Data AI basics

Using a computer as a computer scientist

Louis Jachiet
louis.jachiet@telecom-paris.fr
09 September 2025

Digital life

Mails – some general and technical considerations

1.1.2 What is a mail?

From a technical standpoint a mail is just a text file looking like this :

```
From: Louis Jachiet <test@jachiet.com>
To: Louis Jachiet <louis.jachiet@telecom-paris.fr>
Cc: cc-ed@jachiet.com
Subject: Re: This is a subject
Message-ID: <20220903164909.uksb6uca3o2zyyd4@moa.enst.fr>
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Disposition: inline
Content-Transfer-Encoding: quoted-printable
In-Reply-To: <20220903164842.fscuyipdyhc67pxk@moa.enst.fr>

Well received!
--
Louis Jachiet
```

1.1.3 What is a mail?

To send a mail, you contact a (mail) server that will take this file and send it to another server, that will send it to another server until it reaches the inbox of the recipients.

1.1.4 The good

Mail is *somewhat* standard : sending text and files to almost anyone

Mail is federated : it does not belong to a single entity

Mail is asynchronous : generally instantaneous (but not always!)

1.1.5 The bad

Most extensions are non standard : encryption, receipt, colorful mails, etc.

Mail is federated : each provider runs its own infrastructure which can create friction

Mail is asynchronous : mail is not always fast...

1.1.6 The ugly

Big mail providers have created a oligopoly : small mails providers often end in spam

Companies are almost spammers : from a communication tool it became the de facto marketing tool

Spam and phishing are rampant : a lot of malicious actors are polluting mails and trying to scam you

1.1.7 How not to write an email

From : duck42@spacelasers.com
To : Prof 3
Cc : Prof 2, Random person
Bcc : Prof 1
Subject : Mail

I had trouble with computers last week first I tried to reinstall my computer and it failed then I decided to use my friend computer but I had to start over and I was no able to send my work yesterday. Here is my file
Attachment : tp.rar

1.1.8 Use a proper subject!

Keep it relatively short, but informative.

Don’t hesitate to modify the subject when replying.

1.1.9 Be polite!

Unless you are replying to a recent mail :

- Add a **salutation** (Hello, Hi, Dear Teacher, Dear Ms X, etc.)
- Add an **appropriate** closing (Sincerely, Regards, Best Regards, Thank you)
- Use the correct **language level**
- Never write an email when you are angry / **always down tone** your angriness (write the email, wait a few days)
- **Don't** use sarcasm

1.1.10 Be specific!

Teachers receives dozens if not hundreds of mails a day : they spend 30s reading them sometimes doing something else in parallel (**caution** : made up statistics).

Each mail should be very specific in what you want!

1.1.11 Give context!

We often assume symmetric relation and that readers will have the same mental context as ours. *It is not the case!*

It is always better to add slightly too much context!

1.1.12 Too much information

Focus on what is *essential*.

Some advice :

- The shape of your mail helps reading it!
- people (generally) don't need to know why you were late, sick, etc.
- the body of a mail can often fit the 280 chars tweet length

1.1.13 Timely follow up

People (teachers & students) loose tracks of mails, it is Ok.

You can send a reminder but :

- we have other things to do
- we might take week-ends and vacation
- the urgency for you is not a criterion if you are the reason for that urgency

---

Mails should ideally be sent during a week day before 5 pm.

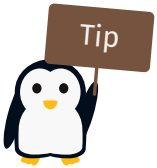
***Non urgent** : wait at least **one week**, **urgent** : go their office*

1.1.14 Proof-read your mail

- Re-read everything you wrote
- Check all items above
- Check if attachments are needed

1.1.14 Proof-read your mail

- Re-read everything you wrote
- Check all items above
- Check if attachments are needed



To write a mail :

- Start by putting the attachment,
- then, write the mail
- then add recipients.

Digital life

How to use mail for Data AI

1.2.2 How to write an email

- Use a proper mail address
- Select carefully the recipients
- Use a proper subject
- Be polite
- Be brief but describe : the problem, your attempt, an eventual (short but VALID) excuse
- For attachments :
  - use meaningful names
  - prefer standardized file types (e.g. PDF, ZIP) not PAGES
  - remove useless files
  - keep size small (if possible)

1.2.3 How to write an email

From : first.lastname@telecom-paris.fr
To : Main teacher
Cc : TAs
Subject : [DATAAI 101] Late submission for assignment XY
Dear teacher,

I am firstname LASTNAME and I am in your course DATAAI 101. We were supposed to submit the TP 4 on moodle before the 5th but sadly I missed the deadline and now the submission on moodle is closed.

I am really sorry for this but would you consider the attached file ? Or allow me to submit on moodle ?

Warm Regards,
–
Firstname Lastname
Attachment : LASTNAME_firstname_TP4_DATAAI_101.zip

1.2.4 Using the Télécom mail

Your `@telecom-paris.fr` address is not forever!

It will disappear a few months after your departure, be wary when communicating with this address!

However, lots of important communications will be sent to this address, check it regularly (or make a redirection)!

It comes with a calendar tool, don't hesitate to use such a tool!

Digital life

Security

1.3.2 Threat model

For each data storage determine :

- **Who** has access to it
- Against **what** I want to protect myself ?
- **How** do I protect myself against it ?

1.3.3 Résumé

Confidentiality : Very low

Could be put on my webpage ?

1.3.3 Résumé

Confidentiality : Very low

Could be put on my webpage ?

- If it includes your mail, what about **spam** ?

1.3.3 Résumé

Confidentiality : Very low

Could be put on my webpage ?

- If it includes your mail, what about **spam** ?
- If if includes a lot of personal infos how to protect “**password lost**” questions and **identity theft** ?

1.3.4 Bank Account Number / IDs

Confidentiality : Low

Who can use it ?

1.3.4 Bank Account Number / IDs

Confidentiality : Low

Who can use it ?

- Scammers.

1.3.5 Personal mail

Confidentiality : High

What are the risks ?

1.3.5 Personal mail

Confidentiality : High

What are the risks ?

- Scammers

1.3.5 Personal mail

Confidentiality : High

What are the risks ?

- Scammers
- Extortion

1.3.5 Personal mail

Confidentiality : High

What are the risks ?

