MODUS CREATE

On the structure and reproducibility of Python packages - data crunch

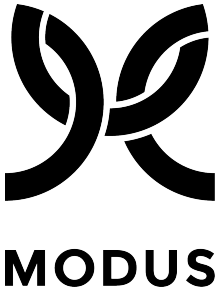
Maria Knorps, Zhihan Zhang

23-09-2024

Zen of Python: excerpts

There should be one
– and preferably only one –
obvious way to do it.

Although that way may not be obvious at first unless you're Dutch.



TWEAG

A Modus Create Company

[tweag/FawltyDeps](https://github.com/tweag/FawltyDeps)

Dependency checker:
finds **undeclared** and **unused** dependencies

Maria Knorps

Johan Herland

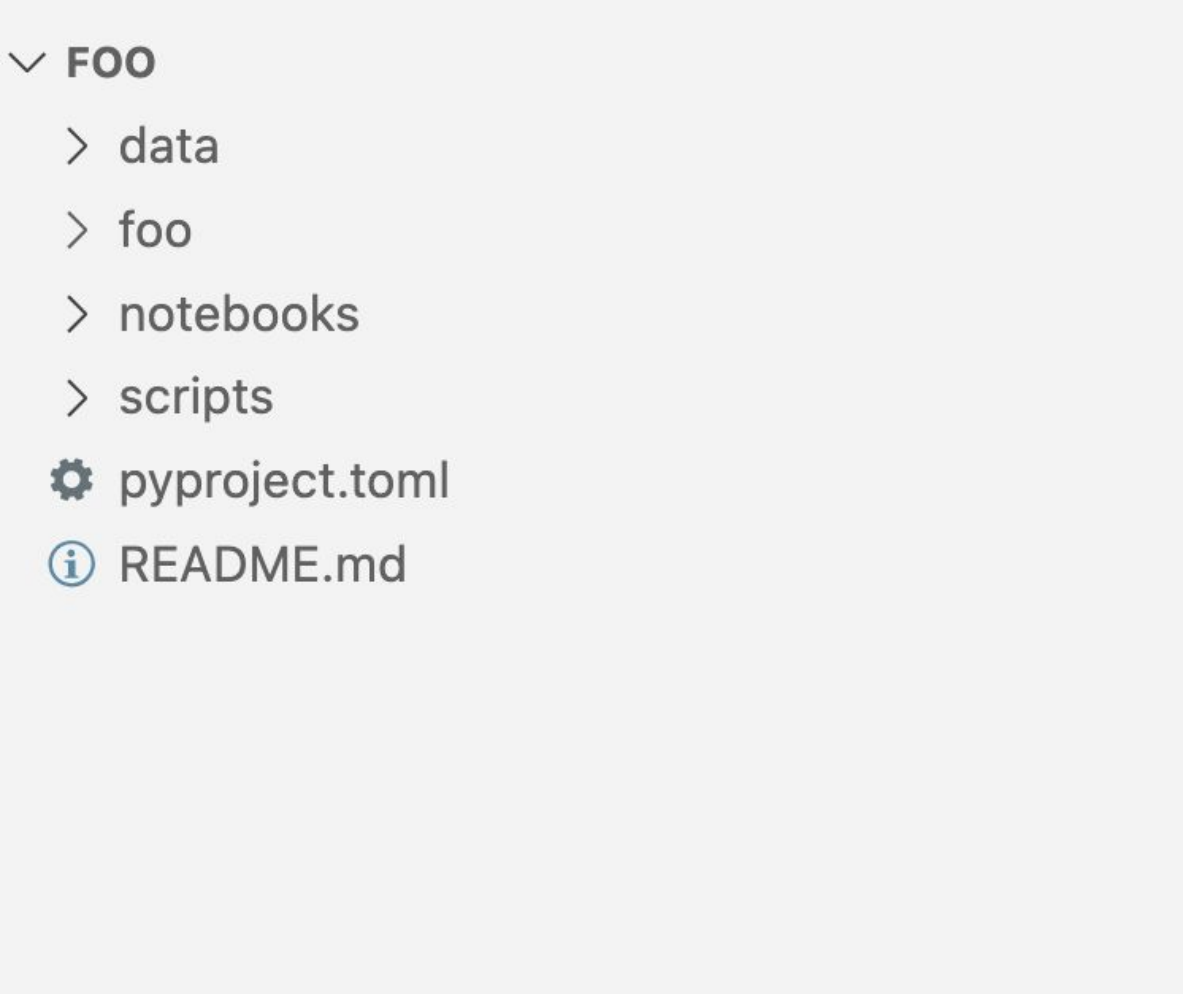
Zhihan Zhang

Structure

FOO

- analysis.ipynb
- collect.py
- data.zip
- file1.csv
- file2.csv
- new_data1.xlsx
- new_data2.xlsx
- new_data3.json
- new_data4.jsonl
- prepare.py
- utils.py

😨


FOO
data
foo
notebooks
scripts
pyproject.toml
README.md

The structure: code hygiene

Project `foo` structure

same name

```
.
├── README.md
├── foo/
│   ├── __init__.py
│   ├── run.py
│   └── utils.py
├── pyproject.toml
└── tests/
```

Project `foo` structure

same name

```
.
├── README.md
├── foo/
│   ├── __init__.py
│   ├── run.py
│   └── utils.py
├── pyproject.toml
└── tests/
```

src

```
.
├── README.md
├── src/
│   ├── __init__.py
│   ├── run.py
│   └── utils.py
├── pyproject.toml
└── tests/
```

Project `foo` structure

same name

```
.
├── README.md
├── foo/
│   ├── __init__.py
│   ├── run.py
│   └── utils.py
├── pyproject.toml
└── tests/
```

src

```
.
├── README.md
├── src/
│   ├── __init__.py
│   ├── run.py
│   └── utils.py
├── pyproject.toml
└── tests/
```

root directory

```
.
├── README.md
├── run.py
├── utils.py
├── pyproject.toml
└── test_utils.py
```

Project `foo` structure

same name

```
.
├── README.md
├── foo/
│   ├── __init__.py
│   ├── run.py
│   └── utils.py
├── pyproject.toml
└── tests/
```

src

```
.
├── README.md
├── src/
│   ├── __init__.py
│   ├── run.py
│   └── utils.py
├── pyproject.toml
└── tests/
```

root directory

```
.
├── README.md
├── run.py
├── utils.py
