Entering Behind the Scenes: Computational Documents

Arnaud Legrand

LIG, Univ. Grenoble Alpes, CNRS, Inria

arnaud.legrand@imag.fr

September 3, 2018

Outline

Where are we?

M2-S0: Computational Documents

M2-S1: A few Recent Controversial Studies

M2-S2: Why is This so Difficult?

M2-S3: Computational Documents: Principles

M2-S4A: Hands On (Jupyter)

M2-S4B: Hands On (Rstudio)

M2-S4C: Hands on (Org-Mode)

M2-S5: Collaborating

M2-S6: Comparative Study

Entering Behind the Scenes: Computational Documents

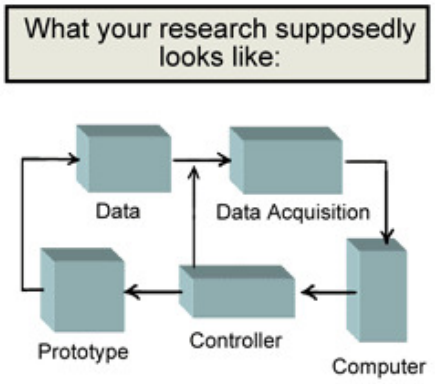


Figure 1. Experimental Diagram

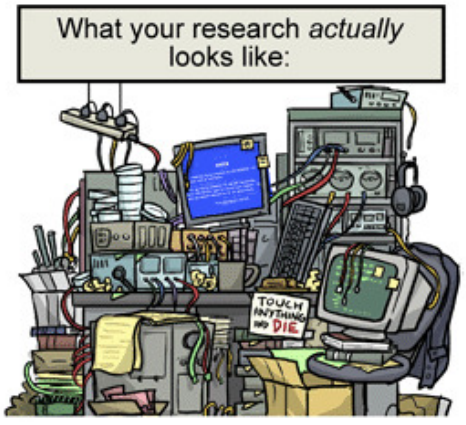


Figure 2. Experimental Mess

1. A few Recent Controversial Studies
2. Why is This so Difficult?
3. Computational Document: Principles
4. Hands on.
   - Jupyter
   - Rstudio
   - Org-Mode
5. Collaborating
6. Comparative study

Where are we?

M2-S0: Computational Documents

M2-S1: A few Recent Controversial Studies

M2-S2: Why is This so Difficult?

M2-S3: Computational Documents: Principles

M2-S4A: Hands On (Jupyter)

M2-S4B: Hands On (Rstudio)

M2-S4C: Hands on (Org-Mode)

M2-S5: Collaborating

M2-S6: Comparative Study

Entering Behind the Scenes: Computational Documents

1. A few Recent Controversial Studies
2. Why is This so Difficult?
3. Computational Document: Principles
4. Hands on.
   - Jupyter
   - Rstudio
   - Org-Mode
5. Collaborating
6. Comparative study

A Few Recent Controversial Studies

Economy: Austerity in Fiscal Policy (1/2)

2010

gross external debt reaches 60 percent of GDP, a country's annual growth declined by two percent
[..]for levels of external debt in excess of 90 percent, GDP growth was roughly cut in half

– *Reinhart et Rogoff: Growth in a Time of Debt*

Economy: Austerity in Fiscal Policy (2/2)

2013

While using RR's working spreadsheet, we identified coding errors, selective exclusion of available data, and unconventional weighting of summary statistics.

– Herndon, Ash et Pollin

combining data across centuries, exchange rate regimes, public and private debt, and debt denominated in foreign currency as well as domestic currency.

– Wray

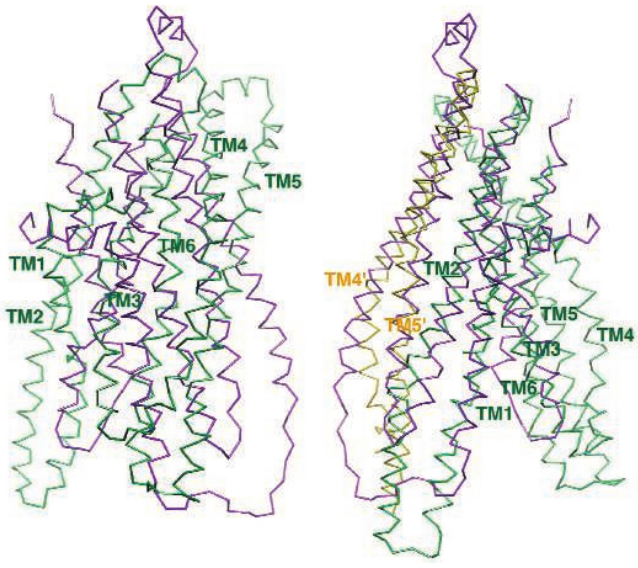
Functional MRI

- 2010: Bennett et al. and the dead salmon ☺
- 2016: Eklund, Nichols, and Knutsson. A bug in fmri software could invalidate 15 years of brain research (*40 000 articles*)
- 2016: But it's more subtle than it looks like. Nichols. $\approx$ *3 600 impacted studies*

Statistical methods and methodology should be improved but no fundamental invalidation

Incorrect Protein Structures

Geoffrey Chang: study the tertiary structures of membrane proteins of multidrug resistant bacteria
MsbA de Escherichia Coli (Science, 2001), Vibrio cholera (Mol. Biology, 2003), Salmonella typhimurium (Science, 2005)



2006: Inconsistencies, alerts, then 5 retractions

> *a homemade data-analysis program had flipped two columns of data, inverting the electron-density map from which his team had derived the protein structure.*
>
> – *a "buggy software"*

Loosing Faith?

- Oncology: "*half of published studies, even in prestigious journals, can't be reproduced in industrial labs*"
- Psychology: "*attempting to reproduce 100 previously published findings, only one-third of published psychology research was found to be reliable*"

Whistle blowers or dysfunctional institutions?

Questioning is part of the scientific processus

Loosing Faith?

- Oncology: "*half of published studies, even in prestigious journals, can't be reproduced in industrial labs*"
- Psychology: "*attempting to reproduce 100 previously published findings, only one-third of published psychology research was found to be reliable*"

Whistle blowers or dysfunctional institutions?

Questioning is part of the scientific processus

Just like rigor and transparency. . .

Where are we?

Entering Behind the Scenes: Computational Documents

1. A few Recent Controversial Studies
2. Why is This so Difficult?
3. Computational Document: Principles
4. Hands on.
   - Jupyter
   - Rstudio
   - Org-Mode
5. Collaborating
6. Comparative study

Why is This so Difficult?