- Scammers
- Extortion
- Intimate mails being leaked

1.3.5 Personal mail

Confidentiality : High

What are the risks ?

- Scammers
- Extortion
- Intimate mails being leaked
- Stolen online accounts

1.3.5 Personal mail

Confidentiality : High

What are the risks ?

- Scammers
- Extortion
- Intimate mails being leaked
- Stolen online accounts
- ...

1.3.6 How to secure your data?

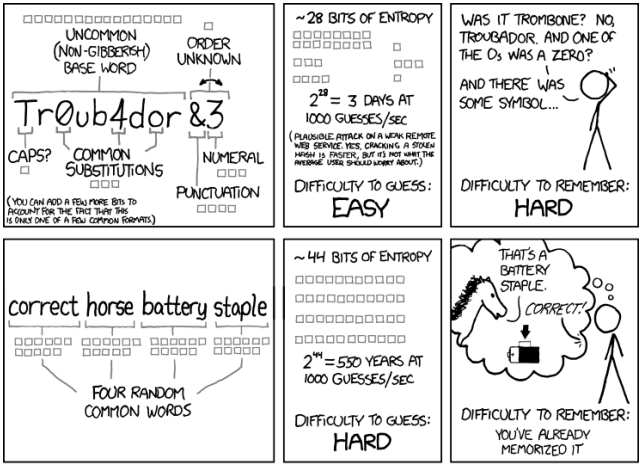
Passwords

- A different password for each website : **use a password manager!**
- Strong passwords (you are CS students, compute the entropy!)

Storage devices

- Encrypt all the drives containing data
- Don't use random USB keys

1.3.7 Crafting passwords?



THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

1.3.8 Crafting passwords

Algorithm :

- Select 3 to 4 random words
- Separate them with random numbers or punctuation

Example :

- mechanization / preference / outback / apologizing
- Separate them with numbers or punctuation

Result : mechanization4preference;outback_apologizing
Entropy 30 000 words → $30\,000^4 \approx 10^{18}$ possibilities

Digital life

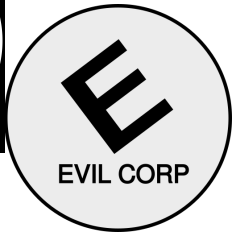
Backups

1.4.2 Who is the biggest enemy of your data?

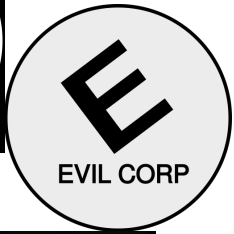
1.4.2 Who is the biggest enemy of your data?

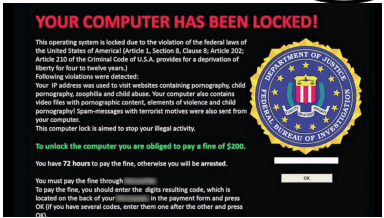
1.4.2 Who is the biggest enemy of your data?

1.4.2 Who is the biggest enemy of your data?



1.4.2 Who is the biggest enemy of your data?





1.4.2 Who is the biggest enemy of your data?

1.4.3 What can happen to your data?

- **Loosing** it
- Being **stolen**
- Have an **accident** (fire, flood, etc.)
- Hard disk **failure**
- Data **corruption**
- Accidental **deletion**
- ...

One of these items happens every few years to everyone...

1.4.4 How to backup data

Golden rule : 3 times

- three different storage types,
- three different places,
- three types of backups.

Can you recover ?

It is important to check that EVERYTHING is really backed-up from time to time and that the data is really there :

- Closed backup provider
- Corrupted USB key
- Some crucial files are missing
- etc.

Mental exercise : you loose your laptop/google/usb key/etc. what do you really loose ?

Digital life

Online presence

1.5.2 Web

What is online will stay online...

but you can **somewhat** control it by adding content.

1.5.3 Creating a web page

Creating a webpage can be done in a **free** and **easy** way, for instance :

- Github pages
- Gitlab
- Wordpress
- Netlify
- and many other possibilities...

Tools for computer scientists

Bibliography

2.1.2 Bibliography

Producing science requires to be **up to date** on recent advances.

2.1.2 Bibliography

Producing science requires to be **up to date** on recent advances.

In computer science, science is first published in **research papers** (theses, journals, conferences, etc.).

2.1.2 Bibliography

Producing science requires to be **up to date** on recent advances.

In computer science, science is first published in **research papers** (theses, journals, conferences, etc.).

To pursue science you need to **find**, **get access**, **read**, **remember** and **cite** the relevant article for your research.

2.1.3 Finding articles

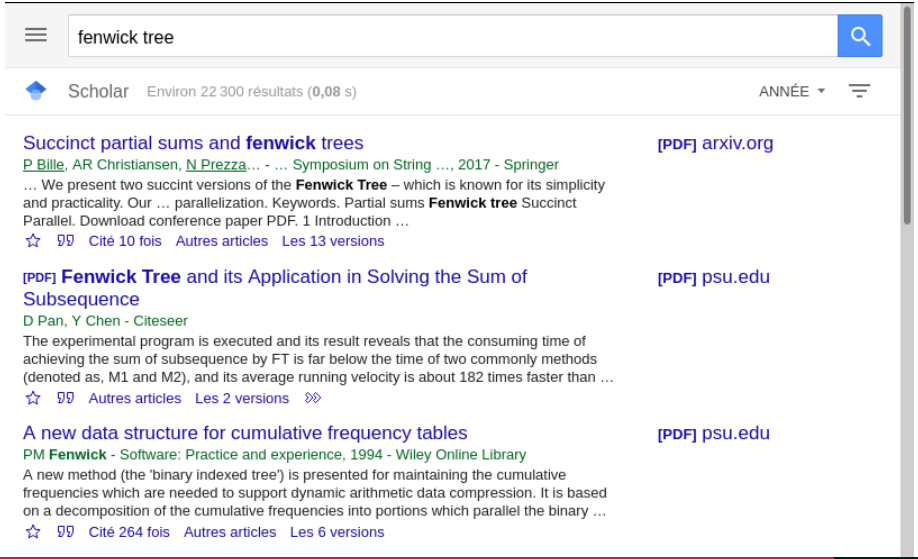
Research articles are generally not easy to read. When starting a completely new subject, it is a good idea to read **blog posts** or **books** about it rather.