├── pyproject.toml
└── test_utils.py
```

different name

```
.
├── README.md
├── bar/
│   ├── __init__.py
│   ├── run.py
│   └── utils.py
├── pyproject.toml
└── tests/
```

Project `foo` structure

same name

```
.
├── README.md
├── foo/
│   ├── __init__.py
│   ├── run.py
│   └── utils.py
├── pyproject.toml
└── tests/
```

src

```
.
├── README.md
├── src/
│   ├── __init__.py
│   ├── run.py
│   └── utils.py
├── pyproject.toml
└── tests/
```

root directory

```
.
├── README.md
├── run.py
├── utils.py
├── pyproject.toml
└── test_utils.py
```

different name

```
.
├── README.md
├── bar/
│   ├── __init__.py
│   ├── run.py
│   └── utils.py
├── pyproject.toml
└── tests/
```

Reproducibility

Levels of reproducibility

Level -1: not reproducible

Level 0: reproducible only by you, today

Level 1: reproducible by others with guidance

Level 2: reproducible today, by anyone with internet access

Level 3: reproducible indefinitely, by anyone with internet access

...

https://www.anaconda.com/blog/8-levels-of-reproducibility

Levels of reproducibility

Level -1: not reproducible

Level 0: reproducible only by you, today

Level 1: reproducible by others with guidance

Level 2: reproducible today, by anyone with internet access

Level 3: reproducible indefinitely, by anyone with internet access

...

https://www.anaconda.com/blog/8-levels-of-reproducibility

Levels of reproducibility

Level -1: not reproducible

Level 0: reproducible only by you, today

Level 1: reproducible by others with guidance

Level 2: reproducible today, by anyone with internet access

Level 3: reproducible indefinitely, by anyone with internet access

...

https://www.anaconda.com/blog/8-levels-of-reproducibility

Levels of reproducibility

Level -1: not reproducible

Level 0: reproducible only by you, today

Level 1: reproducible by others with guidance

Level 2: reproducible today, by anyone with internet access

Level 3: reproducible indefinitely, by anyone with internet access

...

