

Vers une étude reproductible : la réalité du terrain

Christophe Pouzat, Arnaud Legrand, Konrad Hinszen

3 septembre 2018

Plan du cours

M4-S0 : Vers une étude reproductible : la réalité du terrain

M4-S1 : L'enfer des données

M4-S2 : L'enfer du logiciel

M4-S3 : L'enfer du calcul

M4-S4 : Conclusion

Où en sommes nous?

M4-S0 : Vers une étude reproductible : la réalité du terrain

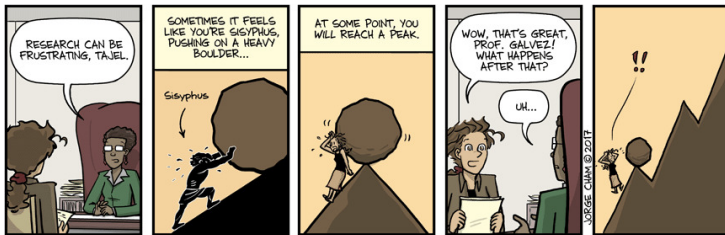
M4-S1 : L'enfer des données

M4-S2 : L'enfer du logiciel

M4-S3 : L'enfer du calcul

M4-S4 : Conclusion

L'enfer de la Recherche Reproductible



Où en sommes nous?

M4-S0 : Vers une étude reproductible : la réalité du terrain

M4-S1 : L'enfer des données

M4-S2 : L'enfer du logiciel

M4-S3 : L'enfer du calcul

M4-S4 : Conclusion

Deux « nouveaux » problèmes

Lorsque nous commençons à travailler sur de « vraies » données nous nous trouvons généralement confrontés à deux problèmes :

- ▶ les données sont de nature « diverse »
- ▶ les données occupent un grand espace mémoire

Les données « non homogènes »

- ▶ Les données grippales du module 3 se prêtent bien à une présentation en table (objet à 2 dimensions)
- ▶ Souvent la forme table doit être **abandonnée** car :
 - ▶ les colonnes n'ont pas la même longueur
 - ▶ les données peuvent être des suites chronologiques **et** des images, etc.

Les données sont « trop grosses »

- ▶ **Format texte** pas toujours approprié pour des nombres
- ▶ Choix d'un **format binaire** car :
 - ▶ Les nombres prennent moins de place mémoire
 - ▶ Nombres en format texte \Rightarrow conversion en binaire pour calculs

Ce qu'il faut garder du format texte : les métadonnées

- ▶ Le format texte permet de stocker les données **et** tout le reste. . .
- ▶ ⇒ ajouter des informations sur les données :
 - ▶ provenance
 - ▶ date d'enregistrement
 - ▶ source
 - ▶ etc.
- ▶ Ces informations sur les données sont ce qu'on appelle les **métadonnées**
- ▶ Elles sont vitales pour la mise en œuvre de la recherche reproductible

Ce qu'il faut garder du format texte : le boutisme

- ▶ Format texte « universel »
- ▶ Formats binaires dépendent de l'architecture ou de l'OS
- ▶ 1010, codé sur 4 bits peut être lu comme :
 - ▶ $1 \times 1 + 0 \times 2 + 1 \times 4 + 0 \times 8 = 5$, c'est le **petit-boutisme**
 - ▶ $1 \times 8 + 0 \times 4 + 1 \times 2 + 0 \times 1 = 10$, c'est le **gros-boutisme**
- ▶ Un stockage binaire pour la recherche reproductible doit spécifier le boutisme

Des formats binaires, pour données composites, permettant la sauvegarde de métadonnées

Rechercher des formats binaires pour :

- ▶ travailler avec de grosses données de natures différentes
- ▶ stocker des métadonnées avec les données
- ▶ avoir un boutisme fixé **une fois pour toute**

'FITS' et 'HDF5'

- ▶ Le Flexible Image Transport System ('FITS'), créé en 1981 est toujours régulièrement mis à jour
- ▶ Le Hierarchical Data Format ('HDF'), développé au National Center for Supercomputing Applications, en est à sa cinquième version, 'HDF5'

'FITS'

- ▶ 'FITS' introduit et mis à jour par les astrophysiciens
- ▶ Format suffisamment général pour utilisation dans différents contextes

Anatomie d'un fichier 'FITS'

- ▶ 1 ou plusieurs segments : **Header/Data Units (HDUs)**
- ▶ HDU constituée :
 - ▶ d'une en-tête (**Header Unit**) suivie, **mais ce n'est pas obligatoire**, par
 - ▶ des données (**Data Unit**)
- ▶ En-tête = paires mots clés / valeurs → **métadonnées**
- ▶ Données tableaux binaires (une à 999 dimensions) ou tables (texte ou binaire)

Manipulation des fichiers 'FITS'

- ▶ Les développeurs du format fournissent une bibliothèque 'C' et des programmes associés (faciles à utiliser)
- ▶ Les utilisateurs de 'Python' pourront utiliser 'PyFITS'
- ▶ Les utilisateurs de 'R' pourront utiliser 'FITSio'

'HDF5'

- ▶ Organisation hiérarchique, ressemble à une arborisation de fichiers
- ▶ Élément structurant : le **group** (répertoire) contient un ou plusieurs **datasets** (jeux de données)
- ▶ Les **groups** peuvent être imbriqués
- ▶ Les métadonnées n'ont pas de structure imposée
- ▶ Les données n'ont pas de structure imposée — elles peuvent être des textes.