When starting an internship or a PhD most of the bibliography will be provided by your advisor.

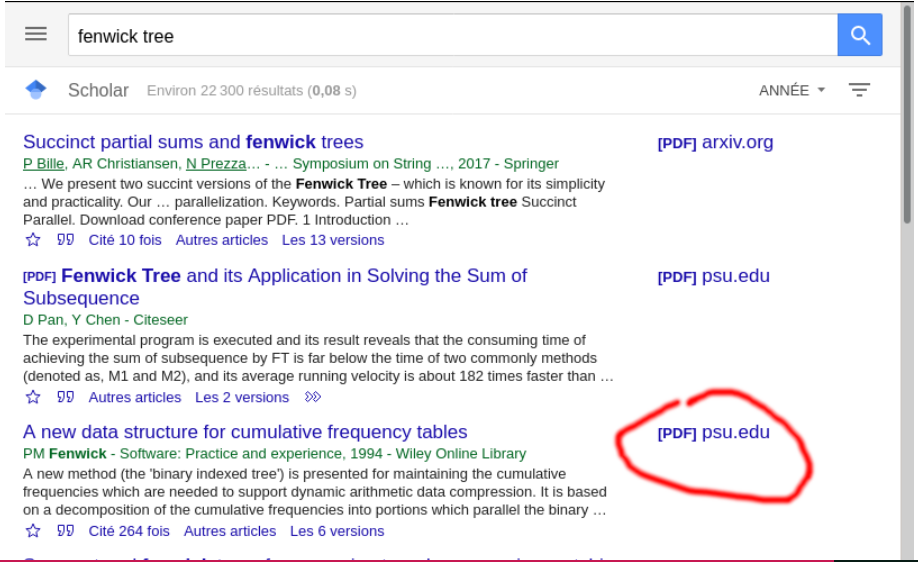
When looking for a specific reference or with a specific question you can use a search engine, the most popular being **Google Scholar**.

2.1.4 Finding article with Google scholar

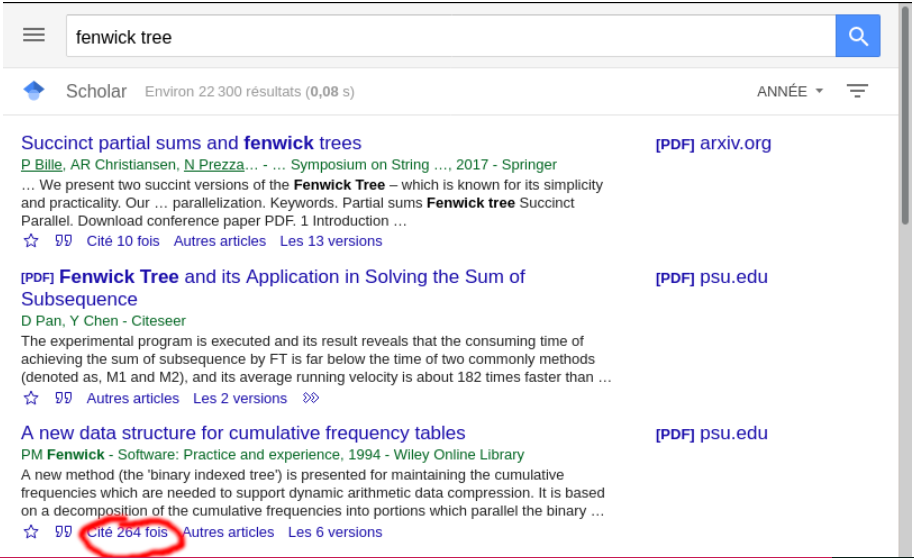
2.1.5 Finding article with Google scholar



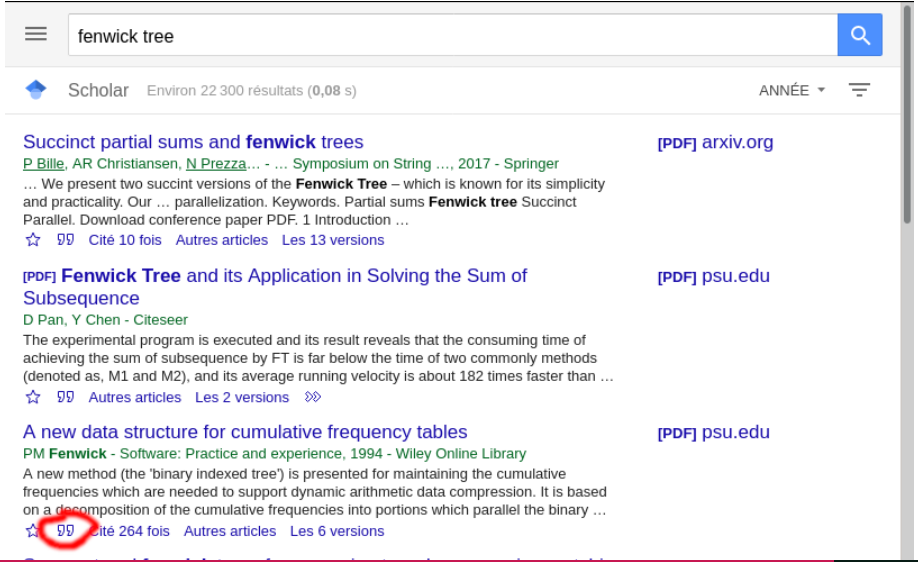
2.1.6 Finding article with Google scholar