1) Information Scarcity

Clearly indicate:

- Provenance and data

  Unavailable data = hardly verifiable results

- Decisions

  Unexplained Decision = Suspicious Choice

Laboratory Notebooks may help

2) Computers broke science

- Point and click:
- Spreadsheets: programming and data manipulation mistakes
  - `Membrane-Associated Ring Finger (C3HC4) 1, E3 Ubiquitin Protein Ligase` → `MARCH1` → 2016-03-01 → 1456786800
  - `2310009E13` → 2.31E+19
- Complex software stack
- Bug: *Coding is a difficult task!*

Are Computers the Only Ones to Blame?

Lack of rigor and organization

- No backup
- No history
- No quality control

Social and Cultural Causes

Article = simplified version of the procedure

Tracing all these information and making them available = substantial investment

If no one requires/inspect such information, why should I worry?

Going Public?

- Weaknesses would become obvious
- Someone may find a flaw
- Someone may benefit from my hard work
- *Data may be sensitive*

Let us give ourselves the means to have everything inspectable on demand.

Tools to Avoid and Possible Alternatives

- Proprietary tools, formats and services
  1. ~~Excel, Word, Evernote~~
     - Markdown, Org-mode, CSV, HDF5, ...
  2. ~~SAS, Minitab, matlab, mathematica, ...~~
     - Scilab, R, Python, ...
  3. ~~Dropbox, online proprietary lab. notebooks , ...~~
     - Framadrop, GitLab/GitHub, ...
- "Intuitive" Tools
  - ~~spreadsheet, graphical interfaces, interactive exploration~~
    - learn self control and slow down... 
    - R, Python, ...

Paradigm Shift

1. Information scarcity, difficulties in accessing data
2. Computation mistakes
3. Lack of scientific and technical rigor

Making everything explicit increases the chances of finding and getting rid of mistakes

Where are we?

M2-S0: Computational Documents

M2-S1: A few Recent Controversial Studies

M2-S2: Why is This so Difficult?

M2-S3: Computational Documents: Principles

M2-S4A: Hands On (Jupyter)

M2-S4B: Hands On (Rstudio)

M2-S4C: Hands on (Org-Mode)

