plotted *every* request as a dot with time along $x \rightarrow$ and latency (on a log scale) along $y \uparrow$?



The plot above shows a sample of 7,633,176 requests to Observable servers over a 7-day period. Color encodes the associated route. Hover to see the route.

<https://observablehq.com/framework/examples/api/>

Official documentation

Observable Framework

Q Search

What is Framework?

Getting started

Project structure

Markdown

JavaScript

Reactivity

Imports

Data loaders

Files

SQL

Themes

Configuration

Deploying

Inputs

Libraries

Contributing

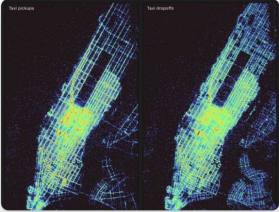
Observable Framework

1.5.1 GitHub 1.8k

What is Framework?

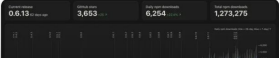
Observable Framework — or “Framework” for short — is an open-source static-site generator for data apps. By *data app* we mean an application that is primarily a display of data. Data apps help you derive insights (to understand) and evaluate potential decisions (to take action).

A data app might be a set of coordinated **interactive visualizations** for “self-service” analysis, perhaps to explore a computational model or to investigate activity;



Taxi rides in New York City →

... or it might be a **live dashboard** that places current events in the context of recent or historical trends;



Via quarto

quarto

Overview Get Started Guide Extensions Reference Gallery Blog Help

Guide

Authoring

Computations

Tools

Documents

Presentations

Dashboards

Websites

Books

Manuscripts

Interactivity

Overview

Observable JS

Introduction

Libraries

Data Sources

QJS Cells

Shiny Reactives

Code Reuse

Examples

Shiny

Widgets

Component Layout

Publishing

Projects

Advanced

Guide > Interactivity > Observable JS

Observable JS

Overview

Quarto includes native support for [Observable JS](#), a set of enhancements to vanilla JavaScript created by [Mike Bostock](#) (also the author of [D3](#)). Observable JS is distinguished by its [reactive runtime](#), which is especially well suited for interactive data exploration and analysis.

The creators of Observable JS (Observable, Inc.) run a hosted service at <https://observablehq.com/> where you can create and publish notebooks. Additionally, you can use Observable JS (“OJS”) in standalone documents and websites via its [core libraries](#). Quarto uses these libraries along with a [compiler](#) that is run at render time to enable the use of OJS within Quarto documents.

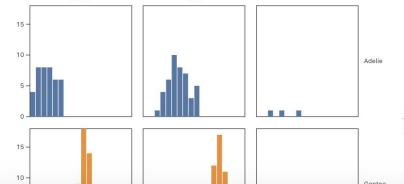
OJS works in any Quarto document (plain markdown as well as Jupyter and Knitr documents). Just include your code in an `{ojs}` executable code block. The rest of this article explains the basics of using OJS with Quarto.

Example

We'll start with a simple example based on Allison Horst's [Palmer Penguins](#) dataset. Here we look at how penguin body mass varies across both sex and species (use the provided inputs to filter the dataset by bill length and island):

Bill length (min):

Islands: ☒ Torgersen ☒ Biscoe ☐ Dream





Thank you for your
attention !



INSTITUT FRANÇAIS DE BIOINFORMATIQUE

