

# 551-1119-00L Microbial Community Genomics

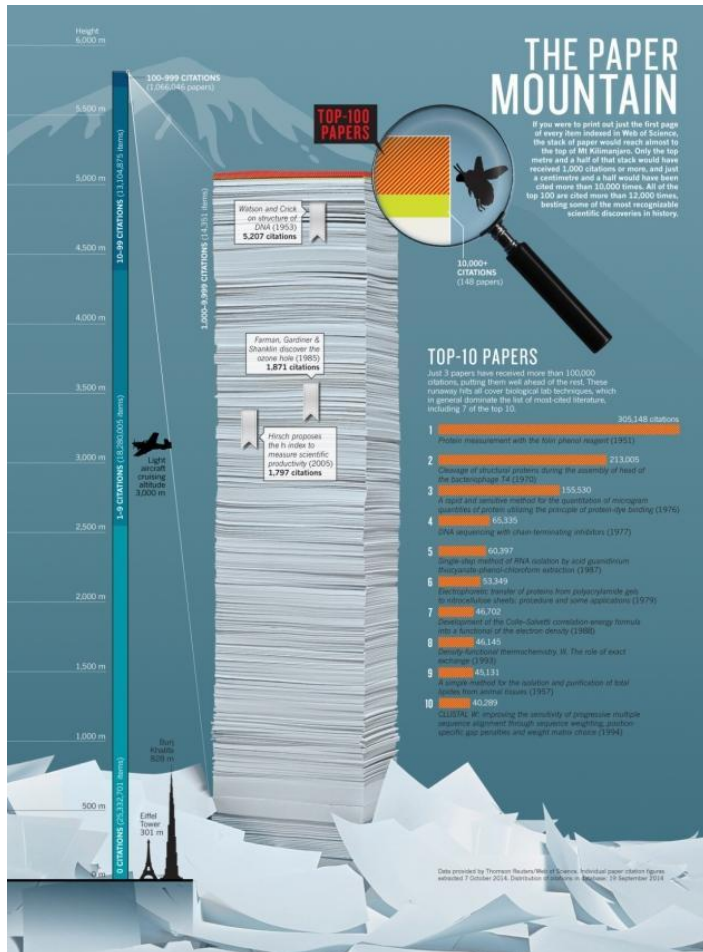
Lecture:

Best practices for data/project/software management

Samuel Miravet-Verde

12-Nov-24

# 0. Computational biology || Why 'good practices' are required?

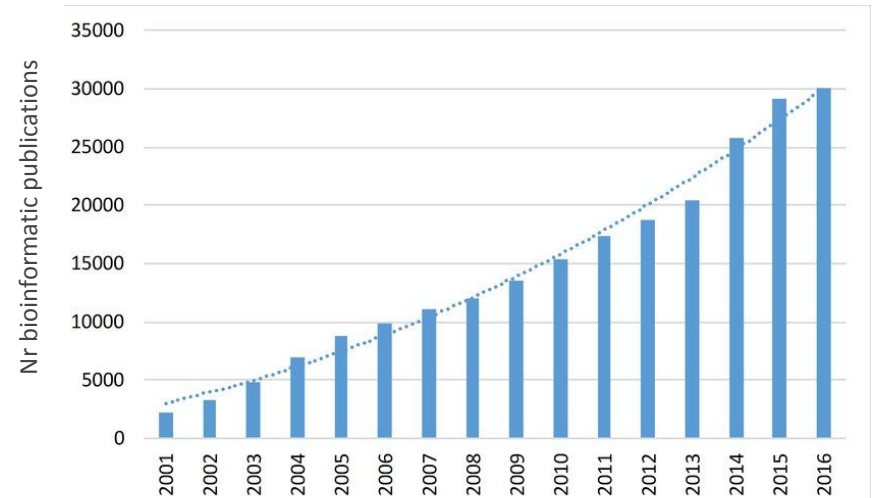


**ALGORITHM:** a process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer.

3 bioinformatics papers in the top 15 most cited papers:

- 10<sup>th</sup> : ClustalW (1994) (40,000 cit. in 2014)
- 12<sup>th</sup> : BLAST (1990)
- 14<sup>th</sup> : PSI-BLAST (1997)

High-reaching tools, we have to ensure bioinformatic tools and analyses done with these tools are properly done



## 0. Computational biology || Why 'good practices' are required?

### What's the point of writing good scientific software?

“**Software written by academics** has a reputation of being **poorer quality** than that software written by professional **software developers**”

“Well documented software takes **extra time** and in a competitive academic job market I feel like this is a **luxury.**”

### How Not to Be a Bioinformatician

“**Stay low level at every level.** Develop your code by anecdote: avoid planning phases, requirement analysis exercises or any structure to your code.”

