La vitrine et l'envers du décor : Le document computationnel

Arnaud Legrand

LIG, Univ. Grenoble Alpes, CNRS, Inria

arnaud.legrand@imag.fr

September 3, 2018

Plan du cours

M2-S0: Le document computationnel

M2-S1: Exemples récents d'études assez discutées

M2-S2: Pourquoi est-ce difficile ?

M2-S3: Le document computationnel : principe

M2-S4A: Prise en main de l'outil (Jupyter)

M2-S4B: Prise en main de l'outil (Rstudio)

M2-S4C: Prise en main de l'outil (Org-Mode)

M2-S5: Travailler avec les autres

M2-S6: Analyse comparée des différents outils

Où en sommes nous ?

M2-S0: Le document computationnel

M2-S1: Exemples récents d'études assez discutées

M2-S2: Pourquoi est-ce difficile ?

M2-S3: Le document computationnel : principe

M2-S4A: Prise en main de l'outil (Jupyter)

M2-S4B: Prise en main de l'outil (Rstudio)

M2-S4C: Prise en main de l'outil (Org-Mode)

M2-S5: Travailler avec les autres

M2-S6: Analyse comparée des différents outils

La vitrine et l'envers du décor: Le document computationnel

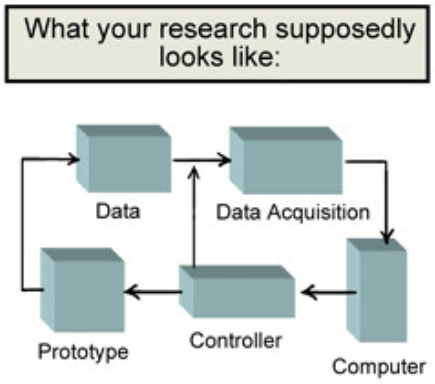


Figure 1. Experimental Diagram

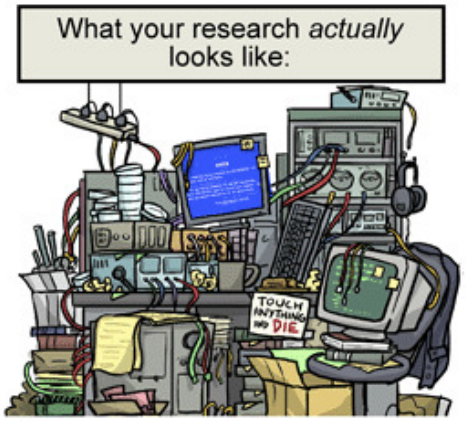


Figure 2. Experimental Mess

1. Exemples récents d'études assez discutées
2. Pourquoi est-ce difficile ?
3. Le document computationnel : principe
4. Prise en main de l'outil Au choix :
   - Jupyter
   - Rstudio
   - Org-Mode
5. Travailler avec les autres
6. Analyse comparée des différents outils

Où en sommes nous ?

M2-S0: Le document computationnel

M2-S1: Exemples récents d'études assez discutées

M2-S2: Pourquoi est-ce difficile ?

M2-S3: Le document computationnel : principe

M2-S4A: Prise en main de l'outil (Jupyter)

M2-S4B: Prise en main de l'outil (Rstudio)

M2-S4C: Prise en main de l'outil (Org-Mode)

M2-S5: Travailler avec les autres

M2-S6: Analyse comparée des différents outils

Le document computationnel

1. Exemples récents d'études assez discutées
2. Pourquoi est-ce difficile ?
3. Le document computationnel : principe
4. Prise en main de l'outil. Aux choix :
   - Jupyter
   - Rstudio
   - Org-Mode
5. Travailler avec les autres
6. Analyse comparée des différents outils

Exemples récents d'études assez discutées

Économie : politiques d'austérité (1/2)

2010

Lorsque la dette extérieure brute atteint 60 pourcents du PIB, la croissance annuelle d'un pays diminue de deux pourcents.
[..] pour des niveaux de dette extérieure dépassant 90 pourcents du PIB, la croissance annuelle est à peu près divisée par deux.

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

Économie : politiques d'austérité (2/2)

2013

En utilisant leurs feuilles Excel, nous avons identifié des erreurs de programmation, des exclusions de certaines données, et des pondérations statistiques non conventionnelles.

– Herndon, Ash et Pollin

R&R combinent des données de siècles différents, des régimes de changes différents, des dettes privées et publiques, et des dettes exprimées en monnaies étrangères et nationales.

– Wray

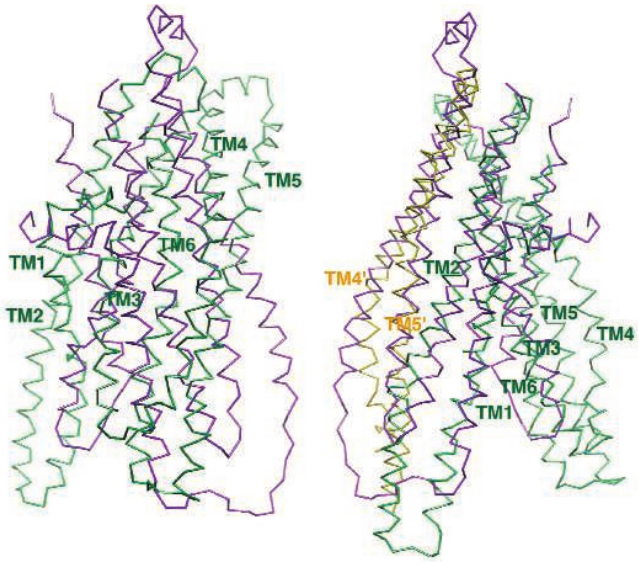
IRM fonctionnelle

- 2010 : Bennett et al. et le saumon mort ☺
- 2016 : Eklund, Nichols, and Knutsson. A bug in fmri software could invalidate 15 years of brain research (*40 000 articles*)
- 2016 : Mais c'est plus subtil que ça. Nichols. $\approx$ *3 600 études concernées*

Des méthodes statistiques à améliorer mais pas de remise en cause fondamentale.