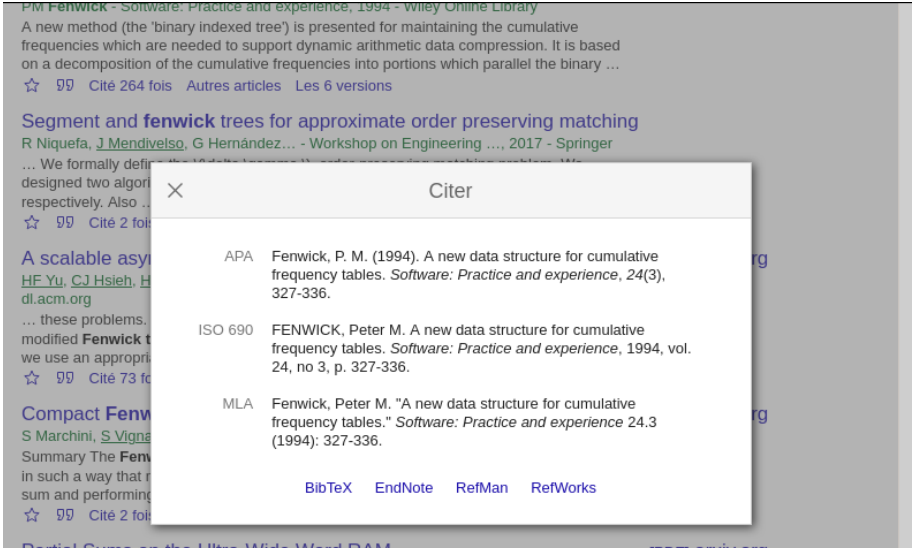
2.1.7 Finding article with Google scholar



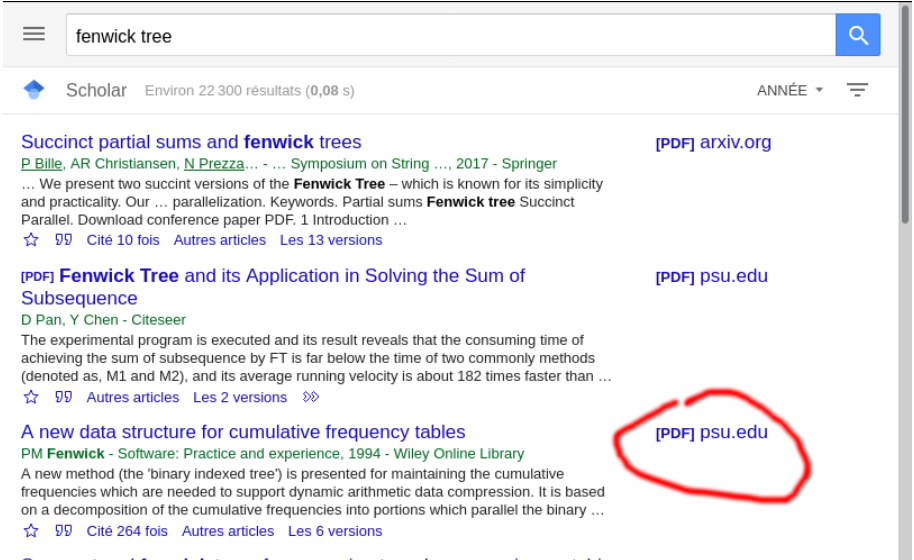
2.1.8 Finding article with Google scholar



2.1.9 Finding article with Google scholar



2.1.10 Getting access to articles



2.1.11 Getting access to articles

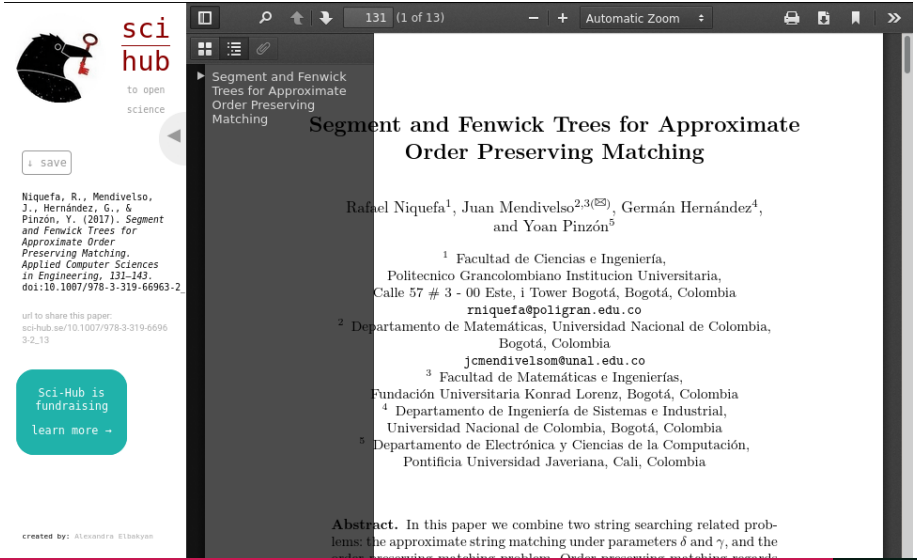
Some articles are behind a **paywall** what can you do :

- See if Google scholar found an opened version
- Access through Télécom (Télécom is subscribed to several publishers)
- Use libgen / sci-hub (legality to discuss)

2.1.12 Getting access to articles with sci-hub

2.1.13 Getting access to articles

2.1.14 Getting access to articles



2.1.15 Read and remember articles

When you have read 3 articles in the past month, it is easy to remember :

- What each article says
- In what version of the article you can find the result
- The precise reference of the article

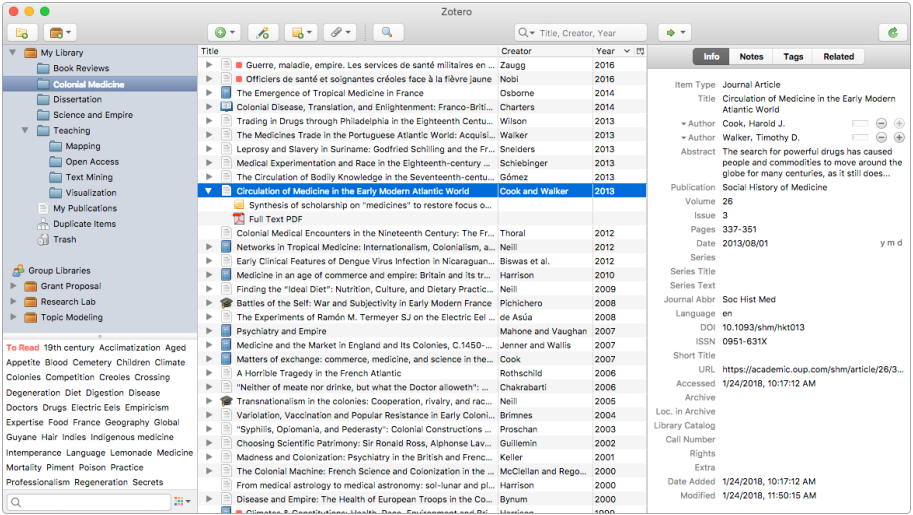
After three years (duration of PhD in France) and 100s of articles, it is much harder...