https://www.anaconda.com/blog/8-levels-of-reproducibility

Levels of reproducibility

Level -1: not reproducible

Level 0: reproducible only by you, today

Level 1: reproducible by others with guidance

Level 2: reproducible today, by anyone with internet access

Level 3: reproducible indefinitely, by anyone with internet access

...

https://www.anaconda.com/blog/8-levels-of-reproducibility

Code reproducibility

```
[1]: import time

     def my_costly_operation():
         time.sleep(60)

         import pandas as pd
         foo = pd.DataFrame({"foo": 1})

         return foo
```

```
[2]: my_costly_operation()
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[2], line 1
----> 1 my_costly_operation()

Cell In[1], line 6, in my_costly_operation()
      3 def my_costly_operation():
      4     time.sleep(60)
----> 6     import pandas as pd
      7     foo = pd.DataFrame({"foo": 1})
      9     return foo

ModuleNotFoundError: No module named 'pandas'
```

Dependencies declaration

requirements.txt

```
pandas
click
```

Dependencies declaration

requirements.txt

```
pandas
click
```

setup.py

```
from setuptools import setup

setup(
 name="MyLib",
 install_requires=[
   "pandas",
   "click>=1.2"
],
 extras_require={
   "http": ["requests"],
   "chinese": ["jieba"]
})
```

Dependencies declaration

requirements.txt

```
pandas
click
```

setup.py

```
from setuptools import setup

setup(
 name="MyLib",
 install_requires=[
   "pandas",
   "click>=1.2"
],
 extras_require={
   "http": ["requests"],
   "chinese": ["jieba"]
})
```

pyproject.toml

```
[project]
name = "mixed"

dependencies = ["pandas", "click",
"black"]

[tool.black]
target-version = ["py37"]

[build-system]
requires = ["setuptools >= 61.0"]
build-backend = "setuptools.build_meta"
```

Dependencies declaration

requirements.txt

```
pandas
click
```

setup.py

```
from setuptools import setup

setup(
 name="MyLib",
 install_requires=[
   "pandas",
   "click>=1.2"
],
 extras_require={
   "http": ["requests"],
   "chinese": ["jieba"]
})
```

pyproject.toml

```
[project]
name = "mixed"

dependencies = ["numpy", "setuptools",
"black"]

[tool.black]
target-version = ["py37"]

[build-system]
requires = ["setuptools >= 61.0"]
build-backend = "setuptools.build_meta"
```

The Experiment

The Experiment: data

Biomedical data

- From scientific biomedical articles

PyPI data

- Most downloaded PyPI packages

Vs

The Experiment: data

Biomedical data

- From scientific biomedical articles

repository	article_subject
SIMEXP/cbrain-plugins-psom	Technical Note
CyTargetLinker/linksetCreator	Software Tool Article
multi-template-matching/MultiTemplateMatching-...	Software
DiODeProject/MuMoT	Research Article
dib-lab/SSUsearch	Methods
severin-lemaignan/pinsoro-kinematics-study	Robotics and AI
yuifu/millefy	Software
misun6312/PBupsModel.jl	Article
goptavares/aDDM-Toolbox	Research Article
kharyuk/chemfin-plasp	Article

PyPI data

- Most downloaded PyPI packages

	project	2023 downloads [M]
1	boto3	8409.0
2	botocore	3935.0
3	setuptools	3426.0
4	charset-normalizer	3331.0
5	typing-extensions	3219.0
6	idna	2997.0
7	s3transfer	2881.0
8	wheel	2497.0
9	google-api-core	2348.0
10	cryptography	2245.0

What metrics we collected?

"Yeah, I keep a clean desk. Now all the mess is in the computer!"

What metrics we collected?

Where they put source code?

- Same name
- Src
- Root directory .
- Different name

"Yeah, I keep a clean desk. Now all the mess is in the computer!"

What metrics we collected?

Where they put source code?

- Same name
- Src
- Root directory .
- Different name

"Yeah, I keep a clean desk. Now all the mess is in the computer!"

How they declare dependencies?

- requirements.txt
- pyproject.toml (poetry)
- setup.py
- Mix of the above
- Or... Not at all!

What metrics we collected?

Where they put source code?