Les fausses structures de protéines

Geoffrey Chang : étude de la structure de protéines présentes dans des bactéries résistant aux antibiotiques.
MsbA de Escherichia Coli (Science, 2001), Vibrio cholera (Mol. Biology, 2003), Salmonella typhimurium (Science, 2005)



2006 : Incohérences, alertes, puis 5 rétractations

> *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.*
>
> – *une "erreur de programmation"*

Crise de foi ?

- Oncologie : "*plus de la moitié des études publiées, même dans des journaux prestigieux, ne peuvent être reproduites en laboratoire industriel*"
- Psychologie : "réplication d'une centaine d'articles *seulement un tiers de résultats cohérents*"

Lanceurs d'alerte ou institutions malades ?

La remise en cause fait partie du processus scientifique

Crise de foi ?

- Oncologie : "*plus de la moitié des études publiées, même dans des journaux prestigieux, ne peuvent être reproduites en laboratoire industriel*"
- Psychologie : "réplication d'une centaine d'articles *seulement un tiers de résultats cohérents*"

Lanceurs d'alerte ou institutions malades ?

La remise en cause fait partie du processus scientifique

Tout comme la rigueur et la transparence. . .

Où en sommes nous ?

M2-S0: Le document computationnel

M2-S1: Exemples récents d'études assez discutées

M2-S2: Pourquoi est-ce difficile ?

M2-S3: Le document computationnel : principe

M2-S4A: Prise en main de l'outil (Jupyter)

M2-S4B: Prise en main de l'outil (Rstudio)

M2-S4C: Prise en main de l'outil (Org-Mode)

M2-S5: Travailler avec les autres

M2-S6: Analyse comparée des différents outils

Le document computationnel

1. Exemples récents d'études assez discutées
2. Pourquoi est-ce difficile ?
3. Le document computationnel : principe
4. Prise en main de l'outil. Au choix :
   - Jupyter
   - Rstudio
   - Org-Mode
5. Travailler avec les autres
6. Analyse comparée des différents outils

Pourquoi est-ce difficile ?

1) Le manque d'informations

Expliciter :

- Sources et données

  Données non disponibles = résultats difficiles à vérifier

- Choix

  Choix non expliqués = choix suspicieux

Le cahier de laboratoire peut vous aider

2) L'ordinateur, source d'erreurs

- Point and click :
- Les tableurs : erreurs de programmation et de manipulation de données
  - `Membrane-Associated Ring Finger (C3HC4) 1, E3 Ubiquitin Protein Ligase` → `MARCH1` → 2016-03-01 → 1456786800
  - `2310009E13` → 2.31E+19
- Pile logicielle complexe
- Bug : *Programmer, c'est dur !*

L'informatique, seule responsable ?

Le manque de rigueur et d'organisation

- Pas de backup
- Pas d'historique
- Pas de contrôle qualité

Une dimension culturelle et sociale

Article = version simplifiée de la procédure

Tracer toutes ces informations et les rendre disponibles = investissement conséquent

Si personne n'exige/n'inspecte ces informations, à quoi bon s'embêter ?

Tout rendre public ?

- Les *faiblesses* deviendraient évidentes
- Quelqu'un pourrait trouver une *erreur*
- Quelqu'un pourrait en tirer avantage à ma place
- *Les données peuvent être sensibles*

Donnons nous les moyens que tout soit inspectable à la demande

Outils à éviter et alternatives

- Outils, formats, et services propriétaires
  1. ~~Excel, Word, Evernote~~
     - Markdown, Org-mode, CSV, HDF5, ...
  2. ~~SAS, Minitab, matlab, mathematica, ...~~
     - Scilab, R, Python, ...
  3. ~~Dropbox, cahiers de labo en ligne propriétaires, ...~~
     - Framadrop, GitLab/GitHub, ...
- Outils "intuitifs"
  - ~~tableurs, interfaces graphiques, exploration interactive~~
    - apprendre à se contrôler... 
    - R, Python, ...

Changement de paradigme

1. Manque d'information, problème d'accès aux données
2. Erreurs de calcul
3. Manque de rigueur scientifique et technique

Expliciter augmente les chances de trouver les erreurs et de les éliminer

Où en sommes nous ?

M2-S0: Le document computationnel

M2-S1: Exemples récents d'études assez discutées

M2-S2: Pourquoi est-ce difficile ?

M2-S3: Le document computationnel : principe

M2-S4A: Prise en main de l'outil (Jupyter)

M2-S4B: Prise en main de l'outil (Rstudio)

M2-S4C: Prise en main de l'outil (Org-Mode)

