ActiGraph™

PIONEERING THE DIGITAL TRANSFORMATION OF CLINICAL RESEARCH™

Reproducible data processing

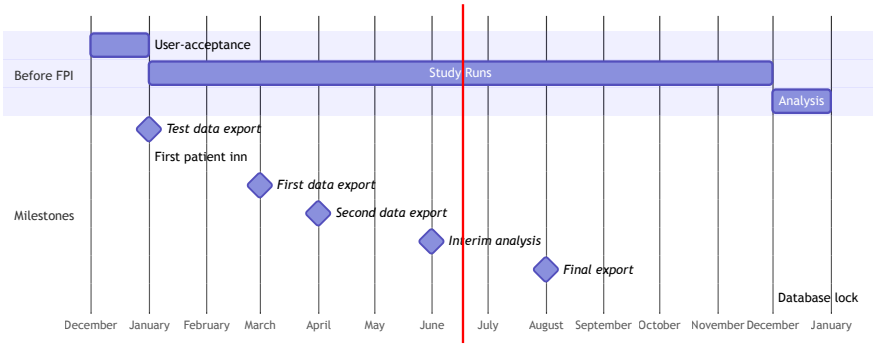
Ali Neishabouri

Outline

1. What do we mean by reproducibility?
2. During the design
3. Implementation
4. Third-party dependencies
5. Platform

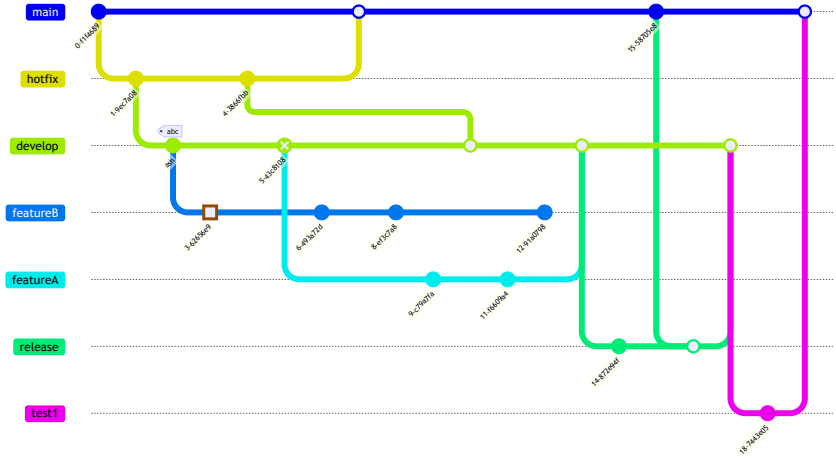
At ActiGraph, we deal with clinical studies

Life of a clinical study



Data can be processed on different platforms

Code changes between executions



Dependencies change between executions

Example (Updating Pandas [6])

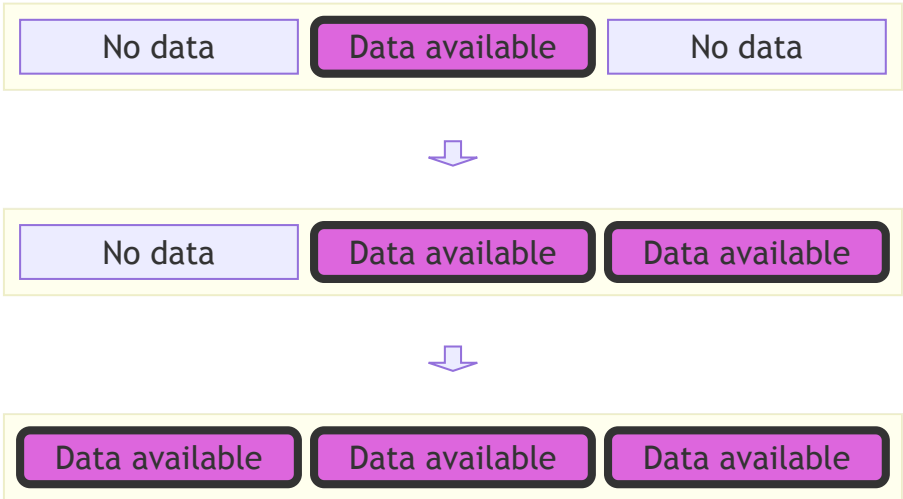
v2.1.4

```
,wear_starts,wear_ends
/,vanhees2013,2022-03-30 11:30:00-0500,2022-03-30 19:29:58.933-0500
/,vanhees2013,2022-03-30 21:59:58.600-0500,2022-04-01 16:14:52.866-0500
```

