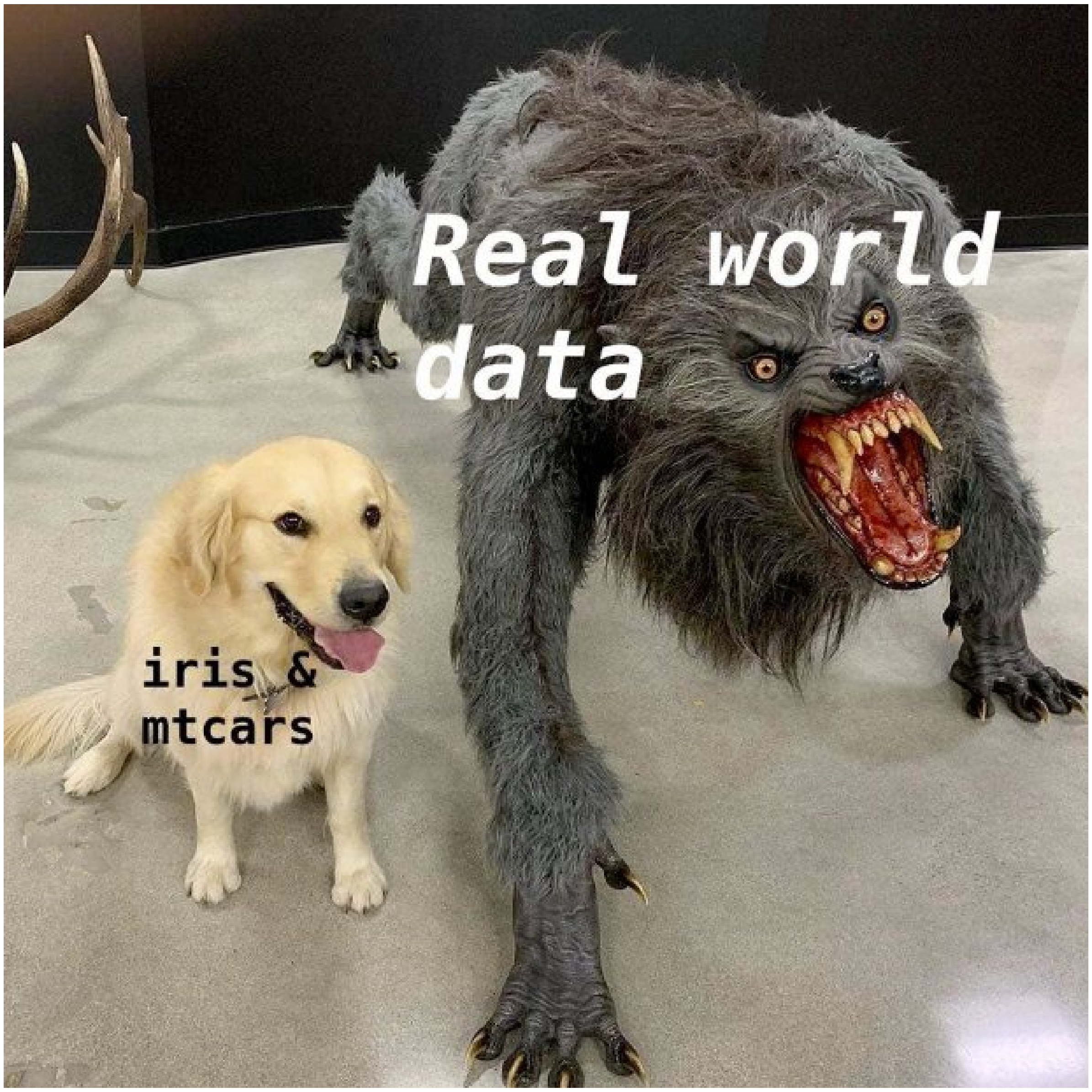
Conducting project & data management

Émilien Schultz - SICSS Paris 2025

Real data are dangerous



Share your experience

- What kind of data do you use ?
  - how big, kind of format ?

- What is your best advice regarding data management you received / you can share ?

Tips from the room

- Create intermediate records & backups
- Write specific scripts for each step
- Chuck your dataset, like on per day if its a continuous
- Start clean because even short projet quickly become bigger
- Add clear meta data & file names
- Ethic is important, know from where your data come from
- Not always "the most the better"
- Look for dedicated tools rather than write everything by yourself
- Version control is useful
- Favor open source tools and if you can give back to the community

Focus on data management

Practical skills

- Collecting data
- Transforming data to digital text
- Downsizing large dataset
- Design a workflow

So many questions...