2.1.16 Read and remember articles

It is a very good thing to have **some system** to keep track of which article you read and what it contained as well as the BibTeX reference.

- A folder with the **PDF** of articles and some **notes**
- A **wiki** or some other **note taking** tool
- A dedicated software, e.g. **Zotero**

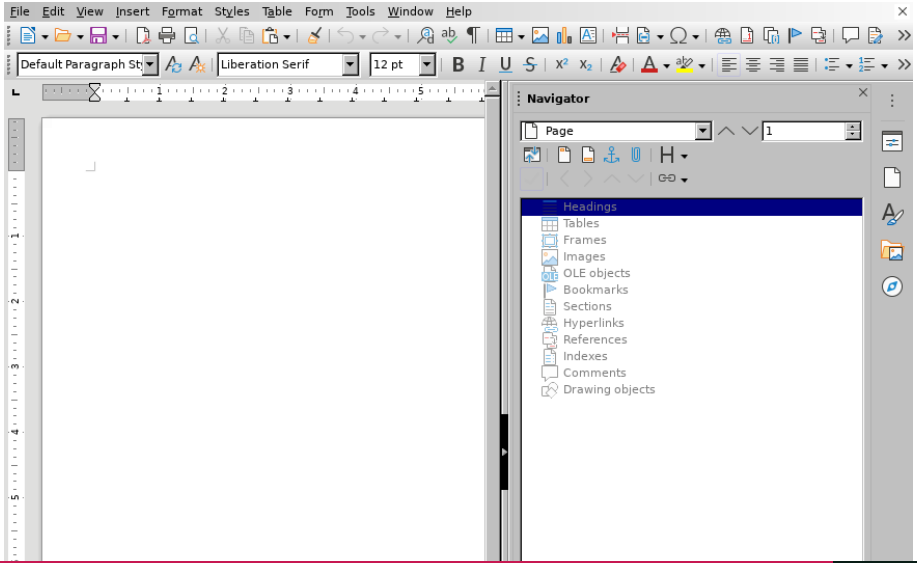
2.1.17 Read and remember articles with Zotero



Tools for computer scientists

Writing research

2.2.2 Office


File Edit View Insert Format Styles Table Form Tools Window Help
Default Paragraph St
Liberation Serif
12 pt
Navigator
Page
1
Headings
Tables
Frames
Images
OLE objects
Bookmarks
Sections
Hyperlinks
References
Indexes
Comments
Drawing objects

2.2.3 LaTeX

In CS, most conferences expect articles to be formatted through LaTeX which is a software layer over TeX.

LaTeXcode for this slide

```
\begin{frame}{\LaTeX}
  In CS, most conferences expect articles to be formatted
  through {\LaTeX} which is a software layer over {\TeX}.

  \hle

  \inputmintedbox{LaTeX}{recslide.tex}
                 {\LaTeX code for this slide}
\end{frame}
```

2.2.4 LaTeX

Pros

- Easy to write science
- Separation of styling and content
- Can be “programmed”
- Works well with git and so
- Good looking documents by default

Cons

- Learning curve is hard for first timers
- Requires help to do something new
- Requires installation of software to write

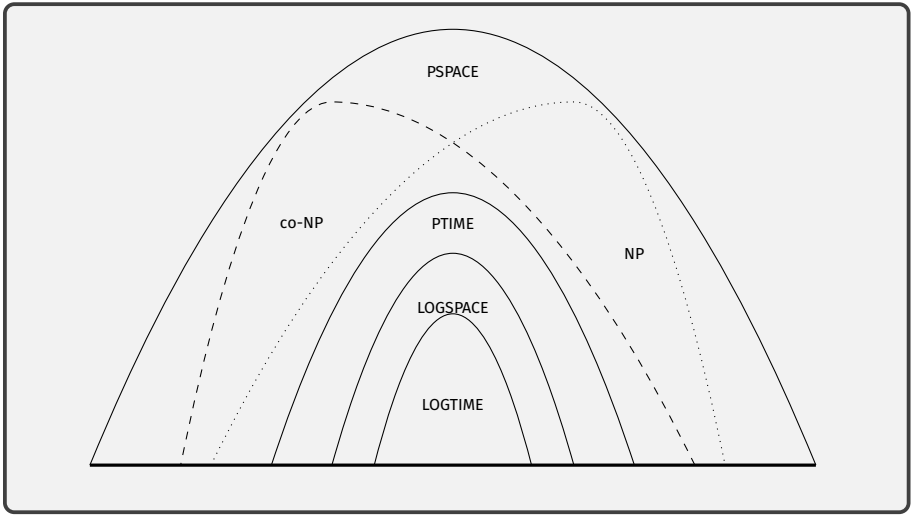
2.2.5 LaTeX

One big advantage of LaTeX is that there exists templates for a multitude of use cases :

- research articles
- presentations
- résumé
- letters
- ...

2.2.6 Drawing figures with Tikz

Research articles often contains complex figures such as :



2.2.6 Drawing figures with Tikz

Research articles often contains complex figures such as :

```
\begin{tikzpicture}
  \tiny \pgftransformscale{.8}
  \draw[very thick] (6,0) -- (-6,0);
  \draw (-1.3,0) parabola bend (0,2.5) (1.3,0) ; \node at (0,1) { LOGTIME };
  \draw (-2,0) parabola bend (0,3.5) (2,0); \node at (0,2.6) { LOGSPACE };
  \draw (-3,0) parabola bend (0,4.5) (3,0); \node at (0,4) {PTIME};
  \draw[dotted] (-4,0) parabola bend (2,6) (4.5,0);  \node at (3,3.5) {NP};
  \draw[dashed] (4,0) parabola bend (-2,6) (-4.5,0); \node at (-2.5,4) {co-NP};
  \draw (-6,0) parabola bend (0,7.2) (6,0); \node at (0,6.5) {PSPACE};
\end{tikzpicture}
\vspace{-2em}
```

