

# How to open your code when part of it cannot be shared?

---

Xavier Olive

3rd/4th April 2025

Over the past 30-ish years...

Over the past 30-ish years...

- **Crowdsourcing**: it is possible to collaborate and enrich large-scale databases: Wikipedia, OpenStreetMap, etc.
- Research and industry leverage capabilities to harness **larger amounts of data**
- Shift from model-informed science to data-informed science
- In academic papers:
  - you expect proofs for theorems
  - you expect description of frameworks and models
  - for data-based science, you expect ...?

## Background question

Academia: culture of collaboration and transparency, so...

Why don't you share your code?

Academia: culture of collaboration and transparency, so...

Why don't you share your code?

- I want to share my code, **but...**

Academia: culture of collaboration and transparency, so...

### Why don't you share your code?

- I want to share my code, **but...**
- I want to share my code, **but...** it's not clean enough
- I want to share my code, **but...** the instances for the problem are not open anyway
- I want to share my code, **but...** part of it is not open
- I want to share my code, **but...** my employer does not agree

## Aircraft Fleet Health Monitoring with Anomaly Detection Techniques

Luis Basora<sup>1,\*,\ddagger,\textcircled{ID}}</sup>, Paloma Bry<sup>1,\ddagger,\textcircled{ID}}</sup>, Xavier Olive<sup>1,\ddagger,\textcircled{ID}}</sup> and Floris Freeman<sup>2</sup>

<sup>1</sup> ONERA DTIS, Université de Toulouse, 31055 Toulouse, France; paloma.bry@protonmail.com (P.B.); xavier.olive@onera.fr (X.O.)

<sup>2</sup> KLM Royal Dutch Airlines, Postbus 7700, 1117 ZL Schiphol, The Netherlands; Floris.Freeman@klm.com

\* Correspondence: luis.basora@onera.fr

\ddagger Current address: 2 Avenue Édouard Belin, CEDEX 4, 31055 Toulouse, France.

\ddagger These authors contributed equally to this work.

**Abstract:** Predictive maintenance has received considerable attention in the aviation industry where costs, system availability and reliability are major concerns. In spite of recent advances, effective health monitoring and prognostics for the scheduling of condition-based maintenance operations is still very challenging. The increasing availability of maintenance and operational data along with recent progress made in machine learning has boosted the development of data-driven prognostics and health management (PHM) models. In this paper, we describe the data workflow in place at an airline for the maintenance of an aircraft system and highlight the difficulties related to a proper labelling of the health status of such systems, resulting in a poor suitability of supervised learning techniques. We focus on investigating the feasibility and the potential of semi-supervised anomaly detection methods for the health monitoring of a real aircraft system. Proposed methods are evaluated on large volumes of real sensor data from a cooling unit system on a modern wide body aircraft from a major European airline. For the sake of confidentiality, data has been anonymized and only few technical and operational details about the system had been made available. We trained several deep neural network autoencoder architectures on nominal data and used the anomaly scores to calculate a health indicator. Results suggest that high anomaly scores are correlated with identified failures in



**Citation:** Basora, L.; Bry, P.; Olive, X.; Freeman, F. Aircraft Fleet Health Monitoring with Anomaly Detection

# Challenges

- *“...but it’s not clean enough”*:  
Poorly written code is better than non-existent code, e.g. ... (you name it)
- *“...but my employer does not agree”*:  
Let’s look into the reasons first? Maybe there are good ones... (maybe...)

My focus here:

- instances are not open;
- part of the code is not open;
- the code is open, the model is not (BADA)



# Limitations to openness

Practical limitations:

- NDA
- Proprietary concerns
- Legal concerns
- Licence concerns

What do you risk?

- Sometimes, companies do not seem to care (*for now*)
- NDA breach, disciplinary sanctions
- Legal actions (EUROCONTROL)

## If everything is still closed

Why do you want to publish in the first place?

(Doesn't it “kind of” work on other public instances? Then explain why it really works better on your problem)

**Abstract:**

*[...] Due to stringent confidentiality protocols, the underlying system remains shrouded in mystery. Nevertheless, we introduce a cutting-edge LSTM variant designed to predict its elusive behaviour—with an accuracy of 68% that, we dare say, redefines the benchmark for secretive systems. The scientific contribution, though cloaked in anonymity, is submitted with the hope that its merit justifies publication—and perhaps even funds a well-deserved three-week post-conference sojourn in New Zealand.*

## If everything is still closed

Why do you want to publish in the first place?