Manipulation des fichiers 'HDF5'

- ▶ Format plus souple \Rightarrow bibliothèque 'C' plus compliquée que son équivalent 'FITS'
- ▶ La bibliothèque distribuée avec 'HDFView' : outil puissant d'exploration et de visualisation
- ▶ 'Python' dispose d'une interface très complète avec 'h5py'
- ▶ Il y a trois paquets 'R' : 'h5', 'hdf5r' et 'rhdf5'

L'archivage

Git(hub/lab/...) : pas bien adapté au stockage de données



Conclusions

- ▶ Vraies données \Rightarrow problèmes de taille et de structure
- ▶ Vraies données complexes \Rightarrow métadonnées
- ▶ 'FITS' et 'HDF5' = solutions pratiques
- ▶ En complexité et flexibilité : 'FITS' < 'HDF5'
- ▶ Plateformes d'archivage \Rightarrow stockage pérenne accessible à tous

Où en sommes nous?

M4-S0 : Vers une étude reproductible : la réalité du terrain

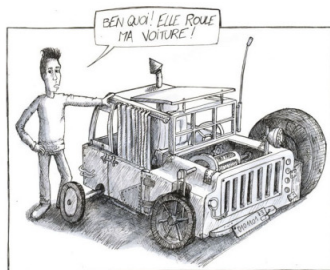
M4-S1 : L'enfer des données

M4-S2 : L'enfer du logiciel

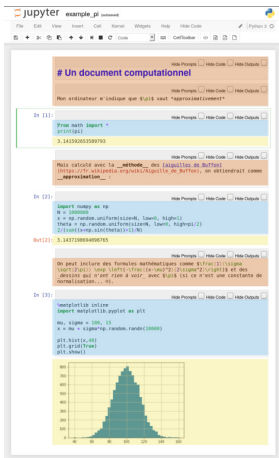
M4-S3 : L'enfer du calcul

M4-S4 : Conclusion

Passage à l'échelle



Des codes complexes...



The screenshot shows a Jupyter Notebook window titled "exemple_jf_ensemble". The notebook contains several cells:

- A title cell: "# Un document computationnel".
- A text cell: "Mon ordinateur m'indique que j'ai 3.142592653589793".
- A code cell (In [1]):

```
from math import *\nprint(pi)
```
- A text cell: "Mais calculé avec la méthode des aiguilles de Buffon! https://fr.wikipedia.org/wiki/Calcul_de_Buffon, on obtiendrait comme `approximation`".
- A code cell (In [2]):

```
import numpy as np\nN = 100000\ns = np.random.uniform(size=N, low=0, high=1)\nx1 = np.random.uniform(size=N, low=1, high=pi/2)\nx2 = (s*np.sqrt(1-x1**2))
```
- A text cell: "On peut inclure des formules mathématiques comme $\sqrt{\frac{1-x_1^2}{2}}$ et des choses qui n'ont rien à voir, avec `sqrt` ici ce n'est une constante de normalisation...".
- A code cell (In [3]):

```
import matplotlib\nimport matplotlib.pyplot as plt\nmu, sigma = 100, 15\nx = mu + sigma*np.random.randn(10000)\nplt.hist(x, 40)\nplt.grid(True)\nplt.show()
```

The final cell displays a histogram of the generated data, showing a bell-shaped distribution centered around 100.

- ▶ Un vrai plat de spaghettis
 - ▶ Pas de vision d'ensemble
 - ▶ Interaction entre plusieurs langages = danger

Des codes complexes...

The screenshot displays a Jupyter Notebook interface for a notebook named 'analyse-syndrome-grippal'. The notebook contains several cells:

- Code Cell 1:** Imports the 'analyse-syndrome-grippal' dataset and displays its structure using `df.info()`.
- Code Cell 2:** Displays the first few rows of the dataset using `df.head()`.
- Table:** A table with columns 'date', 'etat', 'etat_c', 'etat_m', 'etat_n', 'etat_o', 'etat_p', 'etat_q', 'etat_r', 'etat_s', 'etat_t', 'etat_u', 'etat_v', 'etat_w', 'etat_x', 'etat_y', 'etat_z'. The 'etat' column is highlighted in blue.
- Code Cell 3:** Displays the first few rows of the dataset using `df.head()`.
- Code Cell 4:** Displays the first few rows of the dataset using `df.head()`.
- Code Cell 5:** Displays the first few rows of the dataset using `df.head()`.
- Code Cell 6:** Displays the first few rows of the dataset using `df.head()`.
- Code Cell 7:** Displays the first few rows of the dataset using `df.head()`.
- Code Cell 8:** Displays the first few rows of the dataset using `df.head()`.
- Code Cell 9:** Displays the first few rows of the dataset using `df.head()`.
- Code Cell 10:** Displays the first few rows of the dataset using `df.head()`.
- Code Cell 11:** Displays the first few rows of the dataset using `df.head()`.
- Code Cell 12:** Displays the first few rows of the dataset using `df.head()`.
- Code Cell 13:** Displays the first few rows of the dataset using `df.head()`.
- Code Cell 14:** Displays the first few rows of the dataset using `df.head()`.
- Code Cell 15:** Displays the first few rows of the dataset using `df.head()`.
- Code Cell 16:** Displays the first few rows of the dataset using `df.head()`.
- Code Cell 17:** Displays the first few rows of the dataset using `df.head()`.
- Code Cell 18:** Displays the first few rows of the dataset using `df.head()`.
- Code Cell 19:** Displays the first few rows of the dataset using `df.head()`.
- Code Cell 20:** Displays the first few rows of the dataset using `df.head()`.
- Code Cell 21:** Displays the first few rows of the dataset using `df.head()`.
- Code Cell 22:** Displays the first few rows of the dataset using `df.head()`.
- Code Cell 23:** Displays the first few rows of the dataset using `df.head()`.
- Code Cell 24:** Displays the first few rows of the dataset using `df.head()`.
- Code Cell 25:** Displays the first few rows of the dataset using `df.head()`.
- Code Cell 26:** Displays the first few rows of the dataset using `df.head()`.
- Code Cell 27:** Displays the first few rows of the dataset using `df.head()`.
- Code Cell 28:** Displays the first few rows of the dataset using `df.head()`.
- Code Cell 29:** Displays the first few rows of the dataset using `df.head()`.
- Code Cell 30:** Displays the first few rows of the dataset using `df.head()`.