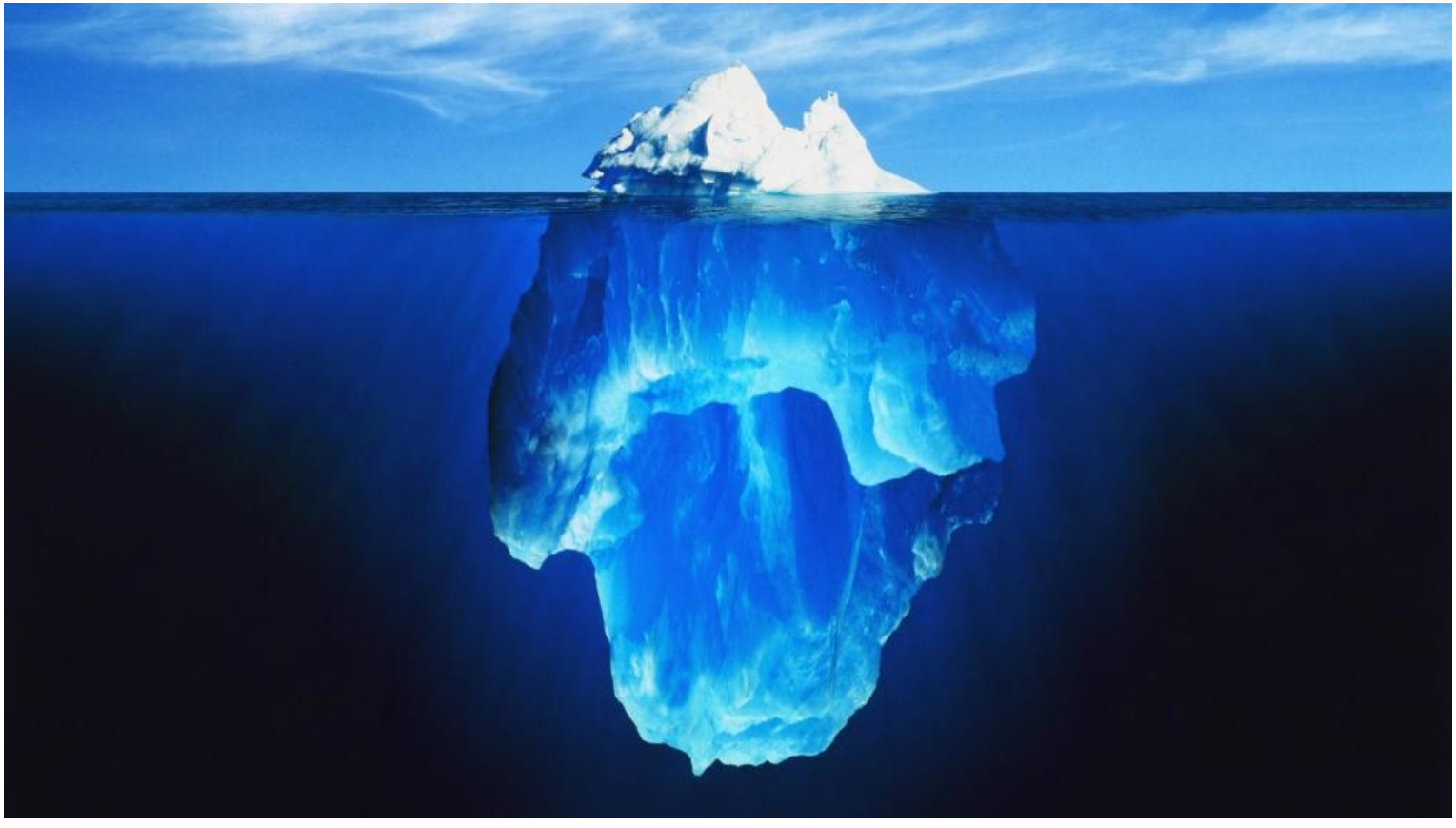
M2-S5: Collaborating

M2-S6: Comparative Study

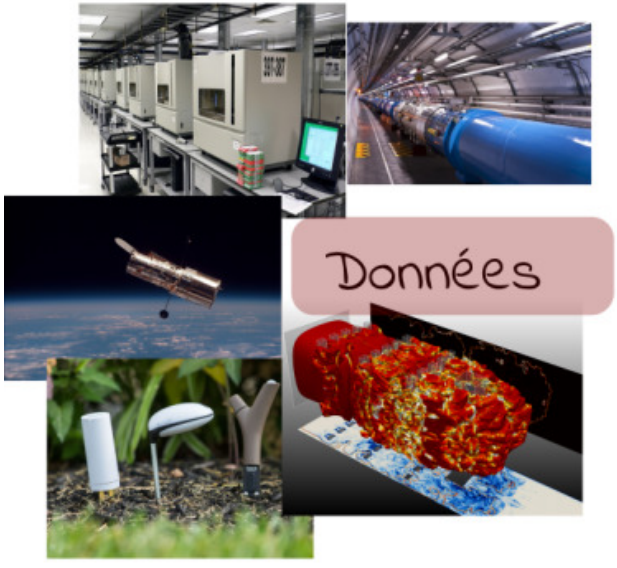
Entering Behind the Scenes: Computational Documents

1. A few Recent Controversial Studies
2. Why is This so Difficult?
3. Computational Document: Principles
4. Hands on.
   - Jupyter
   - Rstudio
   - Org-Mode
5. Collaborating
6. Comparative study