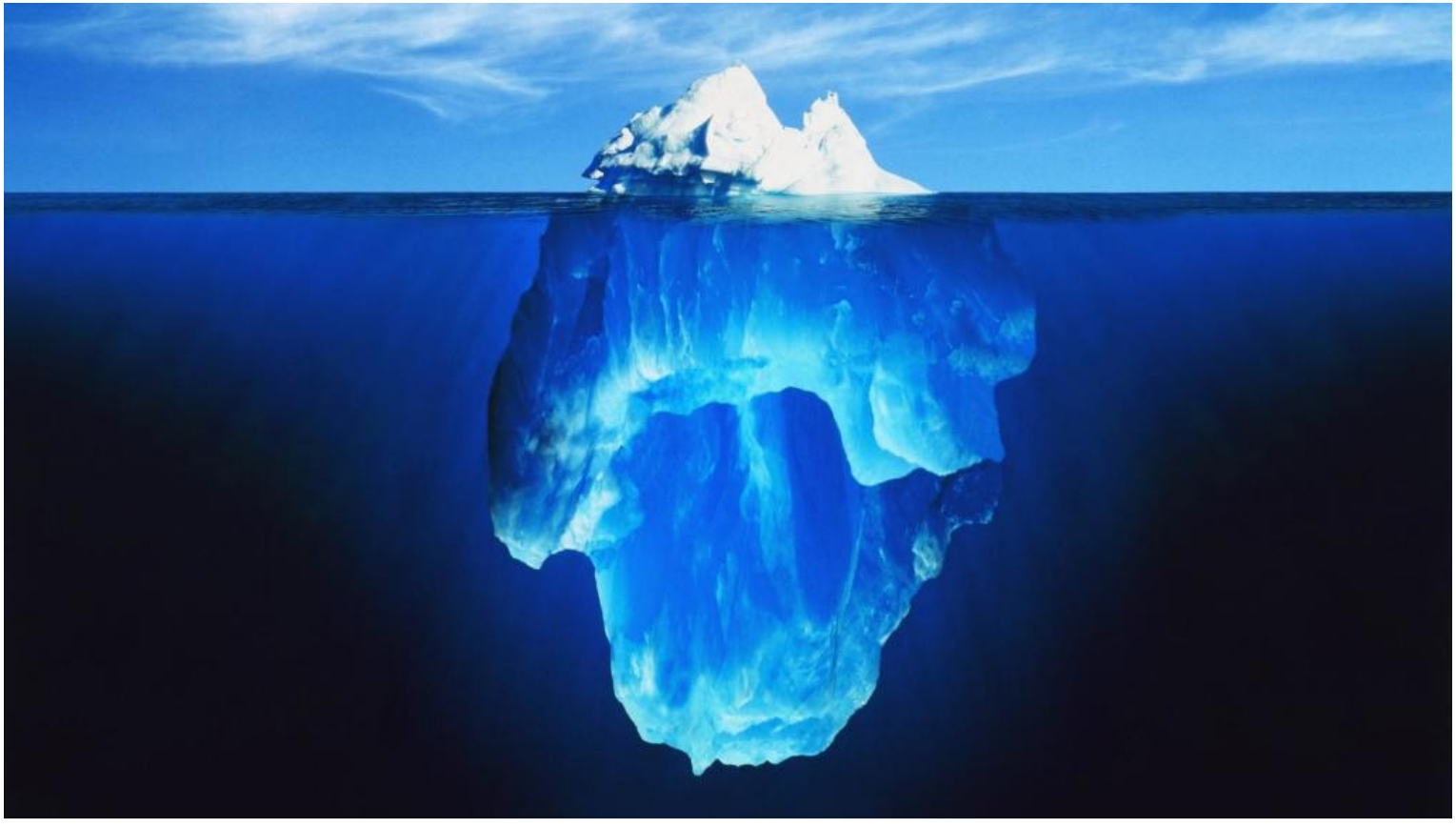
M2-S5: Travailler avec les autres

M2-S6: Analyse comparée des différents outils

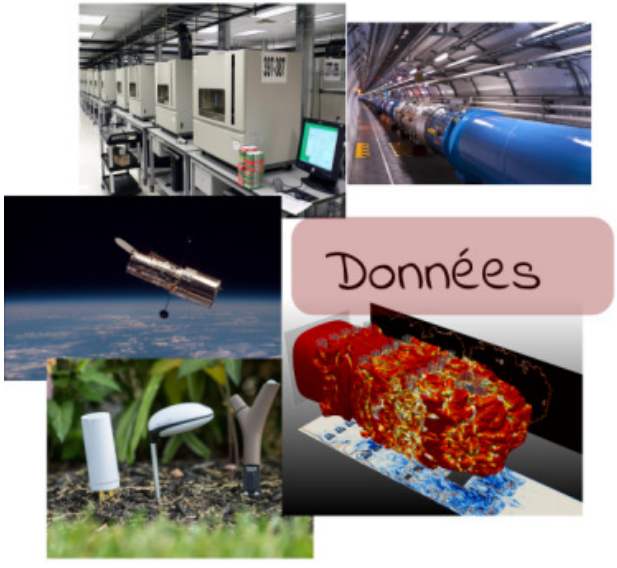
Le document computationnel

1. Exemples récents d'études assez discutées
2. Pourquoi est-ce difficile ?
3. Le document computationnel : principe
4. Prise en main de l'outil. Au choix :
   - Jupyter
   - Rstudio
   - Org-Mode
5. Travailler avec les autres
6. Analyse comparée des différents outils