v2.2.0

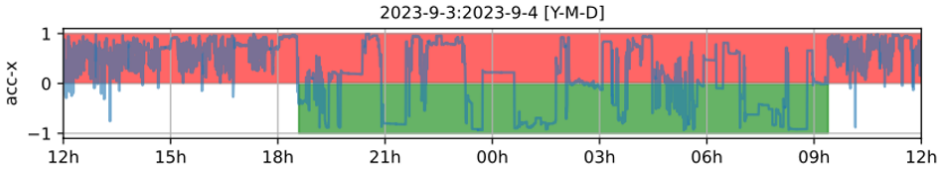
```
wear_starts,wear_ends
',vanhees2013,2022-03-30 11:30:00-0500,2022-03-30 19:29:58.933-0500
',vanhees2013,2022-03-30 21:59:58.599-0500,2022-04-01 16:14:52.866-0500
```

More data comes in between executions, even out of order

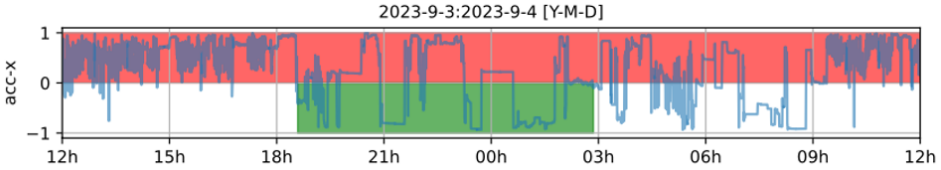


We couldn't get this to work

Time in bed detected by scikit-digital-health [1]

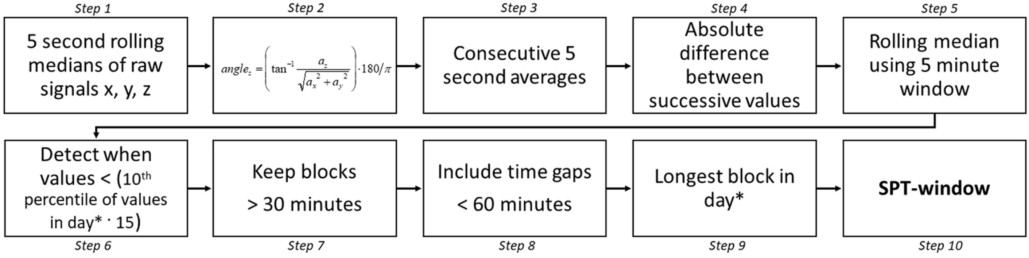


Same thing, with one sample removed at the beginning



ActiGraph

Sneaky things happen [8]



Appending data at the beginning will shift all windows

So what should we do?

- ☐ The design phase

Beware of how your windows are defined

Example ('Grouper' in Pandas [6])

origin : ***Timestamp or str, default 'start_day'***

The timestamp on which to adjust the grouping. The timezone of origin must match the timezone of the index. If string, must be one of the following:

- 'epoch': *origin* is 1970-01-01
- 'start': *origin* is the first value of the timeseries
- 'start_day': *origin* is the first day at midnight of the timeseries
- 'end': *origin* is the last value of the timeseries
- 'end_day': *origin* is the ceiling midnight of the last day

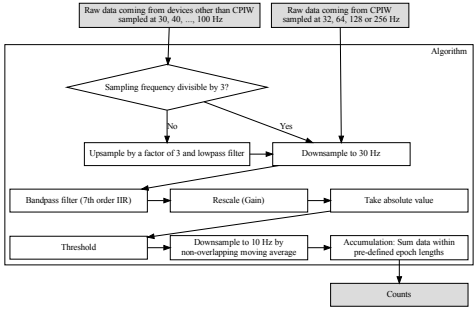
> ***New in version 1.3.0.***

offset : ***Timedelta or str, default is None***

An offset timedelta added to the origin.

Beware of filters and boundary effects

Example (Calculating ActiGraph counts [5])



- The various filters all have boundary effects
- In CentrePoint, a 25 second margin is used to get rid of those.