The notebook also features a bar chart at the bottom, showing the distribution of a variable. The x-axis is labeled 'etat' and the y-axis is labeled 'count'. The bars represent the frequency of each state.

- ▶ Un vrai plat de spaghettis
 - ▶ Pas de vision d'ensemble
 - ▶ Interaction entre plusieurs langages = danger

Des codes complexes...

The screenshot shows a Jupyter notebook interface with a title bar that reads "analyse-système-grépal" and "sur OpenStack 401 en langage python". The main content area displays a large table with multiple columns and rows of data, likely representing system metrics or logs. The table is dense and spans most of the page.

This screenshot shows another Jupyter notebook page, also displaying a large table of data. The table has many columns and rows, similar to the one in the first screenshot. Below the table, there is some text and code, but the table itself is the primary focus.

This screenshot shows a Jupyter notebook page with several different visualizations. At the top, there is a line chart with a blue line and a grid. Below it, there is another line chart with a blue line and a grid. Further down, there is a bar chart with blue bars. The page also contains some text and code blocks interspersed with the charts.

- ▶ Un vrai plat de spaghettis
 - ▶ Pas de vision d'ensemble
 - ▶ Interaction entre plusieurs langages = danger

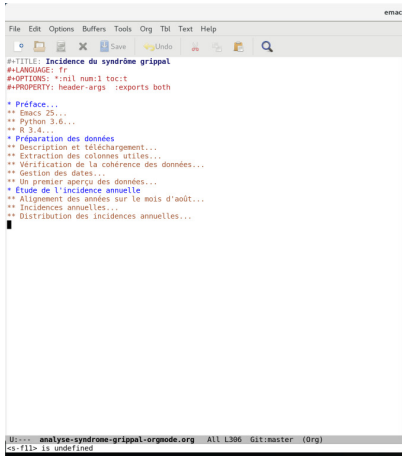
Des codes complexes...

The collage consists of several panels:

- Estimating Color Names by Web Image Searches:** A text-based analysis of search results.
- Text processing:** Snippets of code for handling text data.
- Clustering:** A scatter plot titled "Character distribution of training data" showing data points in various colors. Below it is a "Character's colors and chromatic order identity" visualization, a colorful geometric shape.
- Network analysis:** A "Network graph" showing connections between nodes.
- Ranking:** A "Ranking by results" visualization showing a distribution of values.
- Clustering (repeated):** Another instance of the colorful geometric shape visualization.
- Ranking (repeated):** Another instance of the ranking distribution visualization.
- Clustering (repeated):** A third instance of the colorful geometric shape visualization.
- Ranking (repeated):** A fourth instance of the ranking distribution visualization.
- Clustering (repeated):** A fifth instance of the colorful geometric shape visualization.
- Ranking (repeated):** A sixth instance of the ranking distribution visualization.
- Clustering (repeated):** A seventh instance of the colorful geometric shape visualization.
- Ranking (repeated):** An eighth instance of the ranking distribution visualization.
- Clustering (repeated):** A ninth instance of the colorful geometric shape visualization.
- Ranking (repeated):** A tenth instance of the ranking distribution visualization.
- Clustering (repeated):** An eleventh instance of the colorful geometric shape visualization.
- Ranking (repeated):** A twelfth instance of the ranking distribution visualization.
- Clustering (repeated):** A thirteenth instance of the colorful geometric shape visualization.
- Ranking (repeated):** A fourteenth instance of the ranking distribution visualization.
- Clustering (repeated):** A fifteenth instance of the colorful geometric shape visualization.
- Ranking (repeated):** A sixteenth instance of the ranking distribution visualization.
- Clustering (repeated):** A seventeenth instance of the colorful geometric shape visualization.
- Ranking (repeated):** An eighteenth instance of the ranking distribution visualization.
- Clustering (repeated):** A nineteenth instance of the colorful geometric shape visualization.
- Ranking (repeated):** A twentieth instance of the ranking distribution visualization.
- Clustering (repeated):** A twenty-first instance of the colorful geometric shape visualization.
- Ranking (repeated):** A twenty-second instance of the ranking distribution visualization.
- Clustering (repeated):** A twenty-third instance of the colorful geometric shape visualization.
- Ranking (repeated):** A twenty-fourth instance of the ranking distribution visualization.
- Clustering (repeated):** A twenty-fifth instance of the colorful geometric shape visualization.
- Ranking (repeated):** A twenty-sixth instance of the ranking distribution visualization.
- Clustering (repeated):** A twenty-seventh instance of the colorful geometric shape visualization.
- Ranking (repeated):** A twenty-eighth instance of the ranking distribution visualization.
- Clustering (repeated):** A twenty-ninth instance of the colorful geometric shape visualization.
- Ranking (repeated):** A thirtieth instance of the ranking distribution visualization.