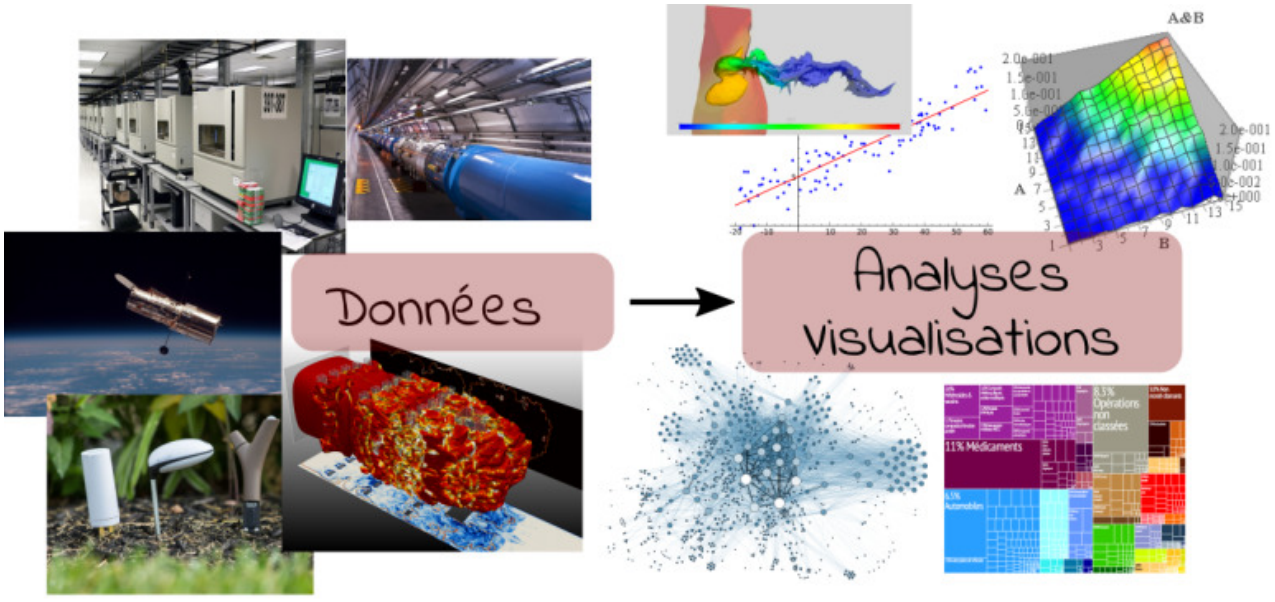
Computational Documents: Principles



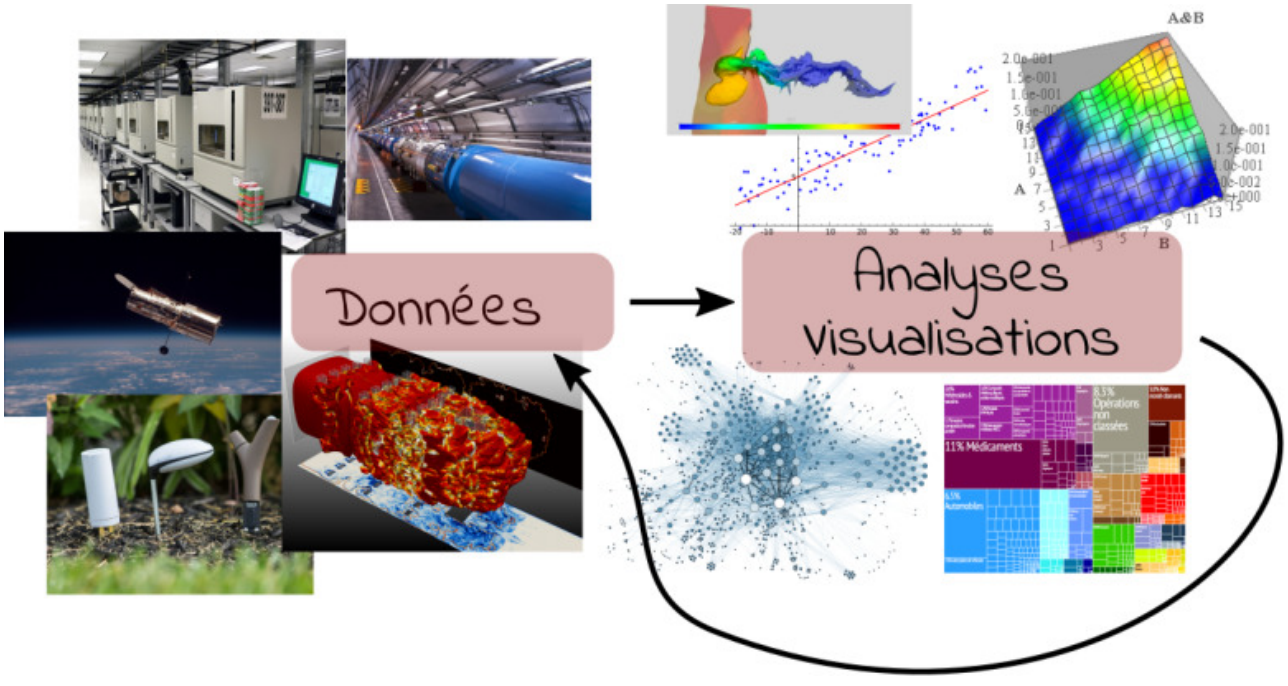
Modern Science



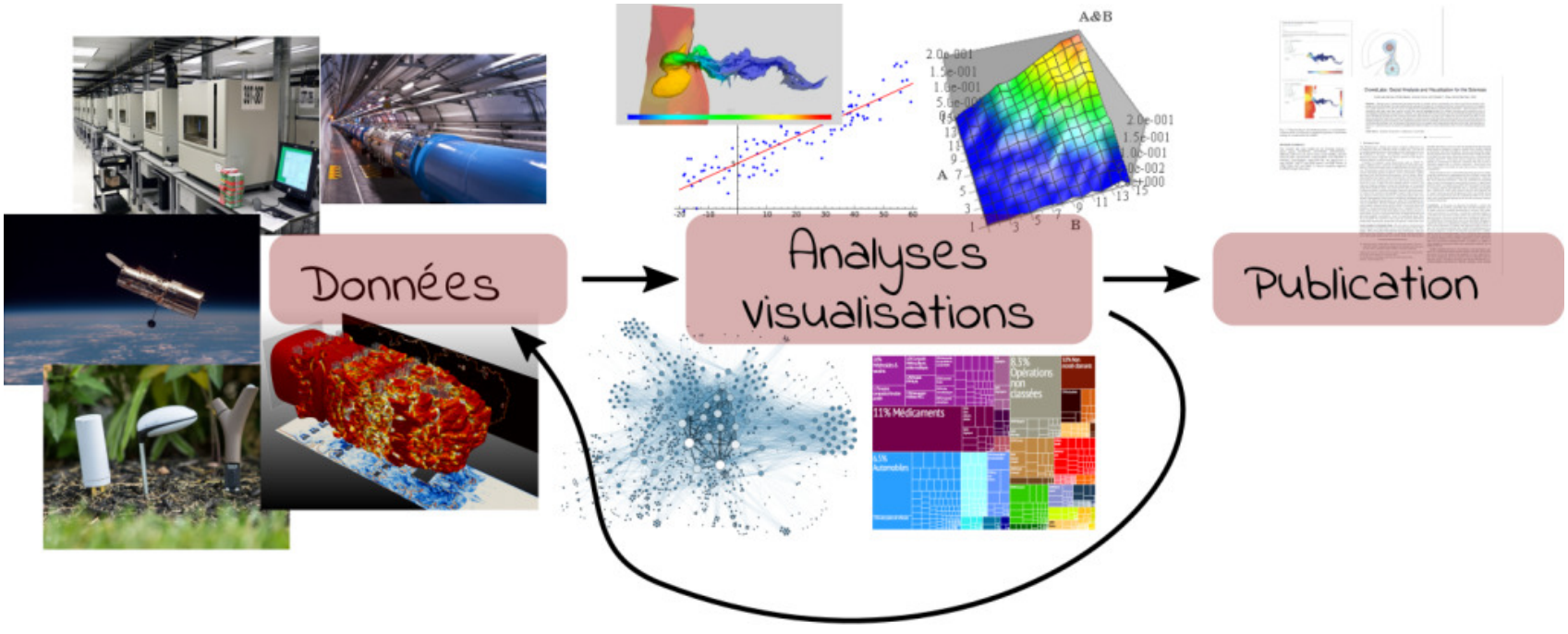
Modern Science



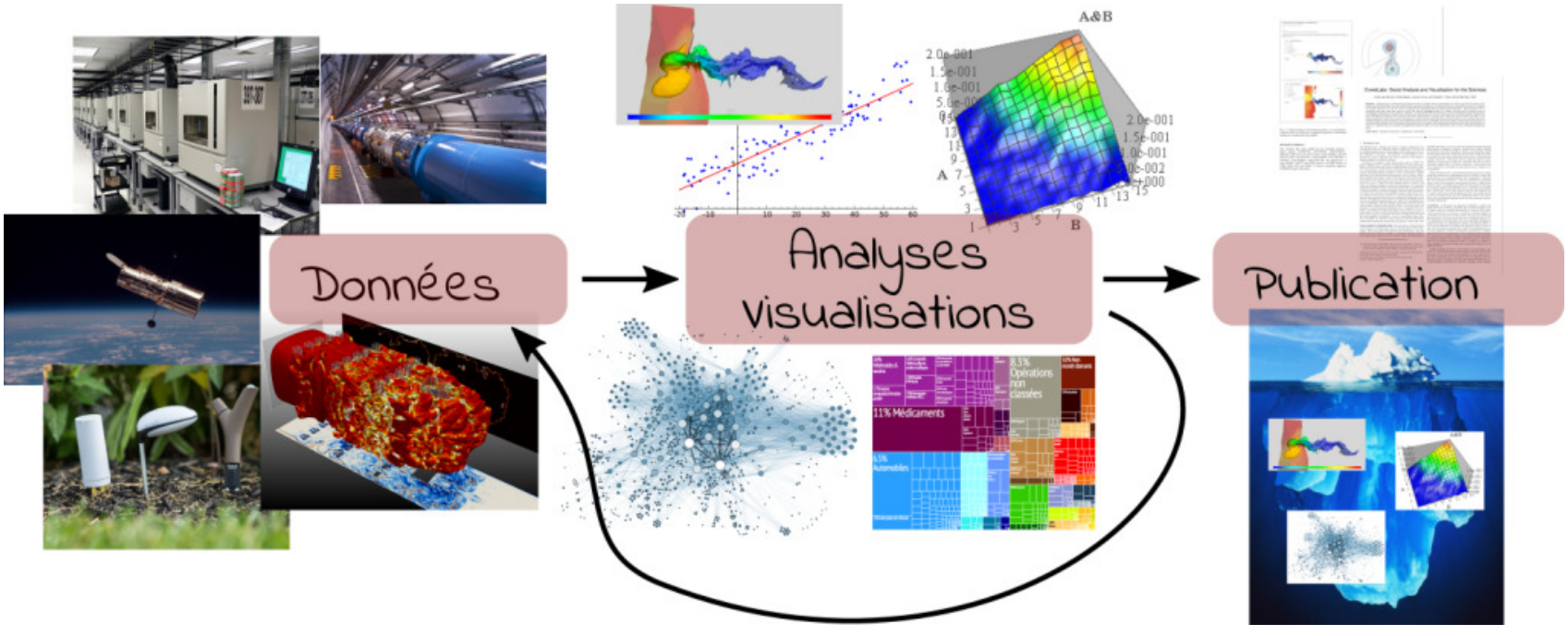
Modern Science



Modern Science



Modern Science



Methodological Goals

Keep track to allow:

- Inspection: justify/understand
- Re-execution: check/fix/improve/reuse

Behind the Scenes

Un document computationnel

Mon ordinateur m'indique que π vaut *approximativement*

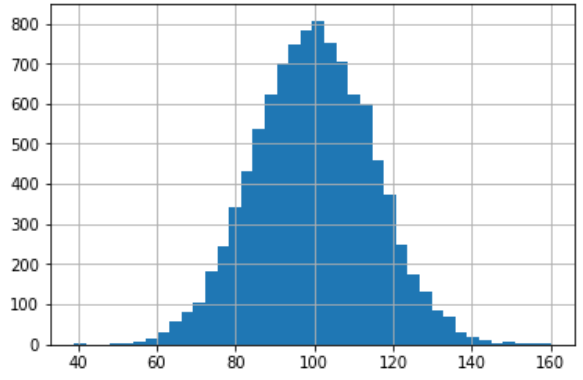
```
3.141592653589793
```

Mais calculé avec la **méthode** des aiguilles de Buffon, on obtiendrait comme **approximation** :

```
import numpy as np
N = 1000000
x = np.random.uniform(size=N, low=0, high=1)
theta = np.random.uniform(size=N, low=0, high=pi/2)
2/(sum((x+np.sin(theta))>1)/N)
```

```
3.1437198694098765
```

On peut inclure des formules mathématiques comme $\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ et des *dessins qui n'ont rien à voir* avec π (si ce n'est une constante de normalisation... ☺).



Behind the Scenes

Document initial dans son environnement

jupyter example_pi (autosaved)

File Edit View Insert Cell Kernel Widgets Help Hide Code Python 3

Code CellToolbar

Hide Prompts Hide Code Hide Outputs

```
# Un document computationnel
```

Hide Prompts Hide Code Hide Outputs

```
Mon ordinateur m'indique que $\pi$ vaut *approximativement*
```

In [1]:

Hide Prompts Hide Code Hide Outputs

```
from math import *
print(pi)
```

```
3.141592653589793
```

Hide Prompts Hide Code Hide Outputs

```
Mais calculé avec la __méthode__ des [aiguilles de Buffon]
(https://fr.wikipedia.org/wiki/Aiguille_de_Buffon), on obtiendrait comme
__approximation__ :
```

In [2]:

Hide Prompts Hide Code Hide Outputs

```
import numpy as np
N = 1000000
x = np.random.uniform(size=N, low=0, high=1)
theta = np.random.uniform(size=N, low=0, high=pi/2)
2/(sum((x+np.sin(theta))>1)/N)
```

Out[2]: 3.1437198694098765

Hide Prompts Hide Code Hide Outputs

```
On peut inclure des formules mathématiques comme $\frac{1}{\sigma
\sqrt{2\pi}} \exp \left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ et des
_dessins qui n'ont rien à voir_ avec $\pi$ (si ce n'est une constante de
normalisation... ☺).
```

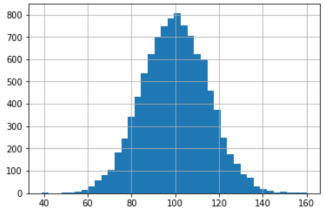
In [3]:

Hide Prompts Hide Code Hide Outputs

```
%matplotlib inline
import matplotlib.pyplot as plt

mu, sigma = 100, 15
x = mu + sigma*np.random.randn(10000)

plt.hist(x,40)
plt.grid(True)
plt.show()
```



Document final

Un document computationnel

Mon ordinateur m'indique que π vaut *approximativement*

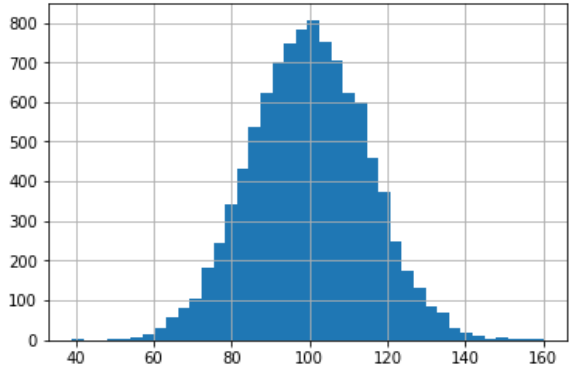
```
3.141592653589793
```

Mais calculé avec la **méthode** des aiguilles de Buffon, on obtiendrait comme **approximation** :

```
import numpy as np
N = 1000000
x = np.random.uniform(size=N, low=0, high=1)
theta = np.random.uniform(size=N, low=0, high=pi/2)
2/(sum((x+np.sin(theta))>1)/N)
```

```
3.1437198694098765
```

On peut inclure des formules mathématiques comme $\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ et des *dessins qui n'ont rien à voir* avec π (si ce n'est une constante de normalisation... ☺).



Behind the Scenes

Document initial dans son environnement

jupyter example_pi (autosaved)

File Edit View Insert Cell Kernel Widgets Help Hide Code Python 3

Code CellToolbar

Hide Prompts Hide Code Hide Outputs

```
# Un document computationnel
```

Hide Prompts Hide Code Hide Outputs

```
Mon ordinateur m'indique que $\pi$ vaut *approximativement*
```

In [1]:

Hide Prompts Hide Code Hide Outputs

```
from math import *
print(pi)
```

```
3.141592653589793
```

Hide Prompts Hide Code Hide Outputs

```
Mais calculé avec la __méthode__ des [aiguilles de Buffon]
(https://fr.wikipedia.org/wiki/Aiguille_de_Buffon), on obtiendrait comme
__approximation__ :
```

In [2]:

Hide Prompts Hide Code Hide Outputs

```
import numpy as np
N = 1000000
x = np.random.uniform(size=N, low=0, high=1)
theta = np.random.uniform(size=N, low=0, high=pi/2)
2/(sum((x+np.sin(theta))>1)/N)
```

Out[2]: 3.1437198694098765

Hide Prompts Hide Code Hide Outputs

```
On peut inclure des formules mathématiques comme $\frac{1}{\sigma
\sqrt{2\pi}} \exp \left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ et des
_dessins qui n'ont rien à voir_ avec $\pi$ (si ce n'est une constante de
normalisation... ☺).
```