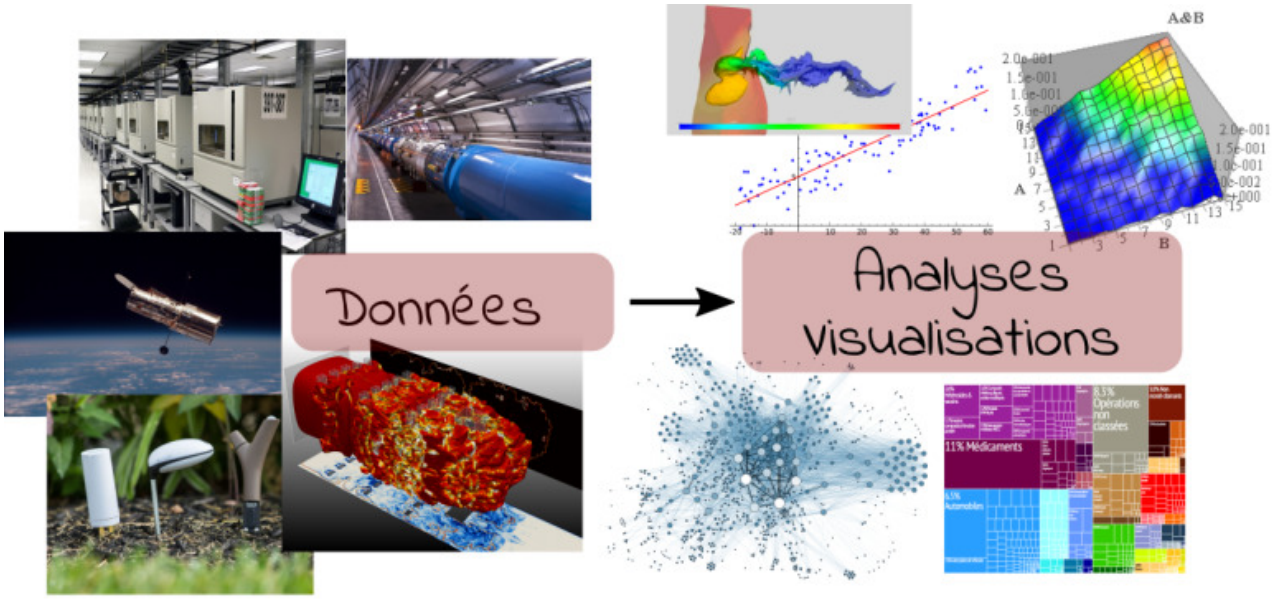
Le document computationnel : principe



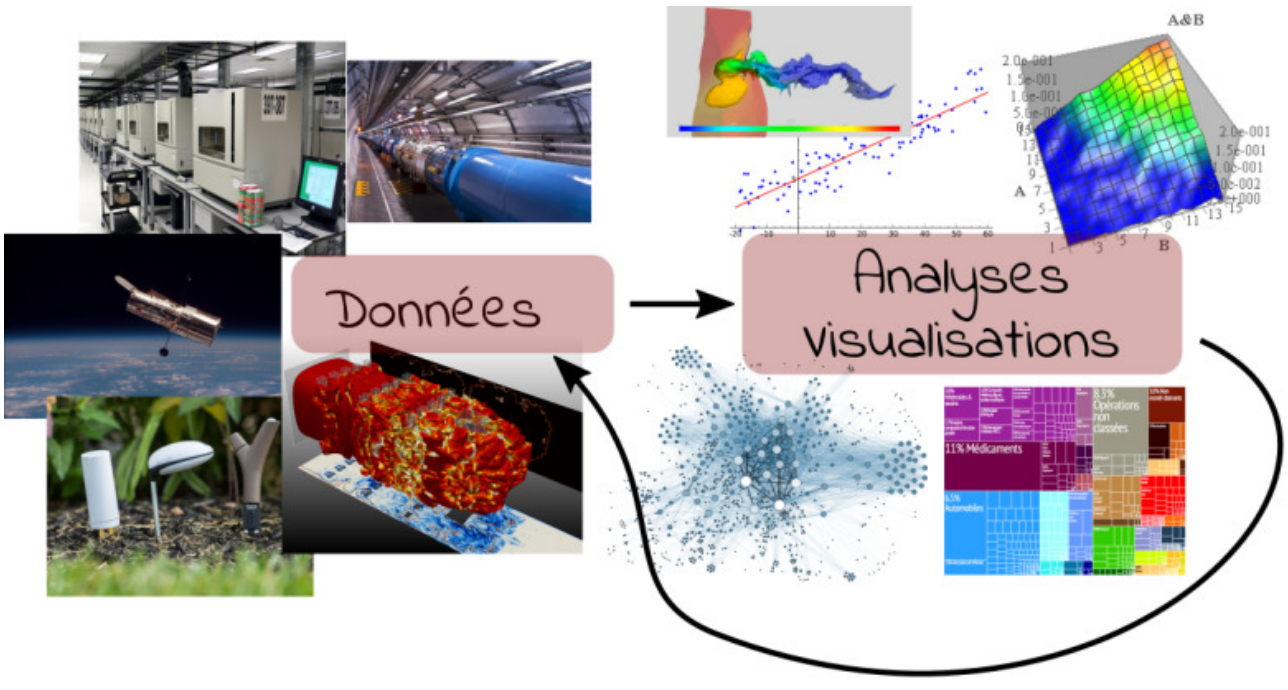
La science moderne



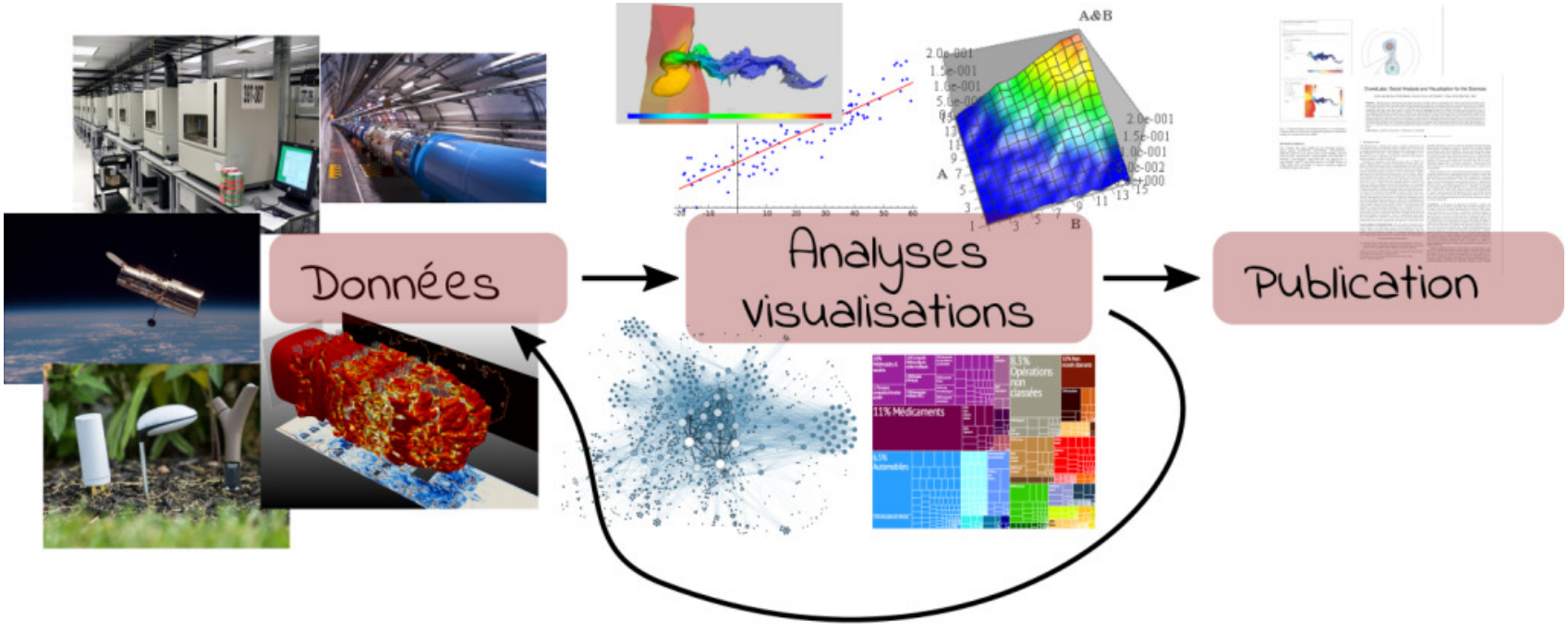
La science moderne



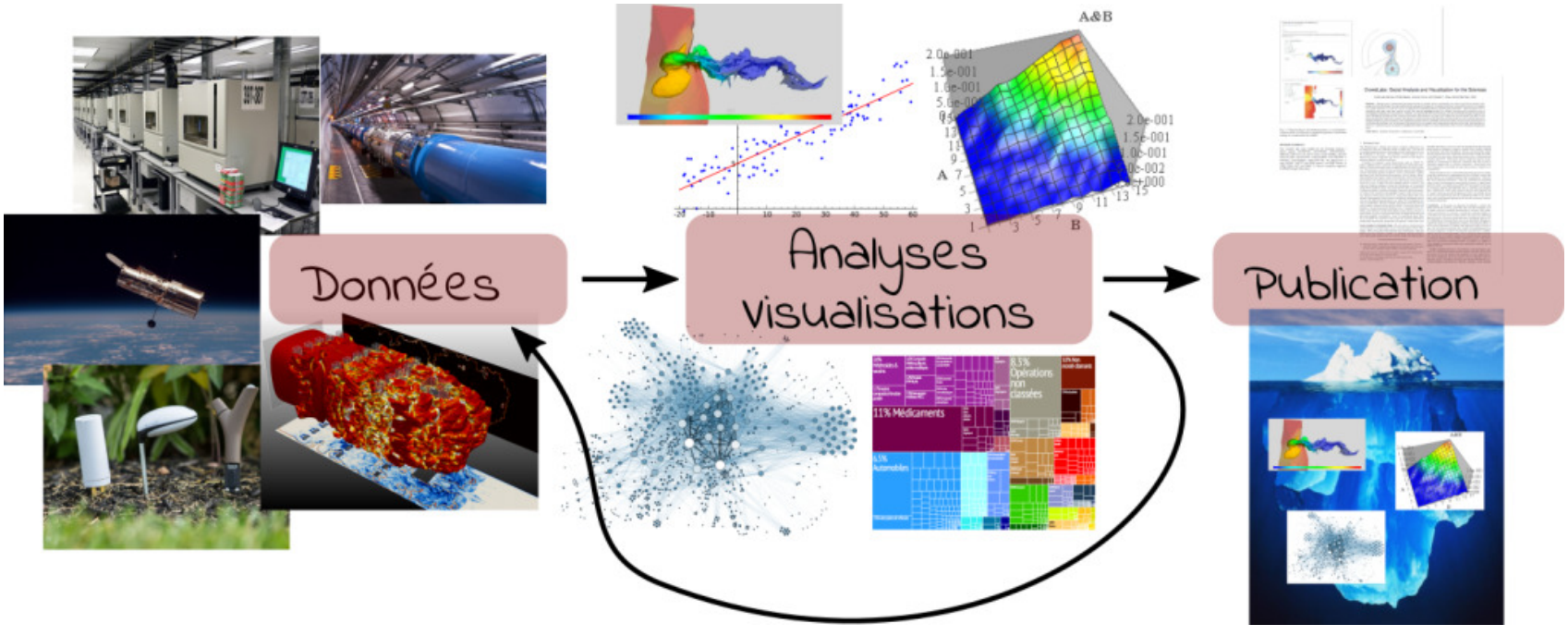
La science moderne



La science moderne



La science moderne



Objectifs méthodologiques

Garder trace afin de :

- Inspecter : justifier/comprendre
- Refaire : vérifier/corriger/réutiliser

La vitrine. . . et l'envers du décor

Un document computationnel

Mon ordinateur m'indique que π vaut *approximativement*

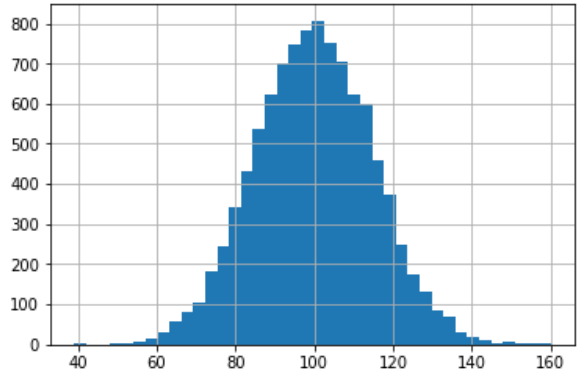
```
3.141592653589793
```

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... ☺).