- ▶ Un vrai plat de spaghettis
 - ▶ Pas de vision d'ensemble
 - ▶ Interaction entre plusieurs langages = danger

Des codes complexes...



```
emac
File Edit Options Buffers Tools Org Tbl Text Help
Save Undo
#*=TITLE: Incidence du syndrome grippal
#*=LANGUAGE: fr
#*=OPTIONS: *nil num:1 toc:t
#*=PROPERTY: header-args :exports both

* Préface...
** Emacs 25...
** Python 3.6...
** R 3.4...
* Préparation des données
** Description et téléchargement...
** Extraction des colonnes utiles...
** Vérification de la cohérence des données...
** Gestion des dates...
** Un premier aperçu des données...
* Étude de l'incidence annuelle
** Alignement des années sur le mois d'août...
** Incidences annuelles...
** Distribution des incidences annuelles...

U:--- analyse-syndrome-grippal-orgmode.org All L306 Git:master (Org)
<*-file> is undefined
```

- ▶ Un vrai plat de spaghettis
 - ▶ Pas de vision d'ensemble
 - ▶ Interaction entre plusieurs langages = danger

Des codes complexes...

```
emac
File Edit Options Buffers Tools Org Tbl Text Help
Save Undo
#*TITLE: Incidence du syndrome grippal
#*LANGUAGE: fr
#*OPTIONS: *mil num:1 toc:1
#*PROPERTY: header-args :exports both


* Préface...
** Emacs 25...
** Python 3.6...
** R 3.4...

* Préparation des données
** Description et téléchargement...
** Extraction des colonnes utiles...
** Vérification de la cohérence des données...
** Gestion des dates...
** Un premier aperçu des données

Nous passons au langage R pour avoir un résumé statistique de notre jeu de données.
#*BEGIN_SRC R :results output :var data=date-inc-sorted
data$date <- as.Date(data$date)
summary(data)
#*END_SRC

#*RESULTS:
:      date      inc
: Min. :1984-12-31 Min. : 0
: 1st Qu.:1993-02-25 1st Qu.: 5164
: Median :2001-04-16 Median : 16188
: Mean   :2001-04-15 Mean   : 63053
: 3rd Qu.:2009-06-04 3rd Qu.: 49576
: Max.   :2017-07-24 Max.   :1801824

Regardons enfin à quoi ressemblent nos données !
#*BEGIN_SRC R :results output graphics :file inc-plot.png :var data=date-inc-sorted
data$date <- as.Date(data$date)
plot(data, type="l", xlab="Date", ylab="Incidence hebdomadaire")
#*END_SRC

#*RESULTS:

U:--- analyse-syndrome-grippal-orgmode.org Top L1 Git:master (Org)
Beginning of buffer
```

- ▶ Un vrai plat de spaghettis
 - ▶ Pas de vision d'ensemble
 - ▶ Interaction entre plusieurs langages = danger

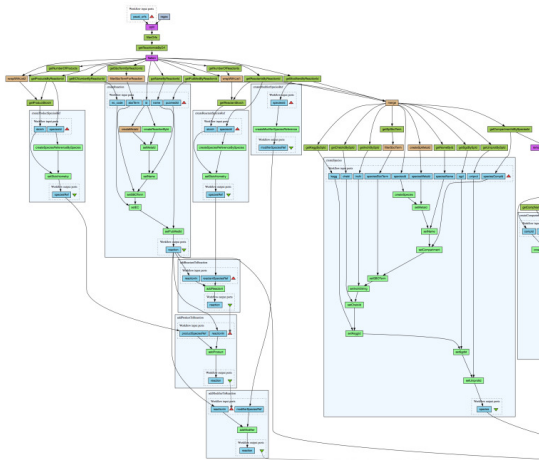
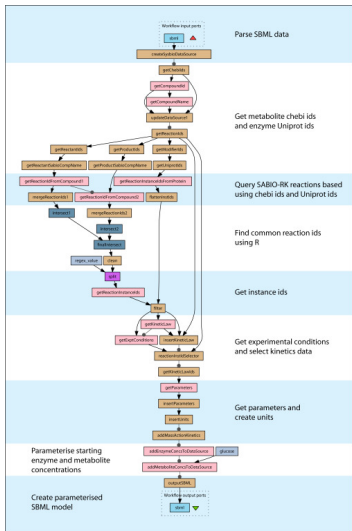
Des codes complexes...

