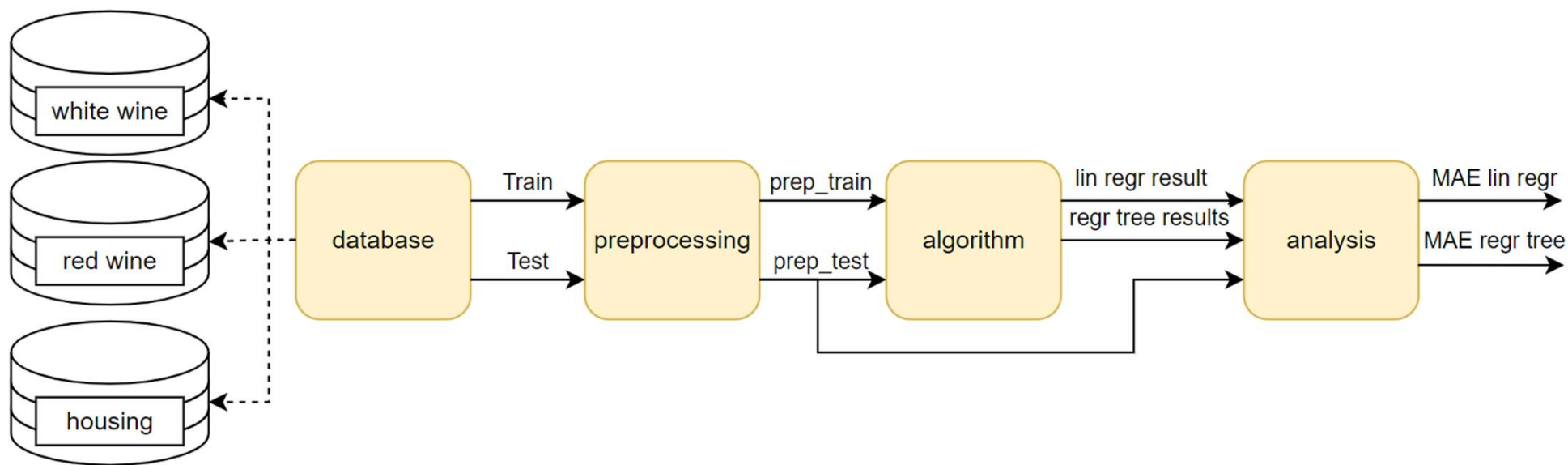# SaRu

Samuel Michel

Ruben De Campos

Introduction

- Github project :
  - Small project description and steps to follow
  - Everything necessary to reproduce the work
    - Environment
    - Data
    - Code

- Small code to try to get it to work

- Goal : split the project in appropriate files as in the designed workflow

Implementation

- Basic workflow
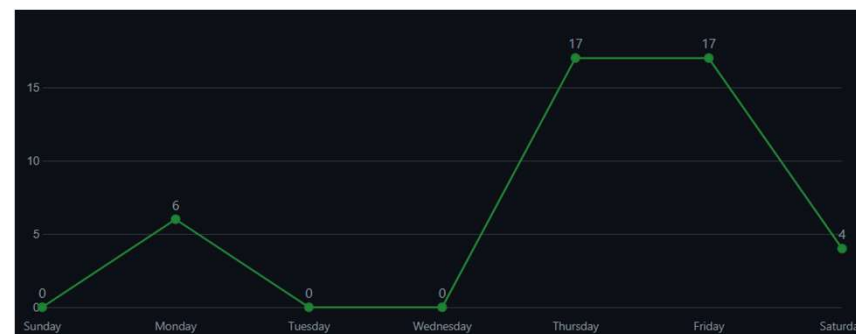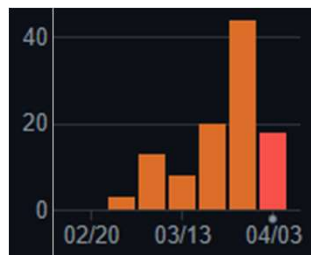
VC Git and Code sharing