ActiGraph

Some operations are intrinsically not deterministic

Example (Auto-calibration is necessary for ENMO [3, 7])

- This requires finding periods of no movement.
- When data can come in out-of-order, this becomes non-deterministic.

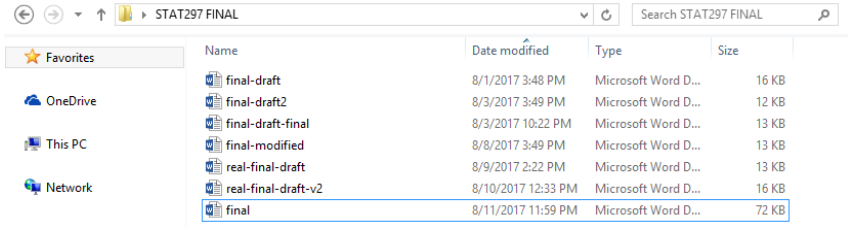
So what should we do?

- ☒ The design phase
- ☐ Implementation

Example (Let's code something up)

We all forget what we did

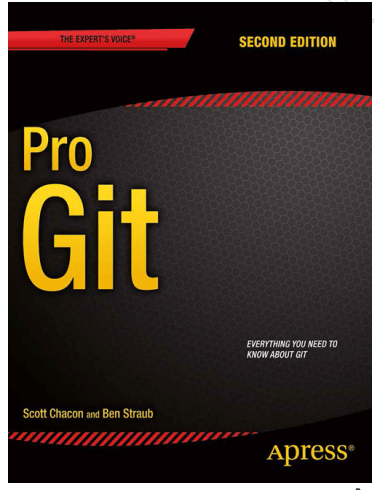
We need git



Largely a solved problem in software engineering

Make sure you master git

- Commit early and often
- Use and abuse branches
- Take care of commit messages
- Tags are your friends



Reduce manual steps as much as you can

snakemake

Regressions

Regression tests

It is crucial to make sure you are not breaking something as you code. There are so many moving pieces in a modern software stack!

Definition (Unit tests)

Unit testing, a.k.a. component or module testing, is a form of software testing by which isolated source code is tested to validate expected behavior

Definition (Approval tests)

You capture the output of a software, system, or program and then compare it with the previously approved version to quickly verify that the current output matches the expected output.

Unit tests

Function to test

```
def foo(x):
  if x>0:
    return x+1
```

The test

```
import unittest

class TestAddFunction(unittest.TestCase):
    def test_add_positive_numbers(self):
        self.assertEqual(add(1, 2), 3)

    def test_add_negative_numbers(self):
        self.assertEqual(add(-1, -2), -3)

    def test_add_positive_and_negative_numbers(self):
        self.assertEqual(add(1, -1), 0)

    def test_add_zeros(self):
        self.assertEqual(add(0, 0), 0)

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

Approval tests

The real function

```
def _transform(
        self,
        inputs: Dict[str, Dataset],
    ) -> Dataset:
    features = inputs["magnitude"].data
    labels = inputs["activity"].data

    peaks = features.groupby(pd.Grouper(
            freq=f"{UWF.WIN_SIZE}s",
            origin="start_day",
            closed="left",
        )).apply(
            lambda x: self._get_steps(x, labels)
    )

    freqs_from_peaks = sampling_rate * 2 / peaks
    steps = pd.DataFrame(
        data=freqs_from_peaks * UWF.WIN_SIZE, columns=["steps"]
    ).replace(np.inf, 0)

    return steps
```

The test

```
,subject,date,bout_id,bout_start,bout_e
0,10821,2020-10-13,0,2020-10-13 08:58:1
1,10821,2020-10-13,1,2020-10-13 09:05:2
2,10821,2020-10-13,2,2020-10-13 09:37:1
3,10821,2020-10-13,3,2020-10-13 09:37:5
4,10821,2020-10-13,4,2020-10-13 09:40:5
5,10821,2020-10-13,5,2020-10-13 09:48:5
6,10821,2020-10-13,6,2020-10-13 09:54:1
7,10821,2020-10-13,7,2020-10-13 09:55:4
8,10821,2020-10-13,8,2020-10-13 09:56:4
9,10821,2020-10-13,9,2020-10-13 10:48:5
10,10821,2020-10-13,10,2020-10-13 11:11
11,10821,2020-10-13,11,2020-10-13 11:14
12,10821,2020-10-13,12,2020-10-13 12:47
```

So what should we do?

- ☒ The design phase
- ☒ Implementation
- ☐ Third-party dependencies

Tracking dependencies I

There are many such tools for Python

Example (PIP)