“If you create the application, **make it difficult to build and interpret.** Have plenty of hidden dependencies and bizarre variables.”

# 1. The data analysis process || Basic workflow to analyse data

## 1. Define the question

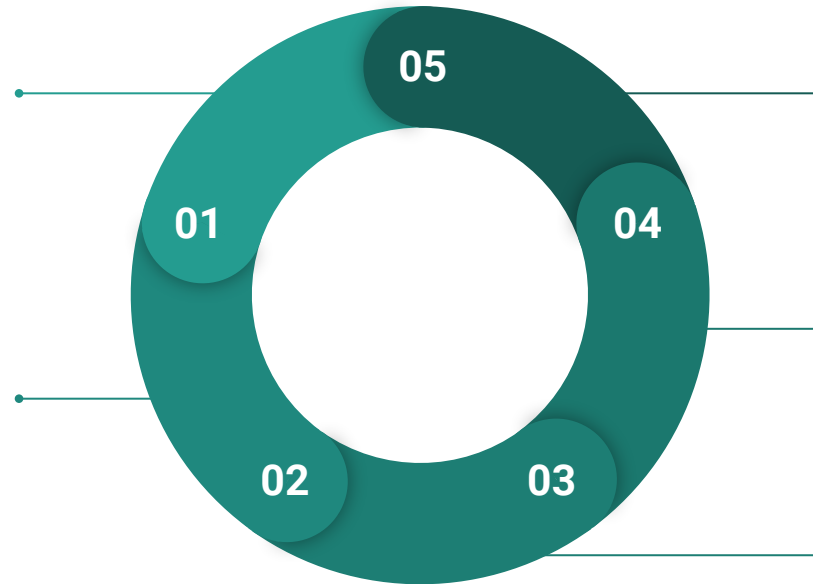
Together with supervisor and/or collaborators

Plan experiments and potential roadblocks to deal during development

Consider the assumptions you will be taking during your analysis

## 2. Data collection

Either from samples or public resources (there are MANY!)



## 5. Sharing results

Summarize remarkable results, helped by text, tables and visualizations

Keep track of intermediary and non-reported results

## 4. Analyze the data

Apply the planned analyses.

Fail Early, Fail Fast → be flexible and adjust to data requirements

## 3. Clean Data

Identify outliers, non-annotated data, etc. and remove it

Discarded data can still be used for other purposes (e.g., controls)

# 1. The data analysis process || Structuring a working directory

Structuring and keeping your working directory organized is essential to ensure **reproducibility**...

... Also for your mental health (and possible collaborators or future contributors!)

A good (and flexible) example:

For your projects:

```
.
├── config.yml
├── data
├── envs
├── LICENSE
├── README.md
├── reports
├── results
├── src
├── start_project.sh
└── workflows
```

[MiqG/project\\_template: Template to give structure to new projects from the start \(github.com\)](#)