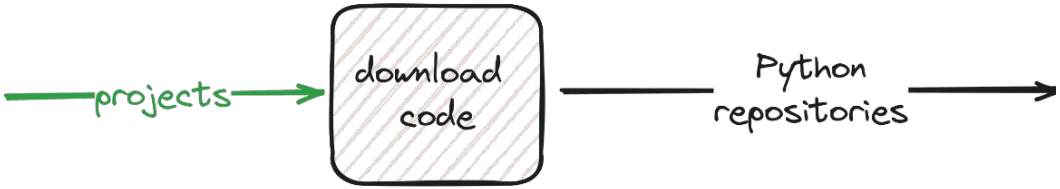
- Same name
- Src
- Root directory .
- Different name

"Yeah, I keep a clean desk. Now all the mess is in the computer!"

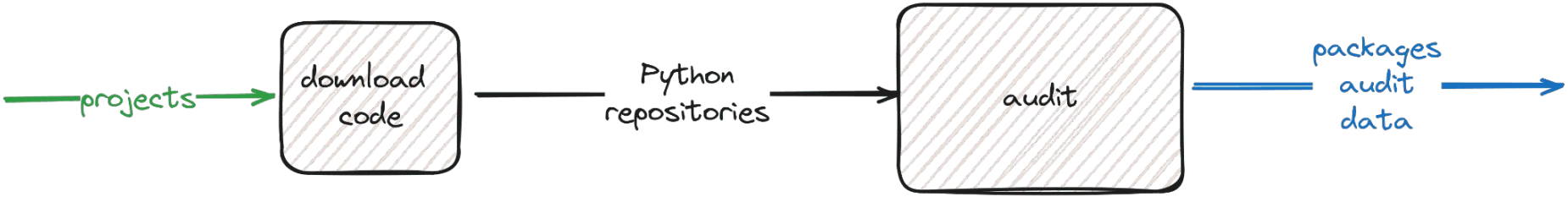
How they declare dependencies?

- requirements.txt
- pyproject.toml (poetry)
- setup.py
- Mix of the above
- Or... Not at all!

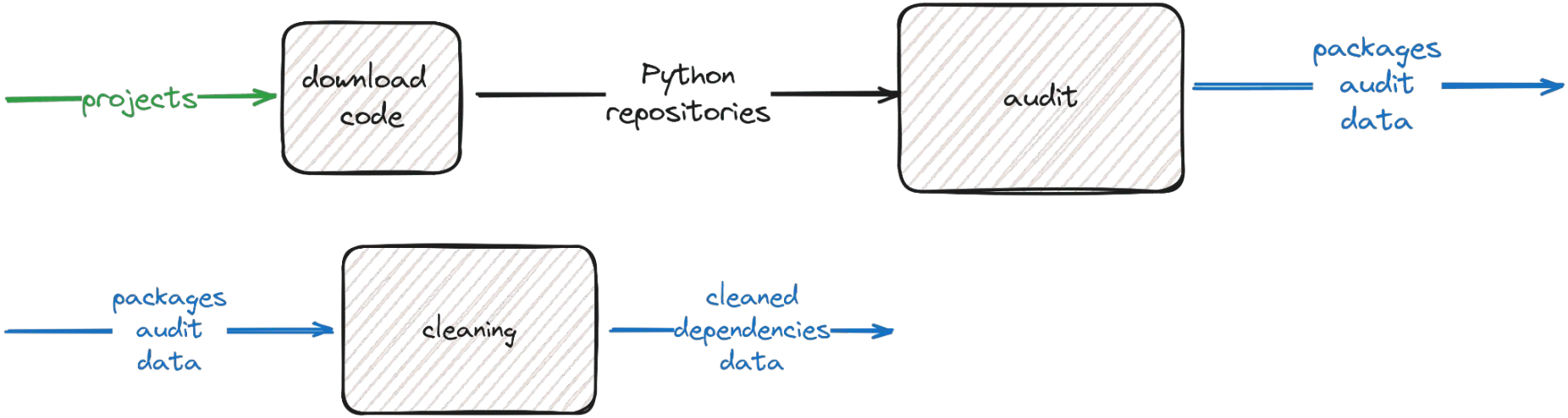
How to collect the data?