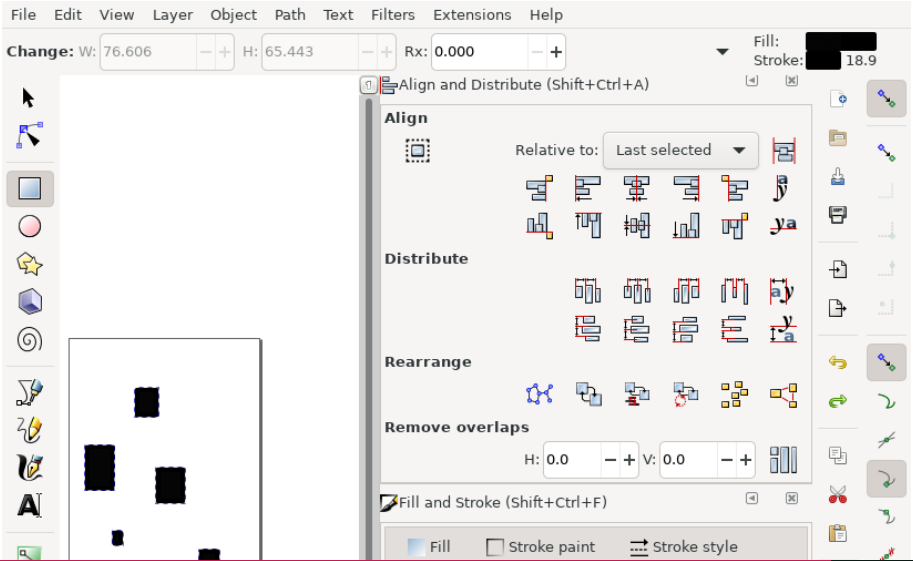
Example of Tikz use from texample.net

2.2.7 Drawing figures with inkscape

For articles, always prefer **vector** graphics to **raster** graphic :

- lighter
- scalable
- editable

2.2.8 Drawing figures with inkscape



Tools for computer scientists

Version control systems

2.3.2 Version Control Systems (VCS)

A system eventually capable of :

- Capture the evolution of files
- Synchronize files across multiple computers
- Allow multiple users to work in parallel on the same documents

2.3.2 Version Control Systems (VCS)

A system eventually capable of :

- Capture the evolution of files
- Synchronize files across multiple computers
- Allow multiple users to work in parallel on the same documents

It can be applied to a wide range of types of files :

- Source code
- Websites
- Reports
- Work environments
- ...

2.3.3 Popular VCS

The most popular ones are :

- **Git**
- Bazaar
- mercurial
- SVN (outdated)
- RCS (very outdated)

2.3.4 Around git

While Git works in a fully decentralized way but whole ecosystems have emerged around it with version control but also :

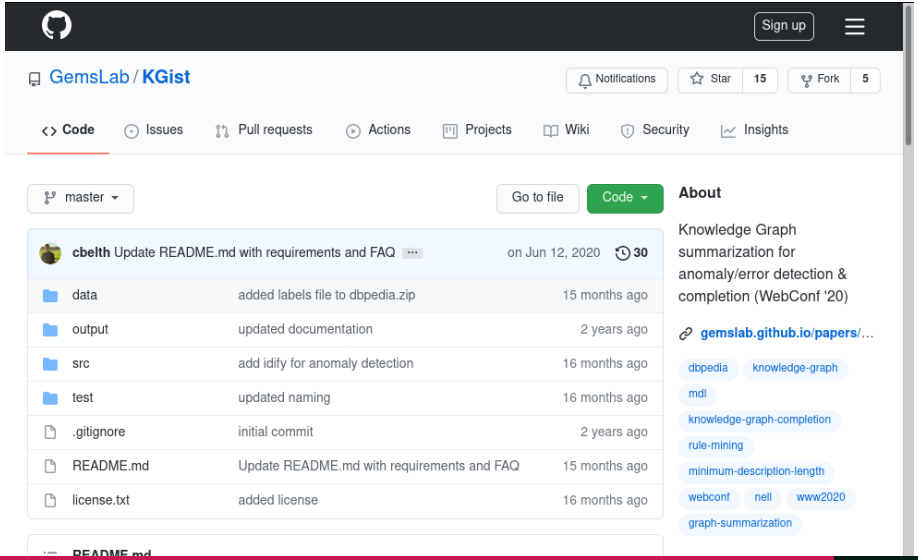
- access management
- file hosting
- wiki
- pull requests
- continuous integration
- ...

2.3.5 Around git

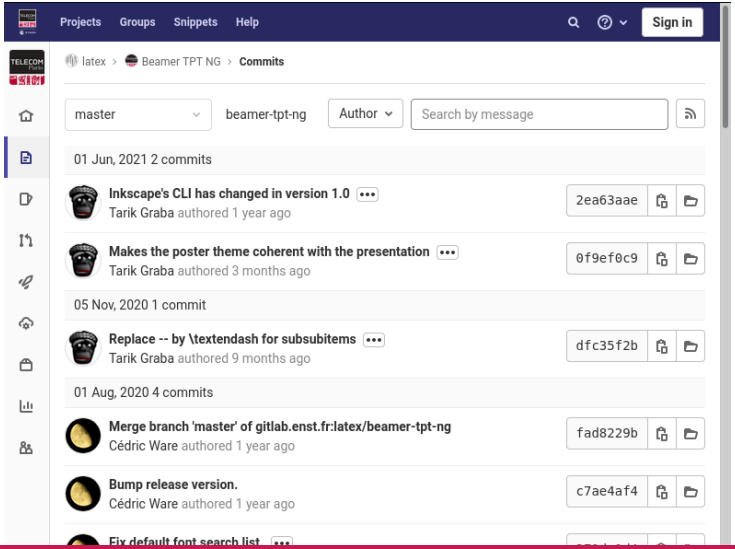
While Git works in a fully decentralized way but whole ecosystems have emerged around it with version control but also :

- access management
- file hosting
- wiki
- pull requests
- continuous integration
- ...

2.3.6 GitHub example



2.3.7 GitLab example



Tools for computer scientists

SSH

2.4.2 SSH PRINCIPLES

Shell : Computers can be controlled with a *Graphical User Interface (GUI)* but historically they can also be controlled from the *Command line*.