La vitrine. . . et l'envers du décor

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 π 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 π (si ce n'est une constante de normalisation... ☺).

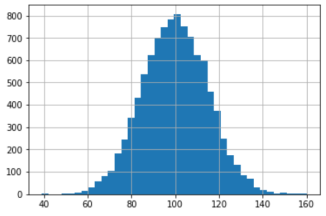
In [3]:

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*

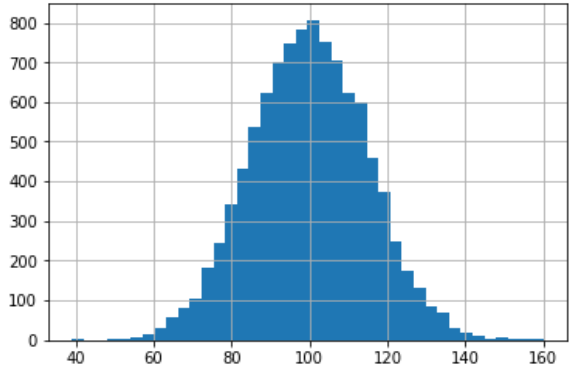
```
3.141592653589793
```

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... ☺).



La vitrine. . . et l'envers du décor

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... ☺).
```

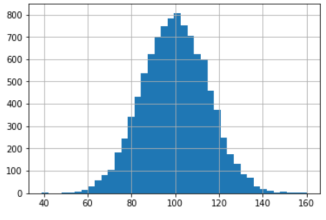
In [3]:

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*

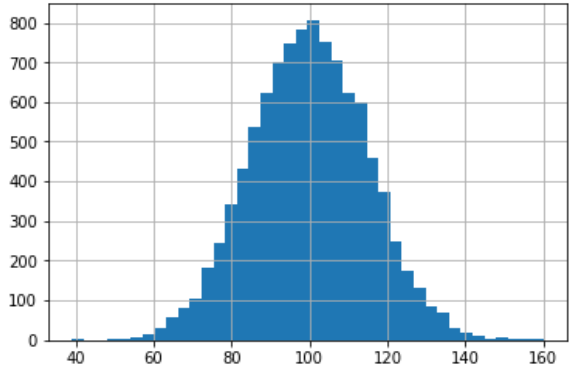
```
3.141592653589793
```

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... ☺).



La vitrine. . . et l'envers du décor

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

Mon ordinateur m'indique que π vaut *approximativement*

In [1]:
```
from math import *
print(pi)
```
```
3.141592653589793
```

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

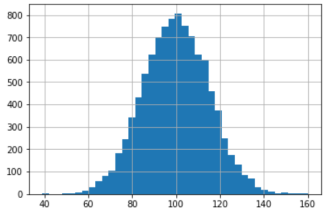
In [2]:
```
import numpy as np
N = 1000000
x = np.random.uniform(size=N, low=0, high=1)
theta = np.random.uniform(size=N, low=0, high=pi/2)
2/(sum((x+np.sin(theta))>1)/N)
```
Out[2]:
```
3.1437198694098765
```

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... ☺).

In [3]:
```
%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

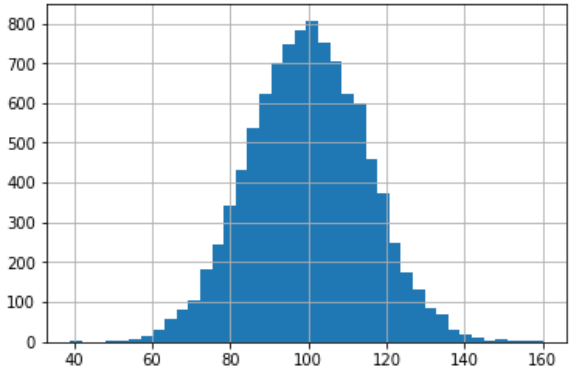
Mon ordinateur m'indique que π vaut *approximativement*

```
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... ☺).



La vitrine. . . et l'envers du décor

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... ☺).
```

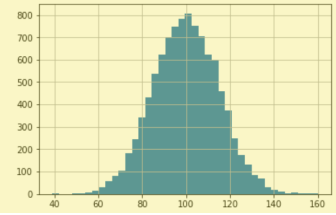
In [3]:

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*

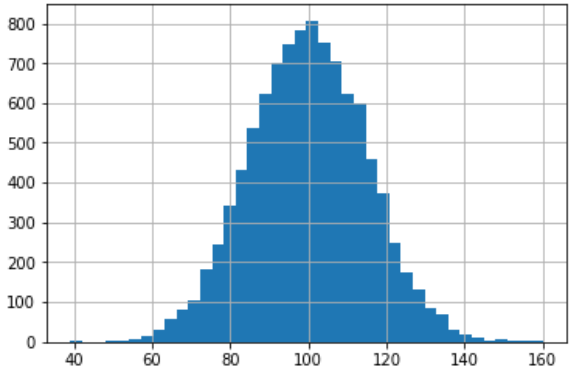
```
3.141592653589793
```

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... ☺).