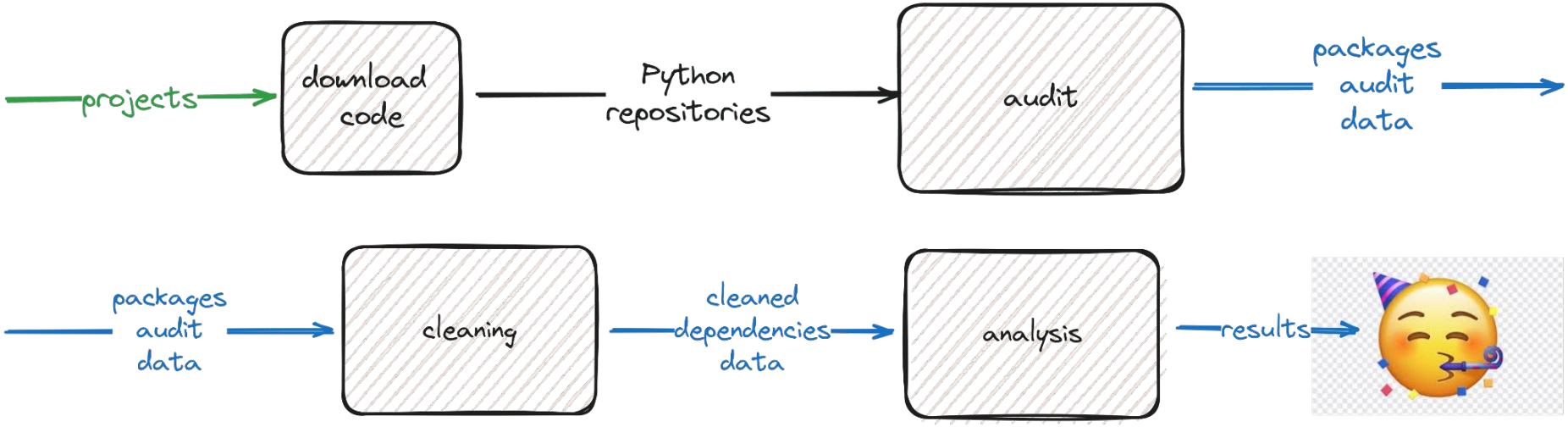
How to collect the data?



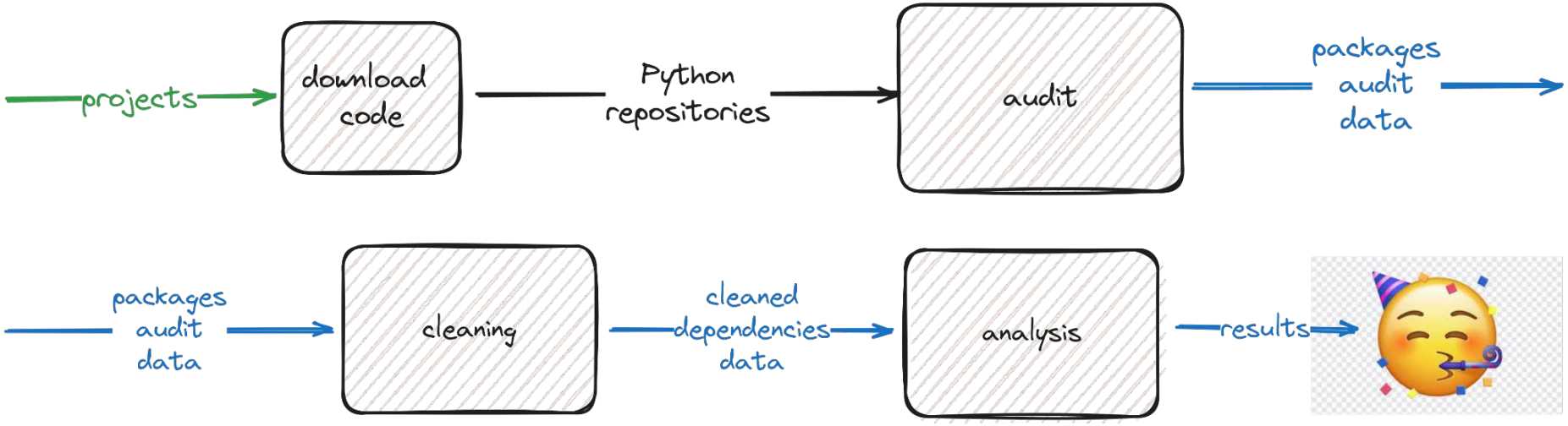
How to collect the data?