(Doesn't it “kind of” work on other public instances? Then explain why it really works better on your problem)

**Abstract:**

*[...] Due to stringent confidentiality protocols, the underlying system remains shrouded in mystery. Nevertheless, we introduce a cutting-edge LSTM variant designed to predict its elusive behaviour—with an accuracy of 68% that, we dare say, redefines the benchmark for secretive systems. The scientific contribution, though cloaked in anonymity, is submitted with the hope that its merit justifies publication—and perhaps even funds a well-deserved three-week post-conference sojourn in New Zealand.*

*Thank ChatGPT for the abstract*

## Main question (focus of this talk)

How to open software when not everything can be shared?

Can such software still be useful to other people?

Examples are many:

- Data is not strictly open, but the code to access it is  
<https://github.com/open-aviation/pyopensky> (LGPL-3.0)
- The model is closed, but the implementation is open  
<https://github.com/eurocontrol-bada/pybada> (EURL-1.2)
- Data used for training is closed, but the model seems open  
<https://github.com/DGAC/Acropole> (AGPL-3.0)

Let's go through several of them

1. Use configuration files:

- *"Data is not open, but the code to access it is"*
- Store tokens, passwords, credentials, specific files, in a file outside the repository

2. Use open alternatives:

- <https://openap.dev/> instead of pyBADA
- OpenSky Network datasets instead of FlightRadar24
- Open source datasets instead of proprietary data

3. Use stubs

- "Here is a very naive way to do things, it doesn't work, but you can implement your own"

## Pattern #1 Configuration files

- GitHub is a major source of password leaks (even with OpenSky Network)
- **NEVER put any password in your code**, even if you don't plan to commit it
- Start with a configuration file, support also environment variables (better for CI):

```
[global]
```

```
username = my_username
```

```
password = my_password
```

- Also support environment variables (better for CI)

```
import os
```

```
os.getenv('MY_PASSWORD')
```

## Pattern #1 Sample code

```
# config.py
# >>> from config import username, password
import configparser

config = configparser.ConfigParser()
config.read('config.toml') # use appdirs to place the file in a standard location

def get_config(name: None | str, category: None | str, env: None | str) -> Any:
    if (value := config.get(category, name, fallback=None)) is not None:
        return value
    if (value := os.getenv(env)) is not None:
        return value
    raise ValueError(f"Cannot find {name} in {category} or ${env}")

def __getattr__(name: str) -> Any:
    return get_config(name, 'global', f'MY_{name.upper()}')
```

## Pattern #2 Use alternatives

Learn about the **Visitor** design pattern (Gang of Four)

```
class BADA:
    def performance(self, df: pd.DataFrame) -> pd.DataFrame:
        ...

class OpenAP:
    def performance(self, df: pd.DataFrame) -> pd.DataFrame:
        ...

def performance(df: pd.DataFrame, model: ?) -> pd.DataFrame:
    return model.performance(df)
```



## Pattern #2 Use alternatives (Protocol)

In Python, you can use the **Protocol** design pattern (PEP 647)

```
from typing import Protocol

class PerformanceModel(Protocol):
    def performance(self, df: pd.DataFrame) -> pd.DataFrame:
        ...

class OpenAP: # or BADA
    def performance(self, df: pd.DataFrame) -> pd.DataFrame:
        pass

def performance(df: pd.DataFrame, model: PerformanceModel) -> pd.DataFrame:
    return model.performance(df)
```

## Pattern #3 Use stubs

Sometimes, there is no such thing as an open alternative

A stub is a minimal implementation that provides predetermined responses for method calls, essentially simulating functionality without any additional logic.

```
class EurocontrolDB:
    # [...]

    def citypair(self, callsign: str) -> tuple[Airport, Airport]:
        resp = self.ectl_service.query(callsign).get('response')
        if resp is None:
            raise ValueError(f"Cannot find {callsign}")
        return Airport(resp['departure']), Airport(resp['arrival'])

class StubDB:
    def citypair(self, callsign: str) -> tuple[Airport, Airport]:
        return Airport('Trifouillis les Oies'), Airport('Hintertupfingen')
```

- Open source software is a powerful tool for research
- Sharing code is challenging possible, even when parts are not open

- Open source software is a powerful tool for research
- Sharing code is challenging possible, even when parts are not open
- Use configuration files to manage sensitive data
- Explore open alternatives or stubs to simulate unavailable components

- Open source software is a powerful tool for research
- Sharing code is challenging possible, even when parts are not open
- Use configuration files to manage sensitive data
- Explore open alternatives or stubs to simulate unavailable components
- If everything is closed, consider the value of your work and its potential impact

## Questions · Discussion

---