The collage consists of several panels:

- Top Left:** A dense block of text, likely a code snippet or log output, with some lines highlighted in red.
- Top Middle:** A table with multiple columns and rows, possibly a configuration or data table.
- Top Right:** A table with columns labeled 'Date', 'Time', 'Status', 'Action', 'User', and 'IP', containing various entries.
- Middle Left:** A code snippet with several lines of text, some highlighted in red.
- Middle Middle:** A table with columns labeled 'ID', 'Name', 'Age', 'Sex', 'Height', 'Weight', 'Blood Pressure', and 'Heart Rate', containing numerical and categorical data.
- Middle Right:** A code snippet with several lines of text, some highlighted in red.
- Bottom Left:** A table with columns labeled 'ID', 'Name', 'Age', 'Sex', 'Height', 'Weight', 'Blood Pressure', and 'Heart Rate', containing numerical and categorical data.
- Bottom Middle:** A code snippet with several lines of text, some highlighted in red.
- Bottom Right:** A table with columns labeled 'ID', 'Name', 'Age', 'Sex', 'Height', 'Weight', 'Blood Pressure', and 'Heart Rate', containing numerical and categorical data.

- ▶ Un vrai plat de spaghettis
 - ▶ Pas de vision d'ensemble
 - ▶ Interaction entre plusieurs langages = danger

... et difficiles à orchestrer



... et difficiles à orchestrer

Le **Workflow** :

- ▶ Vue de haut niveau plus claire
- ▶ Compositions de codes et mouvements de données explicites
- ▶ Partage, réutilisation, et exécution plus sûre
- ▶ Le notebook en est une forme à la fois appauvrie et plus riche
- ▶ Pas de façon simple/mature de passer d'un notebook à un workflow

Exemples :

- ▶ Galaxy, Kepler, Taverna, Pegasus, Collective Knowledge, VisTrails
- ▶ Légers : dask, drake, swift, snakemake...
- ▶ Hybrides : SOS-notebook...

L'usine à gaz des calculs coûteux

Les **calculs interminables** et les **gros volumes de données**

- ▶ JupyterHub et les supercalculateurs : en développement
- ▶ Checkpoint et Cache
- ▶ Le workflow permet de passer à l'échelle

Des éco-systèmes complexes

Que se cache-t-il derrière ce simple

1

```
import matplotlib
```

Package: python3-matplotlib

Version: 2.1.1-2

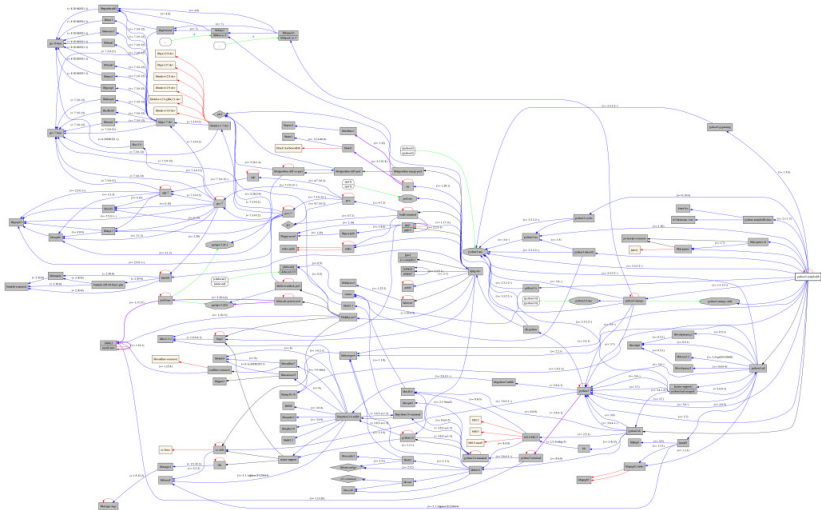
Depends: python3-dateutil, python-matplotlib-data (>= 2.1.1-2),
python3-pyparsing (>= 1.5.6), python3-six (>= 1.10), python3-tz,
libjs-jquery, libjs-jquery-ui, python3-numpy (>= 1:1.13.1),
python3-numpy-abi9, python3 (<< 3.7), python3 (>= 3.6~),
python3-cycler (>= 0.10.0), python3:any (>= 3.3.2-2~), libc6 (>=
2.14), libfreetype6 (>= 2.2.1), libgcc1 (>= 1:3.0), libpng16-16 (>=
1.6.2-1), libstdc++6 (>= 5.2), zlib1g (>= 1:1.1.4)

Des éco-systèmes complexes

Que se cache-t-il derrière ce simple

1

```
import matplotlib
```



Des éco-systèmes complexes

Pas de standard :

- ▶ Linux (apt, rpm, yum), MacOS X (brew, McPorts, Fink), Windows (?)
- ▶ Ni pour l'installation ni pour récupérer les informations... 😞

```
1 import sys
2 print(sys.version)
3 import matplotlib
4 print(matplotlib.__version__)
5 import pandas as pd
6 print(pd.__version__)
```

```
3.6.3 (default, Oct 3 2017, 21:16:13)
[GCC 7.2.0]
2.1.1
0.20.3
```

```
1 library(ggplot2)
2 sessionInfo()
```

```
R version 3.4.3 (2017-11-30)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Debian GNU/Linux buster/sid
```