How to collect the data?



How to collect the data?



- 1,260 biomedical projects
- 1,118 PyPI packages

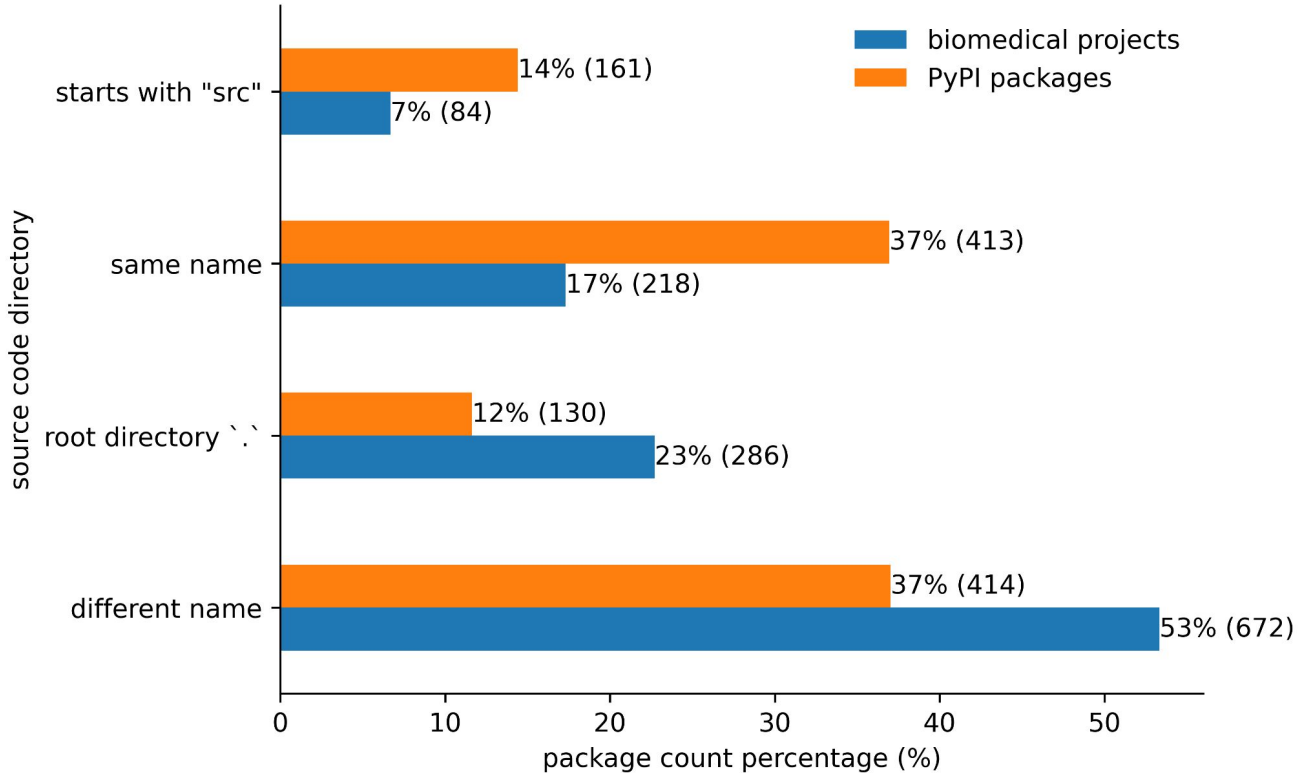
Is our experiment reproducible?

YES!