La vitrine. . . et l'envers du décor

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... ☺).
```

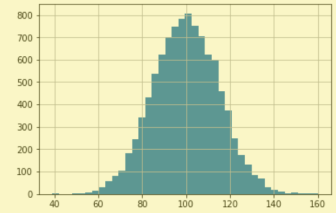
In [3]:

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*

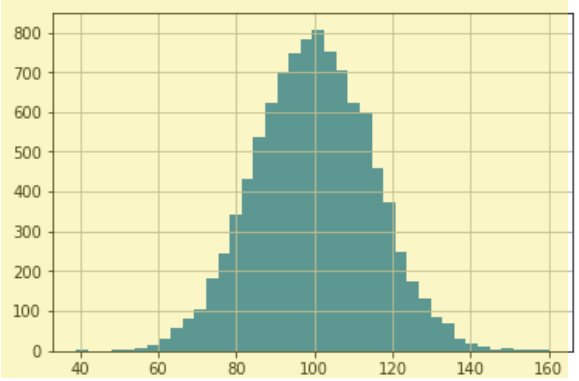
```
3.141592653589793
```

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... ☺).



Les différents outils

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

Principes identiques	Différences
• 1 seul document (explications, code, resultats)	• Syntaxe
• Session	• Interopérabilité
• Export	• Contrôle export

Où en sommes nous ?

M2-S0: Le document computationnel

M2-S1: Exemples récents d'études assez discutées

M2-S2: Pourquoi est-ce difficile ?

M2-S3: Le document computationnel : principe

M2-S4A: Prise en main de l'outil (Jupyter)

M2-S4B: Prise en main de l'outil (Rstudio)

M2-S4C: Prise en main de l'outil (Org-Mode)

M2-S5: Travailler avec les autres

M2-S6: Analyse comparée des différents outils

Le document computationnel

1. Exemples récents d'études assez discutées
2. Pourquoi est-ce difficile ?
3. Le document computationnel : principe
4. Prise en main de l'outil. Au choix :
   - Jupyter
   - Rstudio
   - Org-Mode
5. Travailler avec les autres
6. Analyse comparée des différents outils

Prise en main de l'outil : Jupyter

Lancement

- Ouverture d'un document
- Description rapide
- Sauvegarde
- Aide

Exécution des blocs

- Exécution et récupération des résultats
- Ajout d'un bloc
- Attention à l'ordre
  - Notion de session
  - Incohérences possibles
  - Tout réexécuter depuis le début

Raccourcis clavier, auto-complétion, et Ipython magic

- Raccourcis clavier `<h>`
- Complétion Python (exemple de numpy)
- `%matplotlib`, `%lsmagic`

Utiliser d'autres langages

- Exemple pour R :
  - `%load_ext rpy2.ipython`
  - `%%R %%sh %%perl`
- Interaction entre `R` et `Python` possible
  - mais fragile...

Production et partage du document final

- Résultats stockés dans le document
  - $\rightarrow$ visibles dans gitlab
  - `git pull/push`
- Export HTML/PDF classique

Préparer un document

- Hide-code plugin
- `%%latex` et `%%html`
- Personnaliser les *exporters* de NBConvert

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

Ce qu'il faut retenir

- Beaucoup d'informations en peu de temps
- Mettez en pratique !

Où en sommes nous ?

M2-S0: Le document computationnel

M2-S1: Exemples récents d'études assez discutées

M2-S2: Pourquoi est-ce difficile ?

M2-S3: Le document computationnel : principe

M2-S4A: Prise en main de l'outil (Jupyter)

M2-S4B: Prise en main de l'outil (Rstudio)

M2-S4C: Prise en main de l'outil (Org-Mode)

M2-S5: Travailler avec les autres

M2-S6: Analyse comparée des différents outils

Le document computationnel

1. Exemples récents d'études assez discutées
2. Pourquoi est-ce difficile ?
3. Le document computationnel : principe
4. Prise en main de l'outil. Au choix :
   - Jupyter
   - Rstudio
   - Org-Mode
5. Travailler avec les autres
6. Analyse comparée des différents outils

Prise en main de l'outil : Rstudio

Lancement

- Ouverture d'un document
- Description rapide
- Sauvegarde
- Aide

Exécution des blocs

- Exécution et récupération des résultats
- Ajout d'un bloc
- Attention à l'ordre (notion de session, incohérences possibles)
- Tout réexécuter depuis le début

Raccourcis clavier et auto-complétion

- Raccourcis claviers
- Complétion R
- Folding

Production et partage du document final

- Knit
- Partage à peu de frais via rpubs

Contrôler la visibilité du code et des résultats

- Complétion (paramètres des blocs)

Utiliser un style particulier

- pdf, LaTeX
- html
- word/office

Possibilité de faire du LaTeX (R Sweave : `Rnw`) ou du html (R html : `Rhtml`) directement pour avoir un contrôle parfait.

Utiliser d'autres langages

- Ajout et exécution d'un bloc Python
- Attention, pas de session !
  - Interaction uniquement via fichiers et dans de longs blocs

Ce qu'il faut retenir

- Beaucoup d'informations en peu de temps
- Mettez en pratique !

Où en sommes nous ?

M2-S0: Le document computationnel

M2-S1: Exemples récents d'études assez discutées

M2-S2: Pourquoi est-ce difficile ?

M2-S3: Le document computationnel : principe

M2-S4A: Prise en main de l'outil (Jupyter)

M2-S4B: Prise en main de l'outil (Rstudio)

M2-S4C: Prise en main de l'outil (Org-Mode)

M2-S5: Travailler avec les autres

M2-S6: Analyse comparée des différents outils

Le document computationnel

1. Exemples récents d'études assez discutées
2. Pourquoi est-ce difficile ?
3. Le document computationnel : principe
4. Prise en main de l'outil. Au choix :
   - Jupyter
   - Rstudio
   - Org-Mode
5. Travailler avec les autres
6. Analyse comparée des différents outils

Prise en main de l'outil : Org Mode

Lancement

- Ouverture d'un document
- Description rapide
  - Folding / Navigation
  - Restructuration
- Sauvegarde
- Aide

Exécution des blocs

- Ajout d'un bloc R
- Exécution et récupération des résultats
- Attention à l'ordre
  - Notion de session
  - Incohérences possibles
  - Tout réexécuter depuis le début

Raccourcis clavier

- Bloc expansion
  - R graphique