- Github project :
  - Commit often
  - Work on different branch
  - Keep everything tidy

Insight :

```
github.com/AlDanial/cloc v 1.82  T=0.02 s (1304.2 files/s, 105282.9 lines/s)
-------------------------------------------------------------------------------
Language                        files          blank        comment           code
-------------------------------------------------------------------------------
Python                             11            332            180            435
reStructuredText                    8            209            294            153
YAML                                1             19             15             84
DOS Batch                           1              8              1             26
make                                1              4              7              9
-------------------------------------------------------------------------------
SUM:                               22            572            497            707
-------------------------------------------------------------------------------
```

Nbr of blank line, comment and code using 'cloc'

70%

ratio !

Nbr of commit histogram + graph of our most productive week

VC Git and Code sharing

- Clone of the github repository locally

- Collaborative work

Git log of 3 commits

VC Git and Code sharing

- Graphical view of our commits

- Working on different branches :
  - Dev
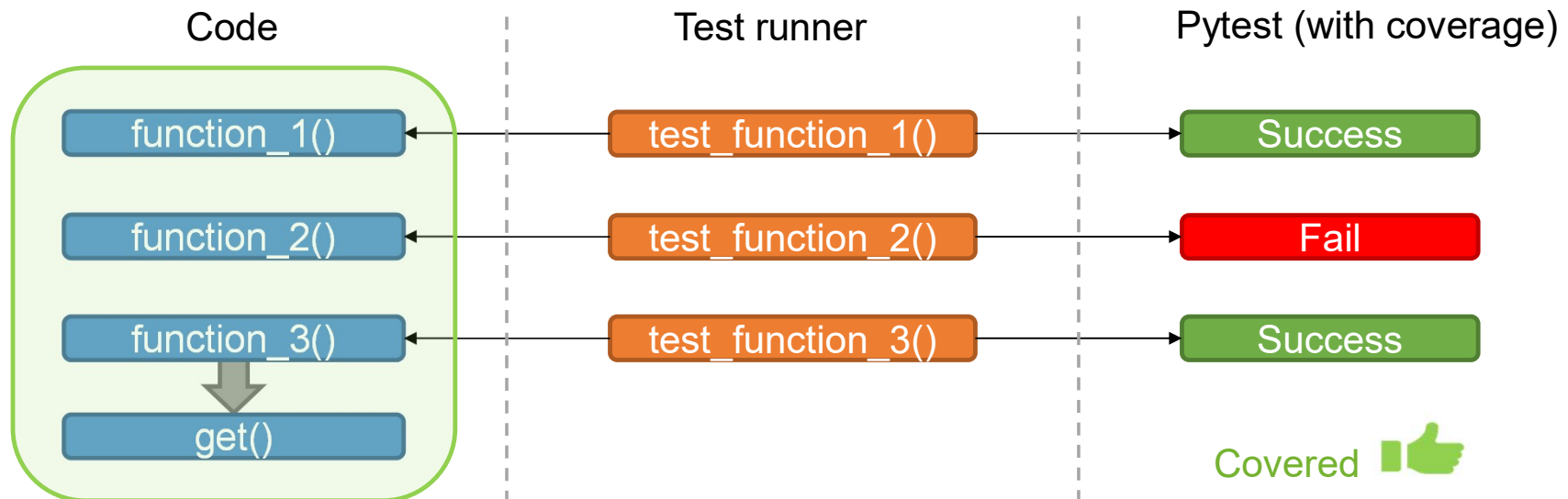 - Features
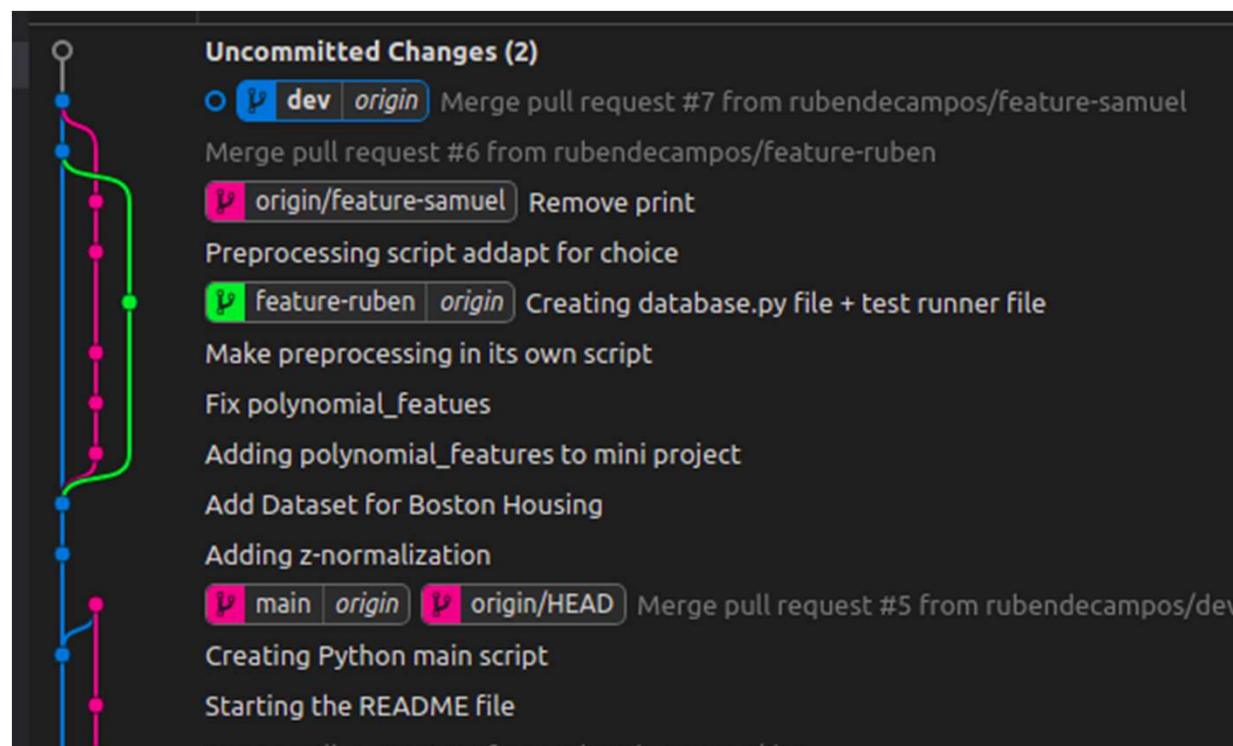
Visual Studio Code's graphical view of the git branches