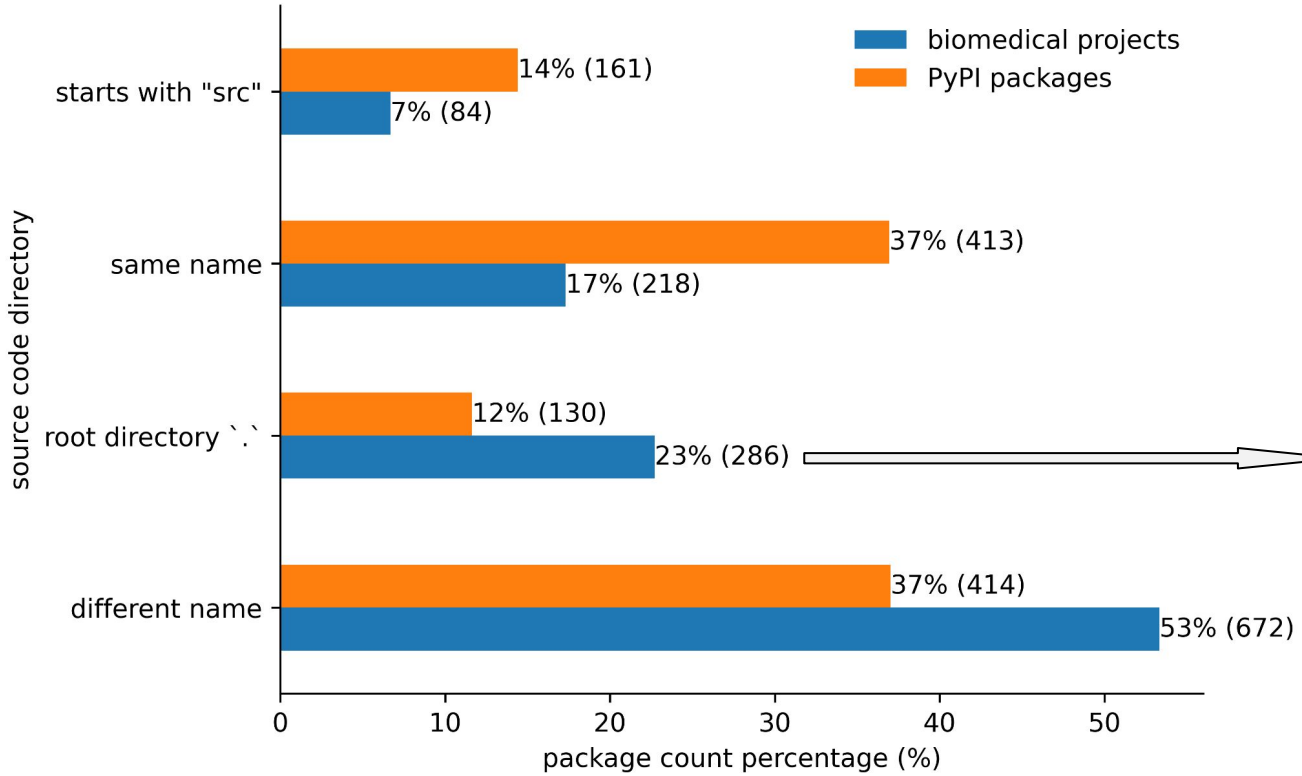
- Separate branch of FawltyDeps
- Separate analysis repo: FawltyDeps-analysis

Results

How they structure code?