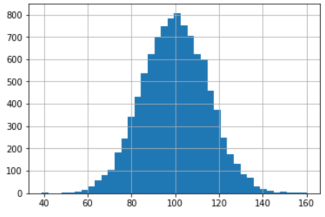
In [3]:

Hide Prompts Hide Code Hide Outputs

```
%matplotlib inline
import matplotlib.pyplot as plt

mu, sigma = 100, 15
x = mu + sigma*np.random.randn(10000)

plt.hist(x,40)
plt.grid(True)
plt.show()
```



MarkDown

Document final

Un document computationnel

Mon ordinateur m'indique que π vaut *approximativement*

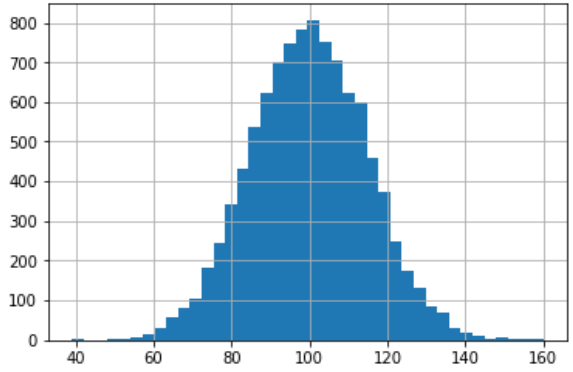
```
3.141592653589793
```

Mais calculé avec la **méthode** des aiguilles de Buffon, on obtiendrait comme **approximation** :

```
import numpy as np
N = 1000000
x = np.random.uniform(size=N, low=0, high=1)
theta = np.random.uniform(size=N, low=0, high=pi/2)
2/(sum((x+np.sin(theta))>1)/N)
```

```
3.1437198694098765
```

On peut inclure des formules mathématiques comme $\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ et des *dessins qui n'ont rien à voir* avec π (si ce n'est une constante de normalisation... ☺).



Behind the Scenes

Document initial dans son environnement

jupyter example_pi (autosaved)

File Edit View Insert Cell Kernel Widgets Help Hide Code Python 3

Code CellToolbar

Hide Prompts Hide Code Hide Outputs

```
# Un document computationnel
```

Hide Prompts Hide Code Hide Outputs

```
Mon ordinateur m'indique que $\pi$ vaut *approximativement*
```

In [1]:

Hide Prompts Hide Code Hide Outputs

```
from math import *
print(pi)
```

```
3.141592653589793
```

Hide Prompts Hide Code Hide Outputs

```
Mais calculé avec la __méthode__ des [aiguilles de Buffon]
(https://fr.wikipedia.org/wiki/Aiguille_de_Buffon), on obtiendrait comme
__approximation__ :
```

In [2]:

Hide Prompts Hide Code Hide Outputs

```
import numpy as np
N = 1000000
x = np.random.uniform(size=N, low=0, high=1)
theta = np.random.uniform(size=N, low=0, high=pi/2)
2/(sum((x+np.sin(theta))>1)/N)
```

Out[2]: 3.1437198694098765

Hide Prompts Hide Code Hide Outputs

```
On peut inclure des formules mathématiques comme $\frac{1}{\sigma
\sqrt{2\pi}} \exp \left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ et des
_dessins qui n'ont rien à voir_ avec $\pi$ (si ce n'est une constante de
normalisation... ☺).
```

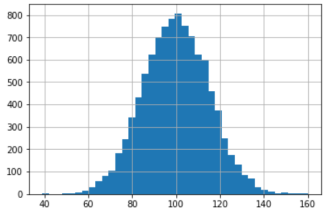
In [3]:

Hide Prompts Hide Code Hide Outputs

```
%matplotlib inline
import matplotlib.pyplot as plt

mu, sigma = 100, 15
x = mu + sigma*np.random.randn(10000)

plt.hist(x,40)
plt.grid(True)
plt.show()
```



Code

Document final

Un document computationnel

Mon ordinateur m'indique que π vaut *approximativement*

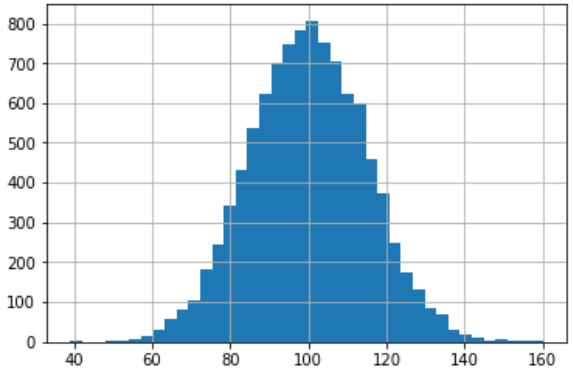
```
3.141592653589793
```

Mais calculé avec la **méthode** des aiguilles de Buffon, on obtiendrait comme **approximation** :

```
import numpy as np
N = 1000000
x = np.random.uniform(size=N, low=0, high=1)
theta = np.random.uniform(size=N, low=0, high=pi/2)
2/(sum((x+np.sin(theta))>1)/N)
```

```
3.1437198694098765
```

On peut inclure des formules mathématiques comme $\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ et des *dessins qui n'ont rien à voir* avec π (si ce n'est une constante de normalisation... ☺).



Behind the Scenes

Document initial dans son environnement

jupyter example_pi (autosaved)

File Edit View Insert Cell Kernel Widgets Help Hide Code Python 3

Code CellToolbar

Hide Prompts Hide Code Hide Outputs

```
# Un document computationnel
```

Hide Prompts Hide Code Hide Outputs

```
Mon ordinateur m'indique que $\pi$ vaut *approximativement*
```

In [1]:

Hide Prompts Hide Code Hide Outputs

```
from math import *
print(pi)
```

```
3.141592653589793
```

Hide Prompts Hide Code Hide Outputs

```
Mais calculé avec la __méthode__ des [aiguilles de Buffon]
(https://fr.wikipedia.org/wiki/Aiguille_de_Buffon), on obtiendrait comme
__approximation__ :
```

In [2]:

Hide Prompts Hide Code Hide Outputs

```
import numpy as np
N = 1000000
x = np.random.uniform(size=N, low=0, high=1)
theta = np.random.uniform(size=N, low=0, high=pi/2)
2/(sum((x+np.sin(theta))>1)/N)
```