Unit Testing

| Code	Test runner	Pytest (with coverage)
function_1()	test_function_1()	Success
function_2()	test_function_2()	Fail
function_3()	test_function_3()	Success
get()		Covered 👍

```python
def test_error_0():
    '''Test error=0'''

    expect = 0
    error = analysis.analyser(np.array([0,1,2]),np.array([0,1,2]))
    assert np.isclose(error,expect)
```
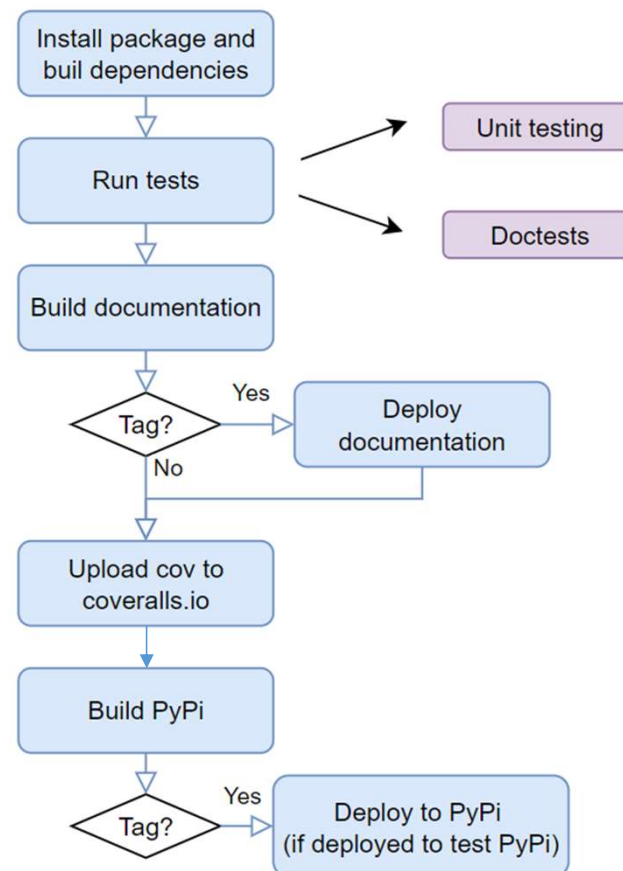
Continuous Integration

- Action to verify each modification made to the implementation

Good practice for CI
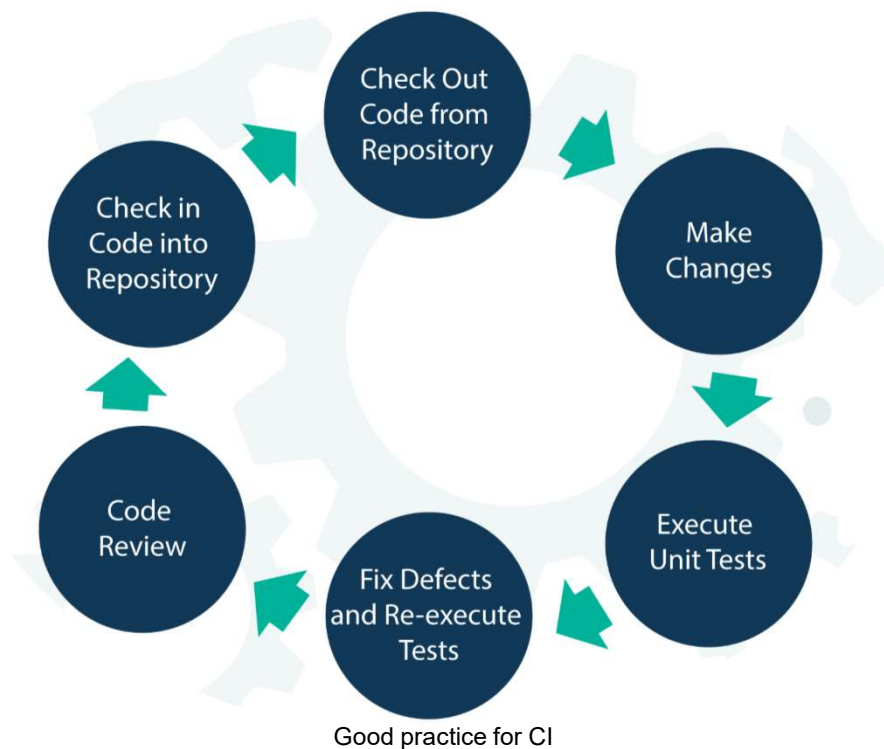
Continuous integration

Job workflow:

- on push

Test results:

```
Name                Stmts   Miss   Cover
-----------------------------------------
algorithm.py           32      9     72%
analysis.py            11      1     91%
database.py            33      9     73%
preprocessing.py       30      2     93%
run.py                 39     39      0%
test_algorithm.py      52      0    100%
test_analysis.py       35      0    100%
test_database.py       24      0    100%
test_prepro.py         80      0    100%
-----------------------------------------
TOTAL                 336     60     82%
```
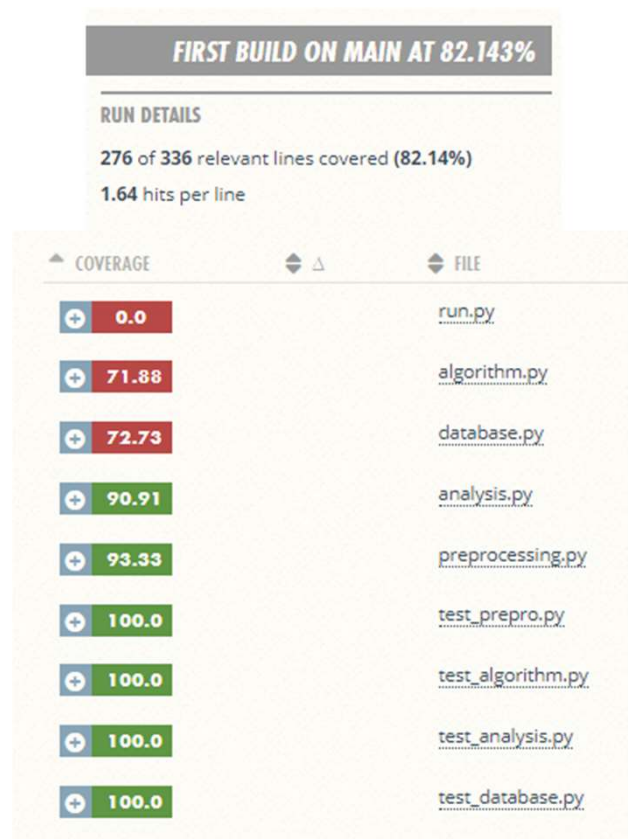
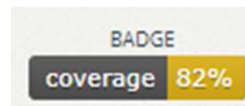Unit testing results (w/ coverage)

```
Doctest summary
===============
    12 tests
     0 failures in tests
     0 failures in setup code
     0 failures in cleanup code
build succeeded.
```

Doctests results

Coveralls.io

BADGE

coverage 82%

FIRST BUILD ON MAIN AT 82.143%

RUN DETAILS

276 of 336 relevant lines covered (82.14%)

1.64 hits per line

COVERAGE	Δ	FILE
0.0		run.py
71.88		algorithm.py
72.73		database.py
90.91		analysis.py
93.33		preprocessing.py
100.0		test_prepro.py
100.0		test_algorithm.py
100.0		test_analysis.py
100.0		test_database.py

Coverage from coveralls.io

90.91 /analysis.py

```python
1   from sklearn import metrics                                              2×
2   import numpy as np                                                       2×
3
4   '''analysis module that compare the predicted result with the real result'''
5
6   def analyser(predict,real):                                              2×
7       """Perform the absolute error between the prediction of the algorithm
8       and the real data.
9
10      Parameters
11      ----------
12      predict : pandas.dataframe
13          predicted value, given by our algorithm
14      real : Pandas.dataframe
15          real value (from the test_set)
16
17      Returns
18      -------
19      error : float
20          the absolute error"""
21
22      if(not isinstance(predict,np.ndarray)):                              2×
```

Documentation

Welcome to SaRu's documentation!

Continuous Integration `passing` | coverage `84%` | downloads `15/month` | pypi `v1.0.1` | docs `latest` | report `issues`

- Deployed to Github pages for every push followed by a 'tag' (new version)

- This is the 'face' of our software, that the users will use to understand it.