Matrix products: default

```
BLAS: /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.7.1
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.7.1
```

locale:

```
[1] LC_CTYPE=fr_FR.UTF-8      LC_NUMERIC=C
[3] LC_TIME=fr_FR.UTF-8      LC_COLLATE=fr_FR.UTF-8
[5] LC_MONETARY=fr_FR.UTF-8  LC_MESSAGES=fr_FR.UTF-8
```

other attached packages:

```
[1] ggplot2_2.2.1
```

loaded via a namespace (and not attached):

```
[1] colorspace_1.3-2 scales_0.5.0 compiler_3.4.3 laz
[5] plyr_1.8.4 pillar_1.1.0 gtable_0.2.0 tib
[9] Rcpp_0.12.15 grid_3.4.3 rlang_0.1.6 mun
```

Contrôler son environnement

Un environnement contrôlé :

- ▶ Travailler dans une machine virtuelle (lourd) ou un conteneur docker (léger)

Conserver le bazar

- ▶ Capture automatique de l'environnement
- ▶ CDE, ReProZip, CARE

Faire le ménage

- ▶ Partir d'un environnement vierge
- ▶ Installer uniquement le nécessaire (et l'explicitier)
- ▶ Docker, Singularity, Guix, Nix...

L'épreuve du temps



aleksandrmarkin@geg

Compatibilité ascendante

- ▶ Python et tout son écosystème à évolution hyper rapide

```
1 python2 -c "print(10/3)"  
2 python3 -c "print(10/3)"
```

```
3  
3.3333333333333335
```

Compatibilité ascendante

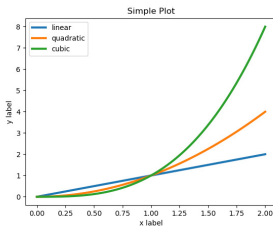
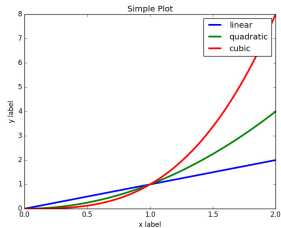
► Python et tout son écosystème à évolution hyper rapide

```
1 python2 -c "print(10/3)"
```

```
2 python3 -c "print(10/3)"
```

```
3
```

```
3.3333333333333335
```

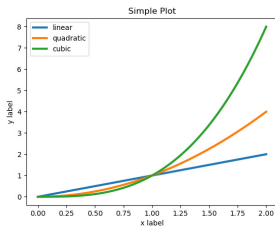
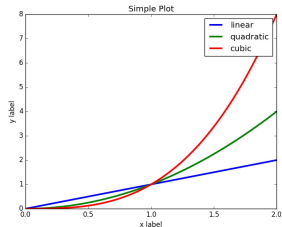


Compatibilité ascendante

- ▶ Python et tout son écosystème à évolution hyper rapide

```
1 python2 -c "print(10/3)"  
2 python3 -c "print(10/3)"
```

```
3  
3.3333333333333335
```



- ▶ Cortical Thickness Measurements (PLOS ONE, June 2012) :
FreeSurfer : differences were found between the Mac and HP workstations and between Mac OSX 10.5 and OSX 10.6.
- ▶ Incompatibilité de formats entre orgmode 7.*, 8.*, 9.*, etc.

Les outils à développement rapide

Rapides, mais fragiles et peu pérennes :

- ▶ Correction ou introduction de bug
- ▶ Il devient nécessaire de vérifier régulièrement la restructurabilité et la fonctionnalité de ces environnements (intégration continue et tests de non régression)



Popper : <http://falsifiable.us/>

Autre possibilité :

- ▶ Se restreindre à ce qui est maîtrisable (C par exemple)

L'archivage

Gestion du code source

- ▶ Github/Gitlab/... : stables, ouverts, ... pérennes ?
 - ▶ ~~Google Code, Gitorious, Code Spaces~~



Software Heritage

HAL
archives-ouvertes.fr

Gestion des environnements

- ▶ Pérérité de l'accès à dockerhub, nix repository, code ocean... ?
- ▶ Une fois l'environnement gelé, quelle est la pérennité d'une VM, d'une image docker... ?

Conserver le plus d'informations possible en automatisant

- ▶ Logiciels, versions, procédure d'installation

Où en sommes nous?