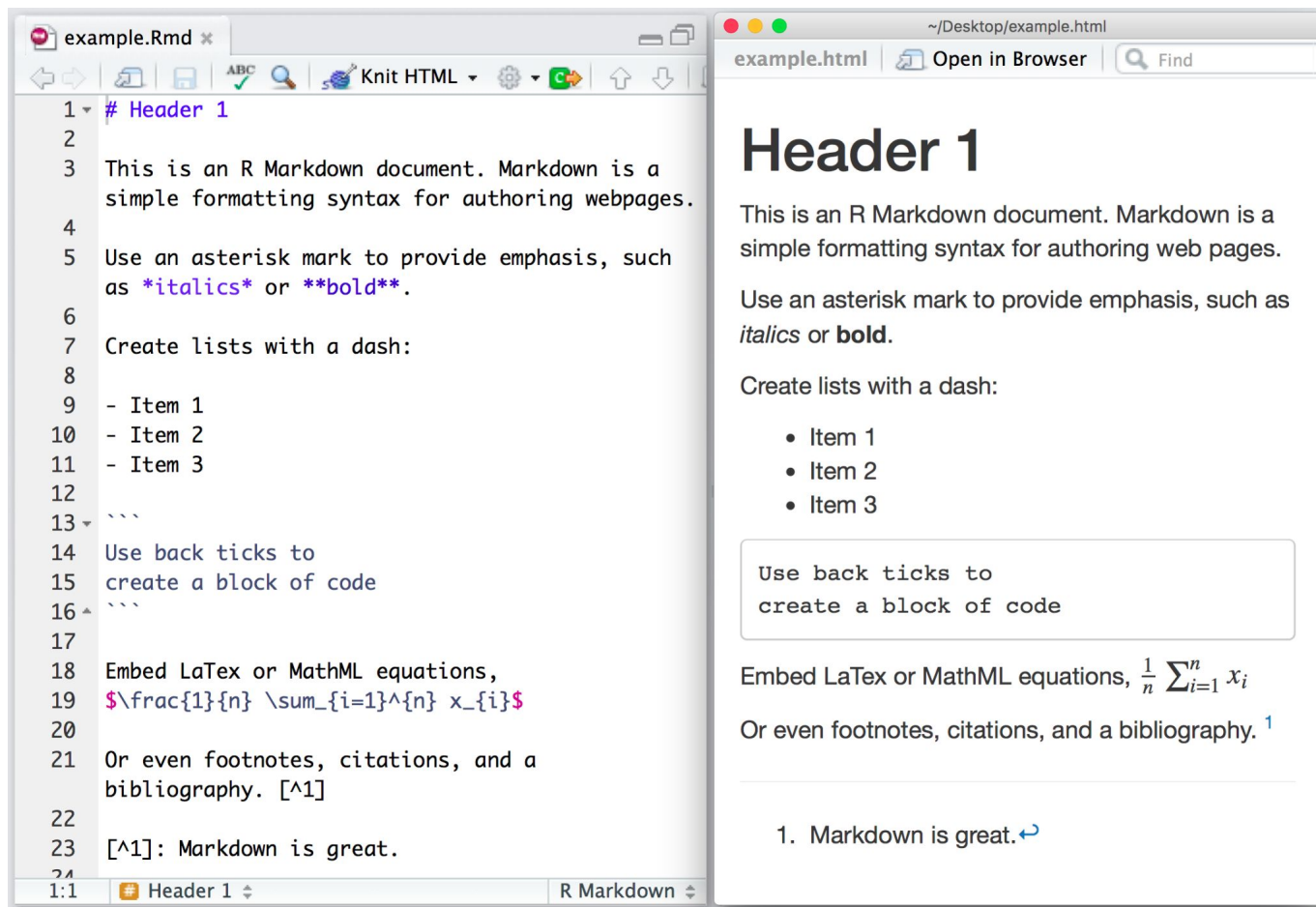
```
.
├── data
├── reports
├── results
├── scripts
└── analysis.ipynb
```

General:  
meaningful naming  
for files

- **Data:** original source of information, also processed tables
- **Reports:** documents summarizing results
- **Results:** figures, final tables
- **Scripts:** source code of functions and procedures you run on the data
- **Analysis:** your main document including all the analysis workflow



## 2. Documenting and reporting | | Embedding code in markdown



The image shows two side-by-side windows. The left window is an R Markdown editor showing the source code for 'example.Rmd'. The right window is a browser displaying the rendered HTML output for 'example.html'.

**Left Window (Source Code):**

```
1 # Header 1
2
3 This is an R Markdown document. Markdown is a
  simple formatting syntax for authoring webpages.
4
5 Use an asterisk mark to provide emphasis, such
  as italics or bold.
6
7 Create lists with a dash:
8
9 - Item 1
10 - Item 2
11 - Item 3
12
13 ```
14 Use back ticks to
15 create a block of code
16 ```
17
18 Embed LaTeX or MathML equations,
19  $\frac{1}{n} \sum_{i=1}^n x_i$ 
20
21 Or even footnotes, citations, and a
  bibliography. [^1]
22
23 [^1]: Markdown is great.
24
```

**Right Window (Rendered HTML):**

# Header 1

This is an R Markdown document. Markdown is a simple formatting syntax for authoring web pages.

Use an asterisk mark to provide emphasis, such as *italics* or **bold**.