Secure SHell (SSH) : Allow to get to the command line to another computer

2.4.3 SSH

SSH has many uses :

- Use computers from afar

  reserved for power users

- Run computation

  useful to run computation in e.g. lab rooms

- Move files

  useful e.g. between the school system and your computer

- Protocol of connection (git)

  GIT can use ssh to “push” your changes

- Tunnel internet (much like a VPN)

  Access pay-walled articles from home

3.1 Recent evolution of AI / LLM

GPT 3 released in 2020 and current models are much better, surely General AI is around the corner!

3.1 Recent evolution of AI / LLM

GPT 3 released in 2020 and current models are much better, surely General AI is around the corner!

We don't know :

- they could plateau (i.e. not get any better than they are now)
- they could slightly improve (but stay at roughly the current level)
- they could improve a lot (but not a GAI)
- GAI could be there tomorrow... or in 100s of years!

What is sure : models and the tooling around them are changing quickly.

3.2 My take on AI and GAI

We don't know what the future holds, therefore :

- there is no need to become an expert in using current systems

  Do prompt engineer have a future ?

- it is very likely that what we teach you will be useful **You know that 2*3=6 even if we have calculators**
- you can use AI in class but try to learn how to do things without AI

My take : AI use might be positive if it helps you to learn not if it replaces learning!

3.2 My take on AI and GAI

We don't know what the future holds, therefore :

- there is no need to become an expert in using current systems

  Do prompt engineer have a future ?

- it is very likely that what we teach you will be useful **You know that 2*3=6 even if we have calculators**
- you can use AI in class but try to learn how to do things without AI

My take : AI use might be positive if it helps you to learn not if it replaces learning !

Remark : For classes, using AI is like copy-pasting from any source : you have to reference it !

3.3 Further points on AI

You are studying AI remember that AI has a lot of ethical considerations :

- ecological impacts
- jobs disruption
- cognitive development
- fake news / spam / *etc.*
- ...

3.3 Further points on AI

You are studying AI remember that AI has a lot of ethical considerations :

- ecological impacts
- jobs disruption
- cognitive development
- fake news / spam / *etc.*
- ...

and of course some positive aspects :

- better science / health / education / *etc.*
- replacing human work
- ...

3.3 Further points on AI

You are studying AI remember that AI has a lot of ethical considerations :

- ecological impacts
- jobs disruption
- cognitive development
- fake news / spam / *etc.*
- ...

and of course some positive aspects :

- better science / health / education / *etc.*
- replacing human work
- ...

More consideration on AI ethics in the AI ethics class :)

Good practices for programming

Writing code

4.1.2 IDE

There exists programming environments allowing you to :

- Have syntax coloring
- Automatic indenting or completion
- Shortcut for compilation/run and navigating through the errors
- ...

Don't write code on basic notepads!

Good practices for programming

Virtualenv

4.2.2 Package python

Advanced classes will often require you to install python packages. To avoid “polluting” your computer you can install them in virtualenv.

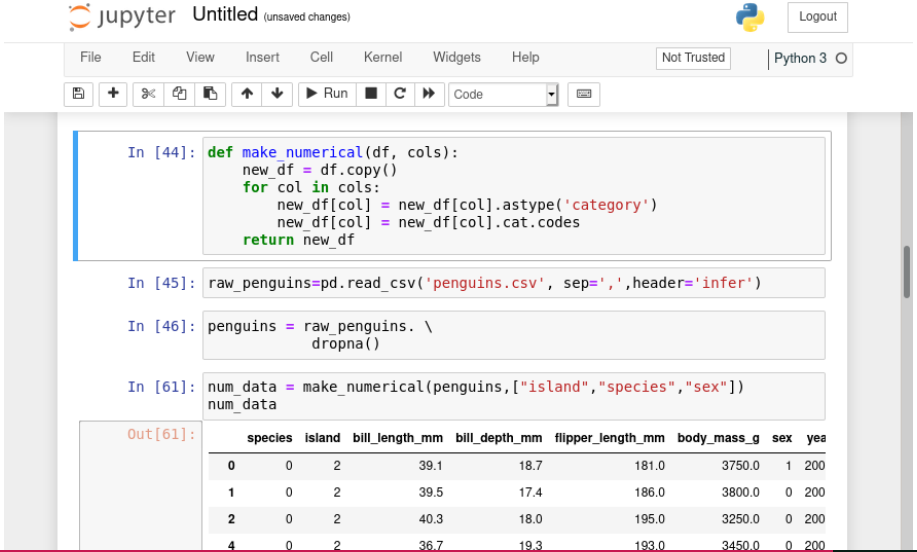
```
#creating the virtual env
virtualenv nameOfTheVirtualenv


# using the virtualenv
source nameOfTheVirtualenv/bin/activate #activation
which pip #check which pip we are using
pip install package name # install package in the venv
python # run python in the venv
jupyter-notebook # run jupyter equipped with the venv
```

Good practices for programming

Notebooks

4.3.2 Jupyter notebooks



4.3.3 Jupyter notebooks

jupyter Untitled (unsaved changes) Logout

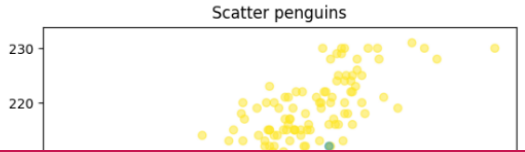
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Run Code

Plotting bill length in function of flipper length (and show species)

In [62]:
```
plt.figure(get_plot())
plt.scatter(num_data['bill_length_mm'], num_data['flipper_length_mm'],
plt.title('Scatter penguins')
plt.xlabel('bill length')
plt.ylabel('flipper length')
plt.show()
```

```
<IPython.core.display.Javascript object>
```



4.3.4 Jupyter notebooks

Notebooks are great for :

- Test quickly code
- Manipulate data
- Visualize data
- Present your work

Overall it is great for “write-only” code.

Notebooks have HUGE limitations :

- You need to know the order in which the cells were launched
- Some piece of code might be deleted
- Hard to test
- Hard to version

4.3.5 Installing Jupyter notebooks

Notebooks are packaged in python :

```
#if using venv:
virtualenv ~/jupyterEnv
source ~/jupyterEnv/bin/activate


# installing them
pip install jupyter
jupyter-notebook
```

Good practices for programming

Writing code

4.4.2 Why write readable code?

Generally, code is **written once** and **read many times** :

- by other people
  - **peer**-review,
  - **homework** correction,
  - inclusion in **other people** code
- also by yourself from the future
  - **debugging**,
  - restarting a paused code session,
  - copying code from the past,
  - etc.