Out[2]: 3.1437198694098765

Hide Prompts Hide Code Hide Outputs

```
On peut inclure des formules mathématiques comme $\frac{1}{\sigma
\sqrt{2\pi}} \exp \left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ et des
_dessins qui n'ont rien à voir_ avec $\pi$ (si ce n'est une constante de
normalisation... ☺).
```

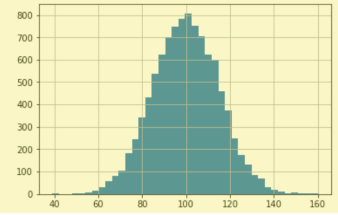
In [3]:

Hide Prompts Hide Code Hide Outputs

```
%matplotlib inline
import matplotlib.pyplot as plt

mu, sigma = 100, 15
x = mu + sigma*np.random.randn(10000)

plt.hist(x,40)
plt.grid(True)
plt.show()
```



Résultats

Document final

Un document computationnel

Mon ordinateur m'indique que π vaut *approximativement*

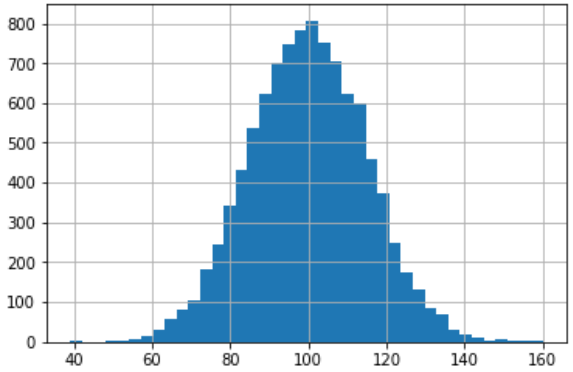
```
3.141592653589793
```

Mais calculé avec la **méthode** des aiguilles de Buffon, on obtiendrait comme **approximation** :

```
import numpy as np
N = 1000000
x = np.random.uniform(size=N, low=0, high=1)
theta = np.random.uniform(size=N, low=0, high=pi/2)
2/(sum((x+np.sin(theta))>1)/N)
```

```
3.1437198694098765
```

On peut inclure des formules mathématiques comme $\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ et des *dessins qui n'ont rien à voir* avec π (si ce n'est une constante de normalisation... ☺).



Behind the Scenes

Document initial dans son environnement

jupyter example_pi (autosaved)

File Edit View Insert Cell Kernel Widgets Help Hide Code Python 3

Code CellToolbar

Hide Prompts Hide Code Hide Outputs

```
# Un document computationnel
```

Hide Prompts Hide Code Hide Outputs

```
Mon ordinateur m'indique que $\pi$ vaut *approximativement*
```

In [1]:

Hide Prompts Hide Code Hide Outputs

```
from math import *
print(pi)
```

```
3.141592653589793
```

Hide Prompts Hide Code Hide Outputs

```
Mais calculé avec la __méthode__ des [aiguilles de Buffon]
(https://fr.wikipedia.org/wiki/Aiguille_de_Buffon), on obtiendrait comme
__approximation__ :
```

In [2]:

Hide Prompts Hide Code Hide Outputs

```
import numpy as np
N = 1000000
x = np.random.uniform(size=N, low=0, high=1)
theta = np.random.uniform(size=N, low=0, high=pi/2)
2/(sum((x+np.sin(theta))>1)/N)
```

Out[2]: 3.1437198694098765

Hide Prompts Hide Code Hide Outputs

```
On peut inclure des formules mathématiques comme $\frac{1}{\sigma
\sqrt{2\pi}} \exp \left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ et des
_dessins qui n'ont rien à voir_ avec $\pi$ (si ce n'est une constante de
normalisation... ☺).
```

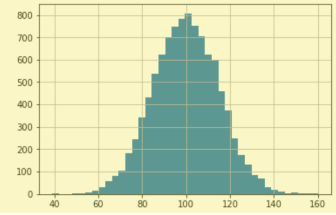
In [3]:

Hide Prompts Hide Code Hide Outputs

```
%matplotlib inline
import matplotlib.pyplot as plt

mu, sigma = 100, 15
x = mu + sigma*np.random.randn(10000)

plt.hist(x,40)
plt.grid(True)
plt.show()
```



Export

Document final

Un document computationnel

Mon ordinateur m'indique que π vaut *approximativement*

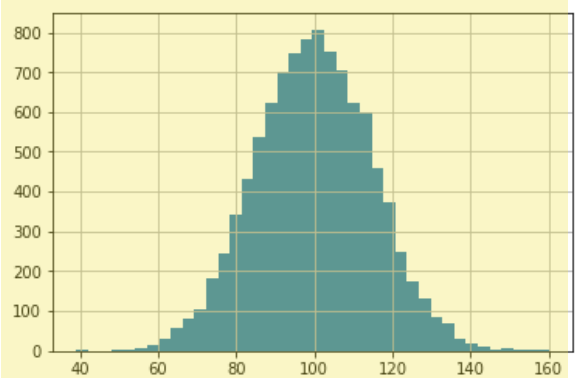
```
3.141592653589793
```

Mais calculé avec la **méthode** des aiguilles de Buffon, on obtiendrait comme **approximation** :

```
import numpy as np
N = 1000000
x = np.random.uniform(size=N, low=0, high=1)
theta = np.random.uniform(size=N, low=0, high=pi/2)
2/(sum((x+np.sin(theta))>1)/N)
```

```
3.1437198694098765
```

On peut inclure des formules mathématiques comme $\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ et des *dessins qui n'ont rien à voir* avec π (si ce n'est une constante de normalisation... ☺).



Existing Tools

1. Jupyter
2. Rstudio/knitR
3. Org mode

Same Principles	Differences
• A single document (explanations, code, results)	• Syntax
• Session	• Interoperability
• Export	• Controling export

Where are we?

Entering Behind the Scenes: Computational Documents

1. A few Recent Controversial Studies
2. Why is This so Difficult?
3. Computational Document: Principles
4. Hands on.
   - Jupyter
   - Rstudio
   - Org-Mode
5. Collaborating
6. Comparative study

Hands On: Jupyter

Startup

- Opening a document
- Quick Tour
- Saving
- Getting help

Running Cells

- Running and getting results
- Adding a cell
- Beware of the order
  - Session notion
  - Possible inconsistencies
  - Restart and run from the beginning

Keyboard Shortcuts, Completion, and Ipython Magic

- Keyboard shortcuts `<h>`
- Python completion (numpy example)
- `%matplotlib`, `%lsmagic`

Using Other Languages

- Example for R:
  - `%load_ext rpy2.ipython`
  - `%%R %%sh %%perl`
- Interactions between `R` and `Python` are possible

Producing and Sharing the Document

- Results are stored in the document
  - $\rightarrow$ pretty-printed in gitlab
  - `git pull/push`
- Export to HTML/PDF

Preparing an Article

- Hide-code plugin
- `%%latex` and `%%html`
- Customize *exporters*:

```
jupyter nbconvert --to mypackage.MyExporter notebook.ipynb
```

Recap

- A lot of information in short time period
- Now it's your turn!