It explains:

1. How to install it

2. How to use it

3. Every functions and modules

4. Everything that is important to note

Documentation

- Generate the documentation with 'Sphinx' (doc written in RestructuredText)

```
$ sphinx-quickstart doc
$ sphinx-build doc sphinx
```

- Docstrings written in the code generate documentation automatically

```
.. automodule:: database
    :members:

.. automodule:: preprocessing
    :members:

.. automodule:: algorithm
    :members:

.. automodule:: analysis
    :members:
```

Generation of the API

Documentation

A docstring comment:

Its documentation:

```
def create_protocols(data, n):
    """Create n protocols, each protocol being split in a training set and a test set (50/50)
    !It clears the content of the 'protocols' dictionary!

    Parameters
    ----------
    data : pandas.dataframe
        the dataset to split into 50% of training and 50% and test
    n : int
        number of protocols to create, they are labeled 'proto1', ..., 'proton'

    Returns
    -------
    protocols : dict
        a dictionary containing all the protocols and their subset
    """
```

Database

database.create_protocols(*data, n*)   [sour

Create n protocols, each protocol being split in a training set and a test set (50/50) !It clears the content of the 'protocols' dictionary!

Parameters:	• **data** (*pandas.dataframe*) – the dataset to split into 50% of training and 50% and test • **n** (*int*) – number of protocols to create, they are labeled 'proto1', …, 'proton'
Returns:	**protocols** – a dictionary containing all the protocols and their subset
Return type:	dict

database.get(*protocol, subset*)   [sour

Get the training or testing set from a protocol

Parameters:	• **protocol** (*str*) – label of the protocol to get (e.g. 'proto1') • **subset** (*str*) – either 'test' or 'train'
Returns:	**the test or training set from a protocol**
Return type:	pandas.dataframe

Documentation

Doctests:

```
.. testsetup::

    import subprocess
    run_commandline = lambda cmd: print(subprocess.check_output(cmd, shell=True).decode())
```

```
.. testcode::

    run_commandline('python run.py --dataset white-wine --prep minmax')
```

```
.. testoutput::
    :options: +NORMALIZE_WHITESPACE

    'proto0' table for the 'white-wine' dataset
    Using the 'minmax' method :
    ------------------------------------------------
    Absolute error using Linear Regression  |  0.10
    Absolute error using Regression Tree    |  0.09

    'proto1' table for the 'white-wine' dataset
    Using the 'minmax' method :
    ------------------------------------------------
    Absolute error using Linear Regression  |  0.10
    Absolute error using Regression Tree    |  0.09
```

Displayed in the documentation:

Results using the minmax method:

```
run_commandline('python run.py --dataset white-wine --prep minmax')
```

Result:

```
'proto0' table for the 'white-wine' dataset
Using the 'minmax' method :
------------------------------------------------
Absolute error using Linear Regression  |  0.10
Absolute error using Regression Tree    |  0.09

'proto1' table for the 'white-wine' dataset
Using the 'minmax' method :
------------------------------------------------
Absolute error using Linear Regression  |  0.10
Absolute error using Regression Tree    |  0.09

'proto2' table for the 'white-wine' dataset
Using the 'minmax' method :
------------------------------------------------
Absolute error using Linear Regression  |  0.10
Absolute error using Regression Tree    |  0.09
```

Packaging

New structure:

Github repository

SaRu code

Manually upload:

- Zip the package

- Upload to (test)PyPi

```
# creates a zip of the package
$ python setup.py sdist
# checks if the package is sane
$ twine check dist/rr-2.0.0.tar.gz
Checking dist/rr-2.0.0.tar.gz: PASSED
# upload to server
$ twine upload -r testpypi dist/rr-2.0.0.tar.gz
```

github action:

pypa/gh-action-pypi-publish@master

License

Licensed under the 3-clause BSD License

- Permissive license

- One of the most popular

- Few restrictions

- 3 for 3 clause (remove the acknowledgement)

    "this product include software developed by …"

Useful links

- Cloc: https://github.com/AlDanial/cloc

- PowerBi: https://app.powerbi.com

- SaRu: https://github.com/rubendecampos/mini-project

- Coverage: https://coveralls.io/github/rubendecampos/mini-project?branch=main