M4-S0 : Vers une étude reproductible : la réalité du terrain

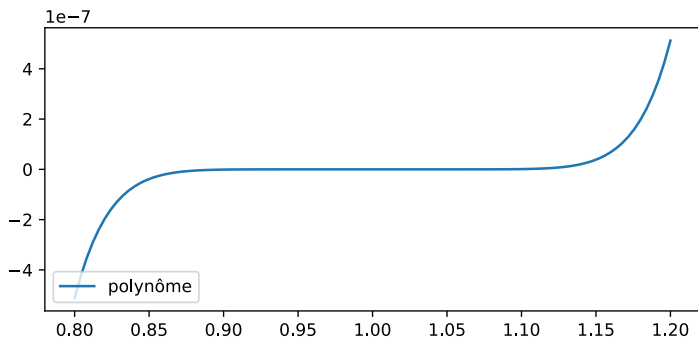
M4-S1 : L'enfer des données

M4-S2 : L'enfer du logiciel

M4-S3 : L'enfer du calcul

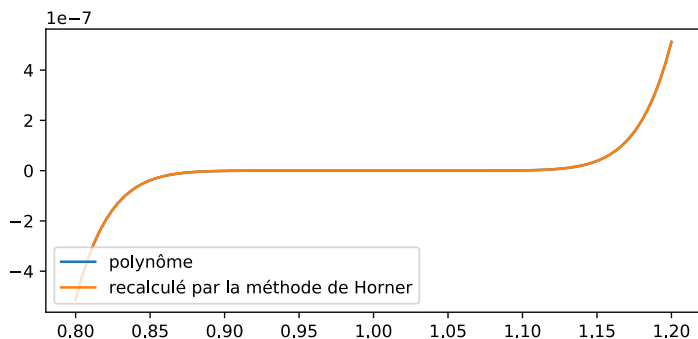
M4-S4 : Conclusion

L'arithmétique à virgule flottante



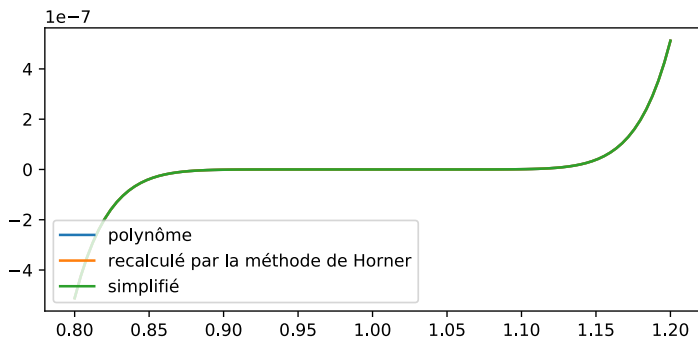
```
1 def polynome(x):  
2     return x**9 - 9.*x**8 + 36.*x**7 - 84.*x**6 + 126.*x**5 \  
3         - 126.*x**4 + 84.*x**3 - 36.*x**2 + 9.*x - 1.
```

L'arithmétique à virgule flottante



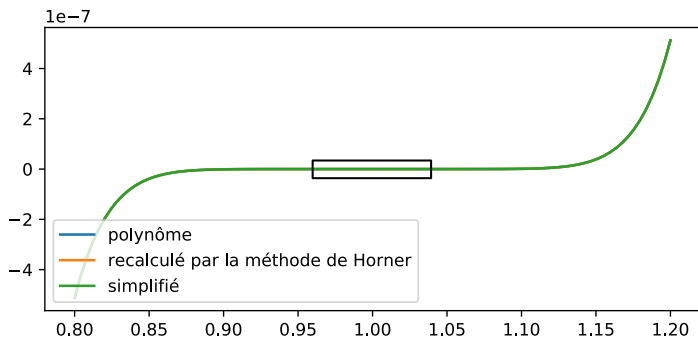
```
1 def horner(x):  
2     return x*(x*(x*(x*(x*(x*(x - 9.) + 36.) - 84.) + 126.) \  
3         - 126.) + 84.) - 36.) + 9.) - 1.
```

L'arithmétique à virgule flottante

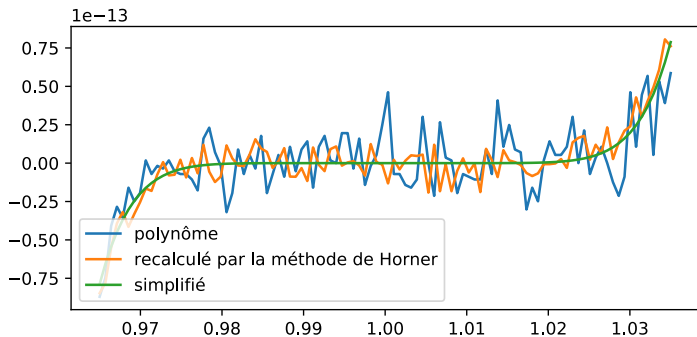


```
1 def simple(x):  
2     return (x-1.)**9  
3     # trop facile!
```

L'arithmétique à virgule flottante



L'arithmétique à virgule flottante



L'arrondi

- ▶ Il y a un arrondi implicite dans chaque opération.
- ▶ $a+b$ est en réalité $\text{arrondi}(a+b)$.
- ▶ Mais :
$$\text{arrondi}(\text{arrondi}(a+b)+c) \neq \text{arrondi}(a+\text{arrondi}(b+c)).$$
- ▶ L'ordre des opérations est donc important.

Pour un calcul reproductible, il faut préserver l'ordre des opérations!!!

Comment l'expliquer à mon compilateur?

Pour accélérer le calcul, les compilateurs peuvent changer l'ordre des opérations, et donc le résultat.

Pour un calcul reproductible, il y a deux approches :

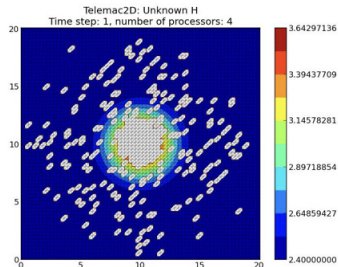
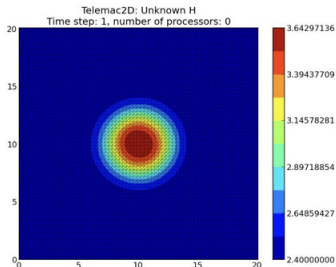
- ▶ Insister sur le respect de l'ordre des opérations,
 - ▶ si le langage le permet.
 - ▶ Exemple : Le module 'ieee_arithmetic' en Fortran 2003
- ▶ Rendre la compilation reproductible :
 - ▶ Noter la version précise du compilateur
 - ▶ Noter toutes les options de compilation

Calcul parallèle

- ▶ Objectif : une exécution plus rapide → Minimiser la communication entre processeurs → Adapter la répartition des données... → et donc l'ordre des opérations
- ▶ Conséquence : le résultat dépend du nombre de processeurs !