```
python -m pip freeze [options]
```

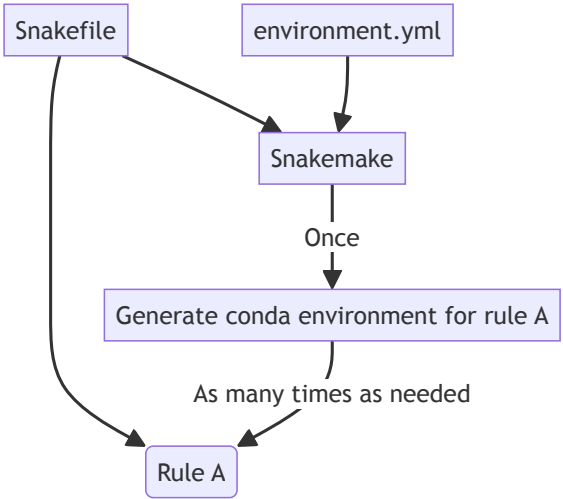
Example (Poetry)

```
poetry lock
```

Example (Conda environment [2])

```
conda list --export
```

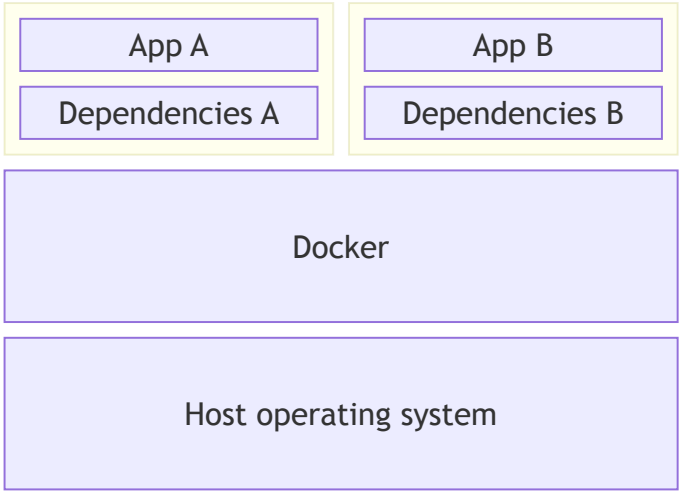
Managing the full pipeline using Snakemake [4]



So what should we do?

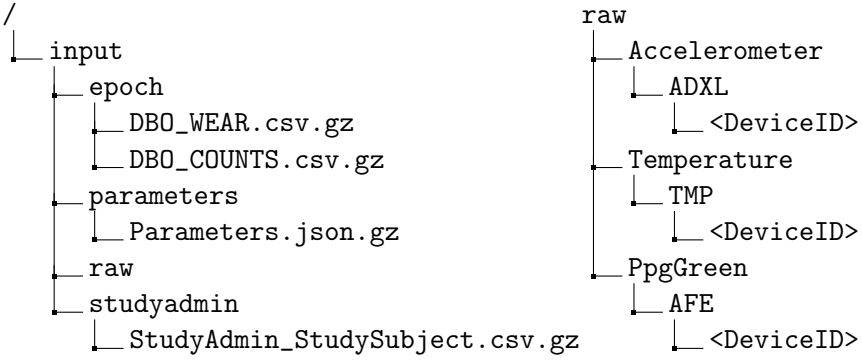
- ☒ The design phase
- ☒ Implementation
- ☒ Third-party dependencies
- ☐ The environment

Freezing everything using containers

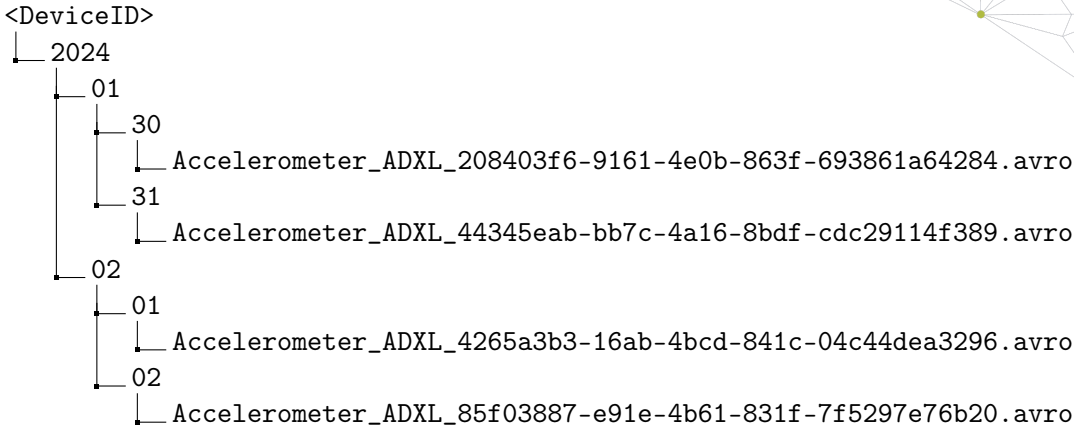


ActiGraph

CentrePoint Designer



Raw data


<DeviceID>
2024
01
30
Accelerometer_ADXL_208403f6-9161-4e0b-863f-693861a64284.avro
31
Accelerometer_ADXL_44345eab-bb7c-4a16-8bdf-cdc29114f389.avro
02
01
Accelerometer_ADXL_4265a3b3-16ab-4bcd-841c-04c44dea3296.avro
02
Accelerometer_ADXL_85f03887-e91e-4b61-831f-7f5297e76b20.avro

Source code

https://github.com/actigraph/icampam_2024_workshop

Bibliography I

[1] Lukas Adamowicz et al. "SciKit Digital Health: Python Package for Streamlined Wearable Inertial Sensor Data Processing". In: *JMIR Mhealth Uhealth* 10.4 (Apr. 21, 2022), e36762. ISSN: 2291-5222. DOI: 10.2196/36762. URL: https://mhealth.jmir.org/2022/4/e36762 (visited on 11/08/2022).

[2] Anaconda. ***Anaconda Software Distribution***. Version 2-2.4.0. Nov. 2016. URL: https://anaconda.com.

[3] Jairo H. Migueles et al. "GGIR: A Research Community–Driven Open Source R Package for Generating Physical Activity and Sleep Outcomes From Multi-Day Raw Accelerometer Data". In: *Journal for the Measurement of Physical Behaviour* 2.3 (Sept. 1, 2019), pp. 188–196. ISSN: 2575-6605, 2575-6613. DOI: 10.1123/jmpb.2018-0063. URL: https://journals.humankinetics.com/view/journals/jmpb/2/3/article-p188.xml (visited on 09/19/2023).