Create lists with a dash:

- Item 1
- Item 2
- Item 3

Use back ticks to create a block of code

Embed LaTeX or MathML equations,  $\frac{1}{n} \sum_{i=1}^n x_i$

Or even footnotes, citations, and a bibliography. <sup>1</sup>

---

1. Markdown is great. ↩

## 2. Documenting and reporting | | Embedding code in markdown

```
In [1]: %matplotlib inline
import pandas as pd
import numpy as np
import plotly
from IPython.display import display, Markdown as md
```

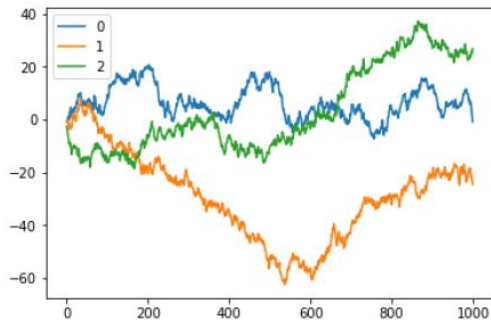
```
In [2]: title = "My Shiny Report"
x = 1000
y = 3
```

```
In [3]: display(md("# Just look at this graph from {}".format(title)))
```

### Just look at this graph from My Shiny Report

```
In [4]: df = pd.DataFrame(np.random.randn(x, y))
df.cumsum().plot()
```

```
Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x7f127adda278>
```



<https://www.rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheat-sheet.pdf>

Extra:

You can call Bash (Unix Shell) programs by calling `system`:

```
# In R
system('ls')
```

```
# In python
import os
os.system('ls')
```

Extra 2:

```
%load_ext autoreload
%autoreload 2
```



### 3. Data integrity || Keeping data & code safe

Data integrity refers to **the reliability and trustworthiness of data throughout its lifecycle**. It can describe the state of your data or even your code → **ALWAYS KEEP YOUR RAW DATA SAFE**

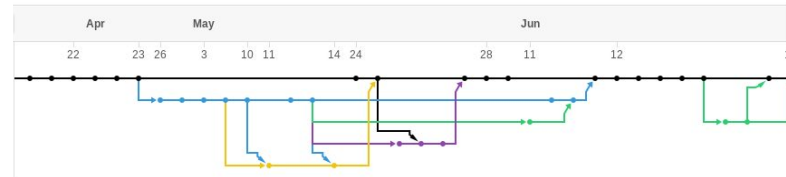
**Digital Object Identifiers** (DOI) are useful to keep track of data versions (and publications, code, etc...)

- [Zenodo](#) can generate a DOI associated to your dataset, processed data, programs...



**Version control** allows to keep track of changes in your code by storing it together the modifications one or more users perform on a file (even if they happen at the same time) → **REPOSITORIES**

- [Github](#) is the most widely used platform. It allows
  - **branching** → to define new functions for your code without breaking the main program/analysis
  - **forking** → copying repositories made by other people
  - **wiki** → markdown edition to document your code
  - **issues** → direct communication with the developers of a tools to ask for new functionalities, report bugs...



<https://docs.github.com/en/get-started>

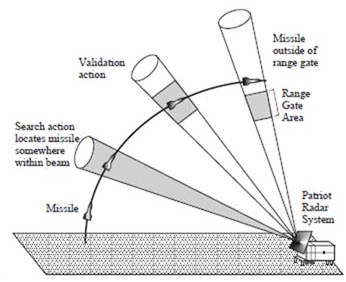
# 4. Bugs | | Types of bugs to take into account when coding



- Grace Murray Hopper, Sept. 9<sup>th</sup>, 1947 → “**First actual case of bug being found**” in the Harvard Mark II computer.

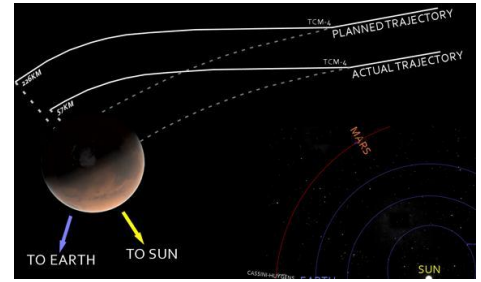
Computer crashed

- Now:
- **Syntax errors:** cannot start
  - **Runtime error:** cannot finish
  - **Semantic errors:** the program runs and produces an output... BUT IT IS NOT THE EXPECTED.



**Dhahran Patriot Missile assumption,**  
 1991: minimal round → 1/3 of delay / 100h

Death of 28 soldiers from the U.S. Army



**Mars Climate Orbiter incompatibility,** 1998:

- NASA : metric system
- External Software : US customary units

Collision in Mars for \$327.6M



**Y2K bug (lack of foresight),** 2000:  
 “19” before year variables → waste of memory

Incalculable, not only money

# 551-1119-00L Microbial Community Genomics

Lecture:

Best practices for data/project/software management

Samuel Miravet-Verde

12-Nov-24