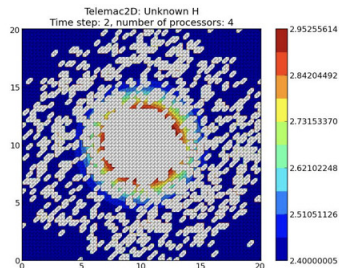
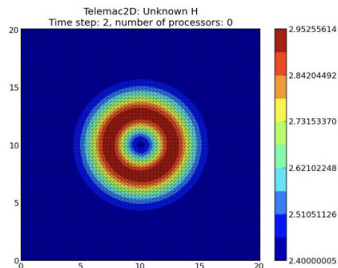
Minimiser cet impact du parallélisme reste un sujet de recherche.

Calcul parallèle : exemple



Source : Thèse de Rafife Nheili, Université de Perpignan, 2016

Calcul parallèle : exemple



Source : Thèse de Rafife Nheili, Université de Perpignan, 2016

Les plateformes de calcul

- ▶ Plateforme de calcul : matériel + infrastructure logicielle
- ▶ Calcul = plateforme + logiciel + données
- ▶ La plateforme définit l'interprétation du logiciel.
- ▶ Plateforme et logiciel définissent l'interprétation des données.
- ▶ Autres aspects définis par la plateforme :
 - ▶ la représentation des entiers (16/32/64 bits)
 - ▶ la gestion des erreurs

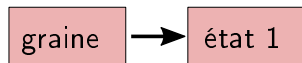
Les nombres aléatoires

- ▶ Utilisés pour simuler les processus stochastiques.
- ▶ En pratique : nombres **pseudo-** aléatoires.
- ▶ Suite de nombres **en apparence** aléatoires. . .
- ▶ . . . mais générés par un algorithme déterministe.

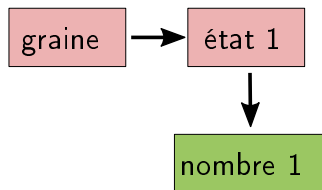
Générateur de nombres pseudo-aléatoires

graine

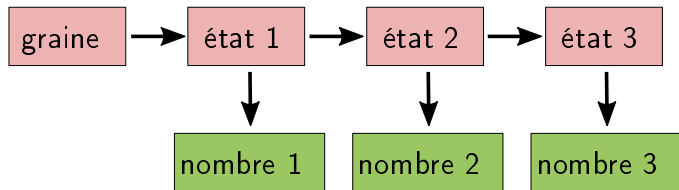
Générateur de nombres pseudo-aléatoires



Générateur de nombres pseudo-aléatoires



Générateur de nombres pseudo-aléatoires



La reproductibilité en théorie

- ▶ Principe : même graine + même algorithme \rightarrow même suite
- ▶ La graine est souvent choisie comme fonction de l'heure
- ▶ Il faut la définir dans le code d'application

La reproductibilité en pratique

- ▶ Même graine + même algorithme \rightarrow même suite : pas évident en arithmétique à virgule flottante !
- ▶ Une astuce simple pour permettre la vérification : tester quelques valeurs du début de la suite

Exemple : le langage Python

```
1 import random
2
3 random.seed(123)
4 for i in range(5):
5     print(random.random())
```

```
0.0523635988509
0.0871866775226
0.40724176367
0.107700234938
0.901198877952
```

Exemple : le langage Python

```
1 import random
2
3 random.seed(123)
4 assert random.random() == 0.052363598850944326
5 assert random.random() == 0.08718667752263232
6 assert random.random() == 0.4072417636703983
```

Les points clés à retenir

- ▶ Les résultats d'un calcul dépendent
 - ▶ du logiciel
 - ▶ des données d'entrée
 - ▶ de la plateforme de calcul : matériel, compilateurs...
- ▶ L'influence de la plateforme est importante pour l'arithmétique à virgule flottante.
- ▶ Notez les paramètres dont peuvent dépendre vos résultats :
 - ▶ version du compilateur, options de compilation
 - ▶ matériel (type de processeur, GPU...)
 - ▶ nombre de processeurs
- ▶ En utilisant un générateur de nombres aléatoires, définissez la graine et vérifiez les premiers éléments de la suite.

Où en sommes nous?

M4-S0 : Vers une étude reproductible : la réalité du terrain

M4-S1 : L'enfer des données

M4-S2 : L'enfer du logiciel

M4-S3 : L'enfer du calcul

M4-S4 : Conclusion

Ce qu'il faut retenir de ce MOOC

Un véritable enjeu

- ▶ Méthodologie scientifique
- ▶ Inspectabilité et réutilisation

Des outils existent

- ▶ Documents computationnels et Workflows, Suivi de version et Archives, Environnements logiciels, Intégration continue. . .
- ▶ Ces outils évoluent en permanence
 - ▶ Choisissez ceux qui sont les plus adaptés à votre contexte
 - ▶ Cherchez un compromis entre modernisme et pérenité

Mettez en pratique, ne vous découragez pas !

- ▶ Prenez des notes rigoureusement
- ▶ Rendez l'information exploitable et accessible
- ▶ Améliorez petit à petit