[4] Felix Mölder et al. ***Sustainable Data Analysis with Snakemake***. Apr. 19, 2021. DOI: 10.12688/f1000research.29032.2. F1000Research: 10:33. URL: https://f1000research.com/articles/10-33 (visited on 06/11/2024). preprint.

[5] Ali Neishabouri et al. "Quantification of Acceleration as Activity Counts in ActiGraph Wearable". In: *Sci Rep* 12.1 (July 13, 2022), p. 11958. ISSN: 2045-2322. DOI: 10.1038/s41598-022-16003-x. URL: https://www.nature.com/articles/s41598-022-16003-x (visited on 06/11/2024).

[6] The pandas development team. ***Pandas-Dev/Pandas: Pandas***. Version v2.2.2. Zenodo, Apr. 10, 2024. DOI: 10.5281/zenodo.10957263. URL: https://zenodo.org/records/10957263 (visited on 06/10/2024).

[7] Vincent T. van Hees et al. "Autocalibration of Accelerometer Data for Free-Living Physical Activity Assessment Using Local Gravity and Temperature: An Evaluation on Four Continents". In: *Journal of Applied Physiology* 117.7 (Oct. 2014), pp. 738–744. ISSN: 8750-7587. DOI: 10.1152/japplphysiol.00421.2014. URL: https://journals.physiology.org/doi/full/10.1152/japplphysiol.00421.2014 (visited on 08/25/2022).

[8] Vincent T. van Hees et al. "Estimating Sleep Parameters Using an Accelerometer without Sleep Diary". In: *Sci Rep* 8.1 (1 Aug. 28, 2018), p. 12975. ISSN: 2045-2322. DOI: 10.1038/s41598-018-31266-z. URL: https://www.nature.com/articles/s41598-018-31266-z (visited on 08/01/2022).