- How to scrap online data ?
- What are the best format to use ?
- What is encoding ?
- What can I do if my dataset is too large ?
- What can I do if my scripts take too much time ?
- What if my computer is too old ?
- How to manage my scripts ?
- ...

We can't discuss all of them

- Digitalization of data
- Fixing OCR errors
- Code optimization / modularization

- Version control for code (git, ...)
- Data legislation ...

But we will be here if you have specific questions

First, document your project

Especially true in collaborative projects

The importance of the filesystem

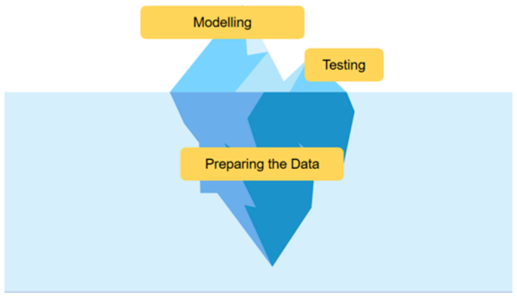
The ideal file system

```
project-name/

   README.md
   data/
       raw/                     # Unprocessed, original data
       intermediate/            # Data after cleaning
       final/                   # Cleaned, analysis-ready data
   src/
       notebooks/
       scripts/
   docs
   outputs/
        figures/
        tables/
        reports/
```

Second, data management

Data architecture : how to organize your data ?



Which data format should you use ?

- CSV or Parquet ?
  - CSV : easy to read by chunck
  - parquet : optimized and compressed

In all case :

- one row per document
- clear index

Sizes of dataset for social sciences

- Tiny ~ 100 Mb
- Small ~ 1 G
- Medium ~ 10 G
- Large ~ 100 GB

(for CS, < 10G is small dataset)

Help yourself : avoid big data

The bigger, the costier (to compute)

Aim to small data for a start

Data manageable on your computer - for a start

Third, clean your data

- Know your data
  - yes, read some of them
- Fix encoding problems
  - or unknown characters
- Deduplicate / remove empty elements

- Remove “noise” : HTML tags, headers

Do you need to remove some elements : emoticons ? stopwords ?

How to do it ?

- Dedicated Python scripts
  - write specific functions to do the cleaning
  - reminder : `regex` can be useful
- Dedicated tools (Open refine)

Avoid to correct your data manually

Build a workflow - write some scripts

BAD DATA
MANUAL FIX
BAD DATA
imgflip.com

What if you data is too big ?

Solution 1 : reduce your data

- Work on subset
- Sample
- Batch

Solution 2 : parallelize

- Dedicated data architecture
- Parallelize (Dask)
- Dedicated tools : Xan for csv

How to sample correctly ?

- Randomly
  - Risk to miss rare elements
- Stratify for skew data

Do you need all your text ?

- Reduce your dataset to optimize treatment
  - Part that contains specific keywords
  - Introduction, conclusion, …
- The limit of models context windows : 512, 1024, more ?
  - For instance CamemBERTv2

How to divide a text ?

The different levels of segmentation

- Complete document
- Paragraph
- Sentence

How to do it :

- brute force : just cut it :)
- intelligent segmentation

Brute force

- Rules of thumb : 1 token ~= 4 chars in English
- Compute the number of token with a model
- Then cut the text

Risk : creating meaningless parts of the text

How to perform intelligent segmentation

Use dedicated models/packages

- Generic : SpaCy
- Specific : wtpsplit

-> *Examples in the notebook*

Balance computation cost and model size

Depending on your task, you need to make choices.

- Large models are :
  - powerful / generic
  - dependant on infrastructures
  - slower
- Small models are :
  - specific
  - faster
  - easy to deploy

How much time will it need to run ?

Have an estimation of your process on a sample

- Use `time` module or `%timeit` magic
- If you use API requests, estimate tokens/cost

How to start ?

- Define a small **sample dataset** that could handle easily
- Define a **baseline** with simple strategy
  - regex, basic ML, …
- Build a first workflow with a **simplified question**
  - e.g. dichotomize your classification
- Develop your intuition

How to continue ?

- Design a annotated **test set** and a testing workflow
- Evaluate the cost expected on the complete dataset
- Run your process on the complete dataset