  - Python, Perl, ...
  - Shell session
- Plusieurs sessions, plusieurs langages !
- Communication entre langages possible

Production et partage du document final

- Git Commit
  - Attention aux fichiers produits
- Export
- Visibilité du code et des résultats
  - Sections cachées

Utiliser un style particulier

- pdf, LaTeX
- html
- Possibilité de reprendre le contrôle

Ce qu'il faut retenir

- Beaucoup d'informations en peu de temps
- Apprivoiser les raccourcis claviers avec la première entrée du journal
- Mettez en pratique !

Où en sommes nous ?

- M2-S0: Le document computationnel
- M2-S1: Exemples récents d'études assez discutées
- M2-S2: Pourquoi est-ce difficile ?
- M2-S3: Le document computationnel : principe
- M2-S4A: Prise en main de l'outil (Jupyter)
- M2-S4B: Prise en main de l'outil (Rstudio)
- M2-S4C: Prise en main de l'outil (Org-Mode)
- M2-S5: Travailler avec les autres
- M2-S6: Analyse comparée des différents outils

Le document computationnel

1. Exemples récents d'études assez discutées
2. Pourquoi est-ce difficile ?
3. Le document computationnel : principe
4. Prise en main de l'outil. Au choix :
   - Jupyter
   - Rstudio
   - Org-Mode
5. Travailler avec les autres
6. Analyse comparée des différents outils

Travailler avec les autres

Préparer un document pour un journal

Pré-requis pour faire un pdf :

- Actuellement caché. En interne *pandoc*, *knitr* ou *emacs/org-mode*
- LaTeX installé
  Export office/word possible dans jupyter mais à configurer. Sinon

export html...

Dans tous les cas :

- Besoin de cacher certaines cellules
- Utiliser le bon style

Produire un tel document demande d'avoir un environnement parfaitement configuré

Convaincre vos co-auteurs

Face à cette complexité, plusieurs réactions :

1. Pas grave, c'est génial ! Je m'y mets !
2. Euh... c'est bien. Mais je n'ai pas le temps d'apprendre...
3. Un nouvel outil ? Jamais !

$\rightsquigarrow$ différentes organisations possibles

Option 1 : les co-auteurs enthousiastes

Il faudra assurer le service après-vente :

- Compatibilité avec les différents environnements
- Gérer cette complexité (Jupyter/Rstudio/Emacs, Git, ...)

C'est la meilleure façon de s'assurer que tout est reproductible et inspectable (et pas uniquement sur votre propre machine...)

Option 2 : investissement a minima

Vos co-auteurs vous laissent gérer le code, les résultats mais adoptent votre style de document.

Ils peuvent :

- Éditer le texte de l'article (Markdown ou Org-Mode)

Ils ne peuvent pas :

- Recalculer
- Exporter et voir le document final

Option 3 : les co-auteurs "réfractaires"

Les co-auteurs ne changent pas leurs habitudes

- Un document *computationnel* séparé produit tous les résultats et toutes les figures
- Un autre document (*classique*) inclut les figures générées

Mais tout est conservé, documenté et recalculable dans votre document computationnel !

Publier / partager votre document

Rpubs

- Parfait pour partage rapide, pas pérenne

Dropbox et autres

- ~~Pérénité~~, accès ??, ...

Gitlab/Github/...

1. Rendre public (tout l'historique !)
2. Faire le ménage et archiver l'état courant dans un site compagnon

Sites compagnons

- Runmycode, Éditeurs, ...
- Article : HAL ; code et données : Figshare / Zenodo

Conclusion

Plusieurs modalités possibles en fonction de :

- vos co-auteurs
- vos contraintes techniques
- vos contraintes de confidentialité/copyright

Où en sommes nous ?

M2-S0: Le document computationnel

M2-S1: Exemples récents d'études assez discutées

M2-S2: Pourquoi est-ce difficile ?

M2-S3: Le document computationnel : principe

M2-S4A: Prise en main de l'outil (Jupyter)

M2-S4B: Prise en main de l'outil (Rstudio)

M2-S4C: Prise en main de l'outil (Org-Mode)

M2-S5: Travailler avec les autres

M2-S6: Analyse comparée des différents outils

Le document computationnel

1. Exemples récents d'études assez discutées
2. Pourquoi est-ce difficile ?
3. Le document computationnel : principe
4. Prise en main de l'outil. Au choix :
   - Jupyter
   - Rstudio
   - Org-Mode
5. Travailler avec les autres
6. Analyse comparée des différents outils

Analyse comparée des différents outils

Un document computationnel, mais pour quoi faire exactement ?

Un cours ou un tutoriel

Un notebook Jupyter

- Facile à prendre en main
- Document dynamique

Un journal

Mon journal en org-mode

- Un seul auteur
- Organisation chronologique
- Étiquettes
- Notes, liens, code

Un cahier de laboratoire

Un cahier de laboratoire en org-mode

- Organisation sémantique
- Conventions
- Plusieurs auteurs
- Étiquettes pour auteurs, expériences, etc.

Un article reproductible

Un article en cours

- Plusieurs auteurs
- Regénérer les figures
- Revenir aux sources

Différences techniques

	Origine	Technologie	Utilisation	Navigation	Format	Article?
Jupyter	2001	Web App., Python	Facile	Limitée	JSON	Difficile
Rstudio/knitr	2011/2014	IDE, Java/R	Facile	Limitée	Rmd	Oui
Org-Mode	1976/2008	Editeur, EmacsLisp	Plus complexe	Puissante	Org	Oui

L'outil importe peu, ce qui importe, c'est :

- collecter l'information
- l'organiser et la rendre exploitable
- la rendre disponible