Even when writing “write-only” code, it is a **good exercise** to train to write readable code.

4.4.3 How write readable code ?

The main tools to write readable code are :

- Use standardized code formatting
- Decompose your code into functions / modules
- Naming correctly variables and functions
- Comments

4.4.4 Code formatting

In general, be consistent on :

- indentation
- white spaces / tabulations
- capitalization
- style of functions calls
- style of comments

There are established conventions on code formatting

- see e.g. `https://pep8.org/` for python
- also there are tools to auto-format

4.4.5 Decomposing code

Writing code is all about taking a complex problem and dividing it recursively into subproblems.

Consequences :

- Don’t reinvent the wheel each time
- Decompose your big projects into smaller projects
- Decompose your code into files
- Decompose your files into functions and classes
- Only a handful of variables should be global
- Files and functions should remain “small”

However, don’t overdo this...

4.4.6 Naming functions / variables

Functions should explain what they do from a high level perspective :

- ☒ doStuff

4.4.6 Naming functions / variables

Functions should explain what they do from a high level perspective :

- ☒ doStuff
- ☑ ComputeVAT

4.4.6 Naming functions / variables

Functions should explain what they do from a high level perspective :

- ☒ doStuff
- ☑ ComputeVAT
- ☒ SumMultiplyCategoryToPrices

4.4.6 Naming functions / variables

Functions should explain what they do from a high level perspective :

- ☒ doStuff
- ☑ ComputeVAT
- ☒ SumMultiplyCategoryToPrices
- ☑ cleanAbort
- ☑ loadDataset
- ☒ readFile

Variables should explain what they store :

- ☒ ff

4.4.6 Naming functions / variables

Functions should explain what they do from a high level perspective :

- ☒ doStuff
- ☑ ComputeVAT
- ☒ SumMultiplyCategoryToPrices
- ☑ cleanAbort
- ☑ loadDataset
- ☒ readFile

Variables should explain what they store :

- ☒ ff
- ☑ nbFalsePositives

4.4.6 Naming functions / variables

Functions should explain what they do from a high level perspective :

- ✗ doStuff
- ✓ ComputeVAT
- ✗ SumMultiplyCategoryToPrices
- ✓ cleanAbort
- ✓ loadDataset
- ✗ readFile

Variables should explain what they store :

- ✗ ff
- ✓ nbFalsePositives
- ✗ numberOfWrongItems, GoodItemsCount (choose one)

4.4.6 Naming functions / variables

Functions should explain what they do from a high level perspective :

- ✗ doStuff
- ✓ ComputeVAT
- ✗ SumMultiplyCategoryToPrices
- ✓ cleanAbort
- ✓ loadDataset
- ✗ readFile

Variables should explain what they store :

- ✗ ff
- ✓ nbFalsePositives
- ✗ numberOfWrongItems, GoodItemsCount (choose one)
- ✓ cleanAbort

4.4.7 Comments

A good comment is one that :

- provides some **documentation** on the code
- explain the **structure** of the code
- provides **insights** on some part of the code
- explain a piece of code from a **high-level** perspective
- Does **not repeat** the code

4.4.7 Comments

A good comment is one that :

- provides some **documentation** on the code

  This function expects two integers x and y and returns x × y

- explain the **structure** of the code

  Here we load the data, here we train the model, etc.

- provides **insights** on some part of the code

  x+=1 # This accounts for the top square

- explain a piece of code from a **high-level** perspective

  Here we try to pre-compute the distance of all points to A

- Does **not repeat** the code

  computeShortestPath() # will compute shortest path

4.4.8 Is it useful for masters?

You will mainly have write-once read-once code this year but you should start using good practices because writing does not take more time and you will need it someday!

Good practices for programming

Testing your code

4.5.2 Tests

Unit tests Each unit test tests some very specific piece of code to make sure it works ok.

e.g. test a function or a member of a class

Functional tests Runs the whole program and make sure the output is what was expected.

perfect to test algorithms

Regression tests Each time a bug is discovered, introduce tests to make sure this bug will not reappear.

4.5.3 Example of unittest package in Python

See https://docs.python.org/3/library/unittest.html

```python
import unittest

class TestStringMethods(unittest.TestCase):

    def test_upper(self):
        self.assertEqual('foo'.upper(), 'FOO')

    def test_isupper(self):
        self.assertTrue('FOO'.isupper())
        self.assertFalse('Foo'.isupper())

    def test_split(self):
        s = 'hello world'
        self.assertEqual(s.split(), ['hello', 'world'])
        # check that s.split fails when the separator is not a string
        with self.assertRaises(TypeError):
            s.split(2)

if __name__ == '__main__':
    unittest.main()
```

4.5.4 Testing in practice

Not using framework to test your code is **unthinkable** for professionals however for a short piece of code for a class you can test manually :

- check that the output has the correct format
- check on small examples it makes sense
- test the output is reasonable
- if possible run manually a few functional tests before submitting your work

5.1 Exercises

- Clone the following repository and create a short presentation : `https://gitlab.telecom-paris.fr/latex/beamer-tpt-ng`
- Read the pep8 (`https://pep8.org/`) and read and do some of the exercises here `http://web.mit.edu/6.005/www/fa15/classes/04-code-review/`
- Create a repository on Télécom's gitlab (use the sign-in with Shibboleth and use your Télécom's account
- Install a password manager (e.g. KeepassX)
- Start doing backups
- Make sure your disk is encrypted (if not consider doing it someday)
- Write yourself a résumé e.g. using LaTeX(see `https://www.overleaf.com/latex/templates/tagged/cv` for templates

5.2 This afternoon!

Logic is used in :

- Maths
- Electronics
- Computer science (theorem proving, proofs of programs)
- Knowledge Representation & semantic Web
- high-level NLP, Law, explanation, argumentation
- Database management

Logic does not hallucinate

Louis Jachiet

09 September 2025

louis.jachiet@telecom-paris.fr