Where are we?

Entering Behind the Scenes: Computational Documents

1. A few Recent Controversial Studies
2. Why is This so Difficult?
3. Computational Document: Principles
4. Hands on.
   - Jupyter
   - Rstudio
   - Org-Mode
5. Collaborating
6. Comparative study

Hands On: Rstudio

Startup

- Opening a document
- Quick Tour
- Saving
- Getting help

Running Chunks

- Running and getting results
- Adding a chunk
- Beware of the order
  - Session notion
  - Possible inconsistencies
  - Restart and run from the beginning

Keyboard Shortcuts, Completion

- Keyboard shortcuts
- R completion
- Folding

Producing and Sharing the Document

- Knit
- Easy sharing via rpubs

Controlling Code and Results Visibility

- Completion (chunk parameters)

Using a Specific Style

- pdf, LaTeX
- html
- word/office

Writing raw LaTeX (R Sweave: `Rnw`) or HTML (R HTML: `Rhtml`) to have full control is possible.

Using Other Langages

- Inserting and running a Python chunk
- Warning: no session !
  - Interaction between R and Python is done solely through files, which encourages to write long chunks 

Recap

- A lot of information in short time period
- Now it's your turn!

Where are we?

Entering Behind the Scenes: Computational Documents

1. A few Recent Controversial Studies
2. Why is This so Difficult?
3. Computational Document: Principles
4. Hands on.
   - Jupyter
   - Rstudio
   - Org-Mode
5. Collaborating
6. Comparative study

Hands On: Org Mode

Startup

- Opening a document
- Quick Tour
  - Folding / Browsing
  - Restructuring
- Saving
- Getting help

Running Code Blocks

- Inserting an R block
- Running and getting résultats
- Beware of the order
  - Session notion
  - Possible inconsistencies
  - Restart and run from the beginning

Keyboard Shortcuts, Completion

- Block expansion
  - R graphics
  - Python, Perl, ...
  - Shell session
- Several sessions, several languages !
- Language interactions

Browsing

- Folding
- Restructuration

Producing and Sharing the Document

- Git Commit
  - Beware of produced files
- Export
- Controling visibility of code and results
  - Hidden sections

Using a Specific Style

- pdf, LaTeX
- HTML
- Writing raw LaTeX or raw HTML in the middle of the org document is easy

Recap

- A lot of information in short time period
- Learn the shortcuts one after the other. The main ones are in the first entry of the journal
- Now it's your turn!

Where are we?

Entering Behind the Scenes: Computational Documents

1. A few Recent Controversial Studies
2. Why is This so Difficult?
3. Computational Document: Principles
4. Hands on.
   - Jupyter
   - Rstudio
   - Org-Mode
5. Collaborating
6. Comparative study

Collaborating

Preparing a Document for a Journal or a Conference

Requirements for producing a pdf:

- Internally, *pandoc*, *knitr* or *emacs/org-mode*
- LaTeX should be installed

Exporting as office/word documents is possible (requires a specific configuration in Jupyter). Otherwise export html...

In any case:

- Need to hide some cells
- Use the right style

Producing such kind of documents requires a perfectly configured environment

Convincing Your Co-authors

When confronted to this complexity, there are several possible attitudes:

1. Nevermind, it's awesome! Let's do this!
2. Err... it looks cool but I really don't have the time to learn this now...
3. Yet another new tool? Forget it!

$\rightsquigarrow$ several possible collaboration modes

Option 1: Enthusiastic Co-authors

You'll have to provide technical support:

- Compatibility issues between the different environments
- Manage this complexity (Jupyter/Rstudio/Emacs, Git, ...)

It is the best way to ensure everything is reproducible (not only on your machine...) and inspectable

Option 2: A Minima Investment

Your co-authors let you manage the code and the results all by yourself but are ready to make efforts to edit your document.

They can:

- Edit the content of the article (Markdown or Org-Mode)

They can't:

- Re-execute the code
- Export and generate the final document

Option 3: "Defiant" Co-auteurs

Co-auteurs do not change their habits

- A separated *computational document* allows to produce all results and figures
- An other (*standard*) document includes generated figures

Everything is stored, documented and can be re-computed in your computational document!

Publishing / Sharing Your Document

Rpubs

- Great for a quick sharing but no durability

Dropbox and alike

- ~~Durability~~, access ??, ...

Gitlab/Github/...

1. Go public (along with the history!)
2. Clean up the repository and archive the current state in a companion website

Companion websites

- Runmycode, Editors, ...
- Article: HAL ; code and data: Figshare / Zenodo

Conclusion

Several options depending on:

- your co-authors
- technical constraints
- confidentiality/copyright constraints

Where are we?

Entering Behind the Scenes: Computational Documents

1. A few Recent Controversial Studies
2. Why is This so Difficult?
3. Computational Document: Principles
4. Hands on.
   - Jupyter
   - Rstudio
   - Org-Mode
5. Collaborating
6. Comparative study

Comparative Study

A computational document. What for ?

A Lecture or a Tutorial

A Jupyter notebook

- Easy to use for students
- Dynamic document

A Journal

My journal with org-mode

- A single author
- Chronological organization
- Labels
- Notes, links, code

A Laboratory Notebook

A laboratory notebook with org-mode

- Semantic organization
- Conventions
- Several authors
- Labels per author, experiment, etc.

A Reproducible Article

An ongoing article

- Several authors
- Regenerate figures
- Track back the sources

Technical Differences

	Origin	Technology	Usage	Browsing	Format	Article?
Jupyter	2001	Web App., Python	Easy	Limited	JSON	Difficult
Rstudio/knitr	2011/2014	IDE, Java/R	Easy	Limited	Rmd	Yes
Org-Mode	1976/2008	Editeur, EmacsLisp	More complex	Powerful	Org	Yes

Technology does not really matter. You need to:

- collect information
- organize it and prepare for exploitation
- make it available