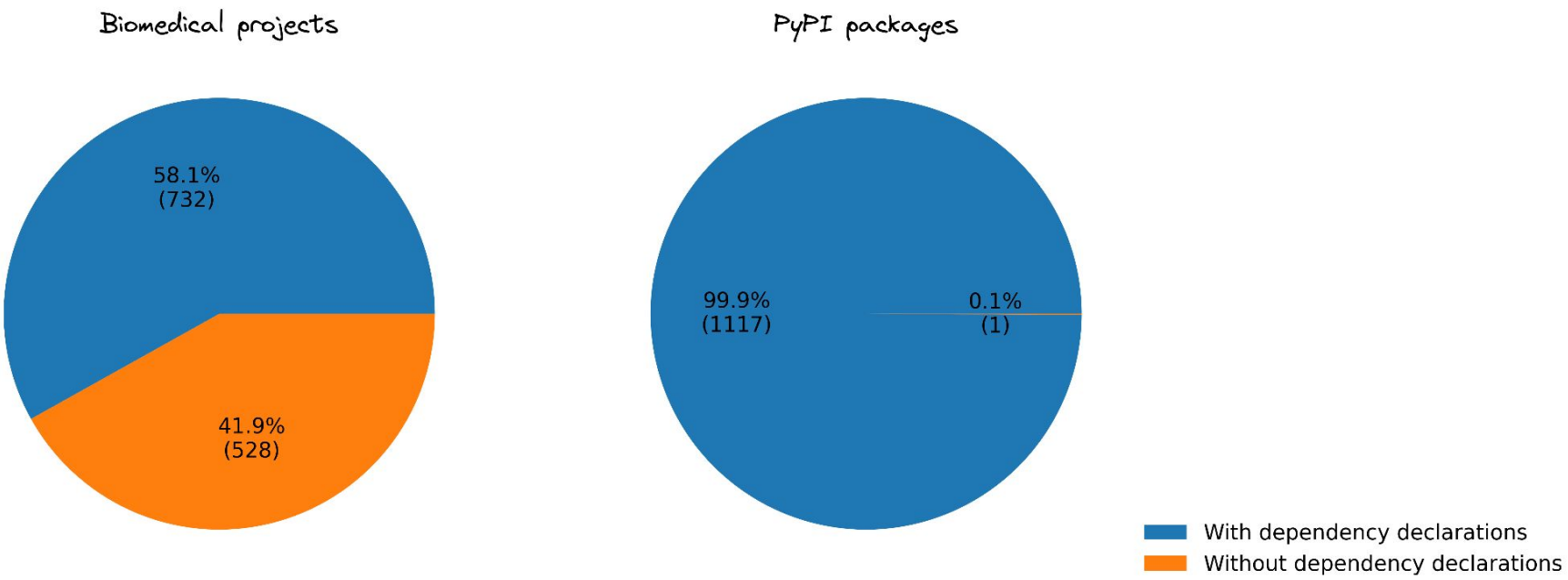


How they structure code?



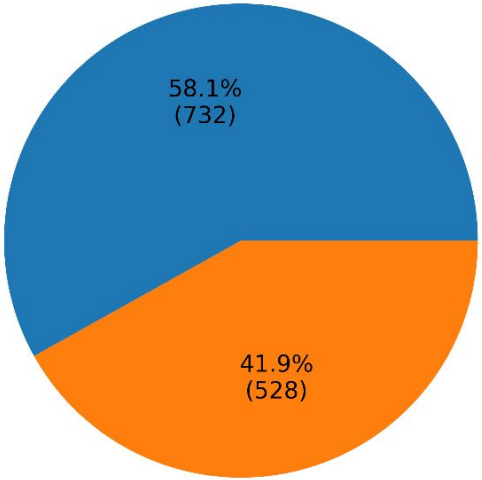
a lot of biomedical projects might be a loose collection of notebooks/scripts

Who declares dependencies?

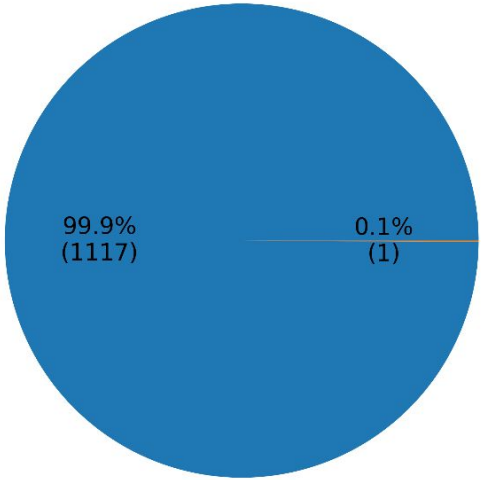


Who declares dependencies?

Biomedical projects



PyPI packages

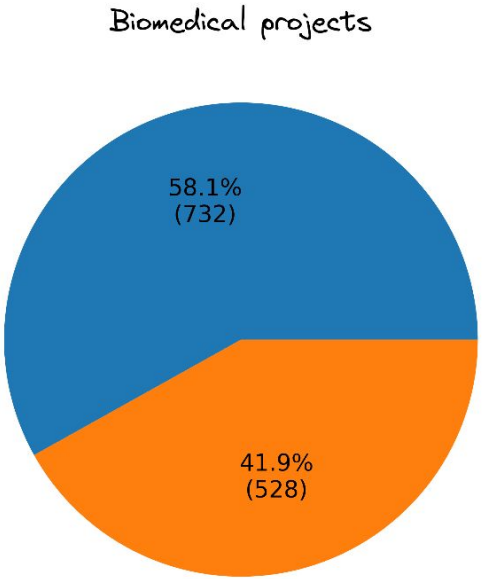


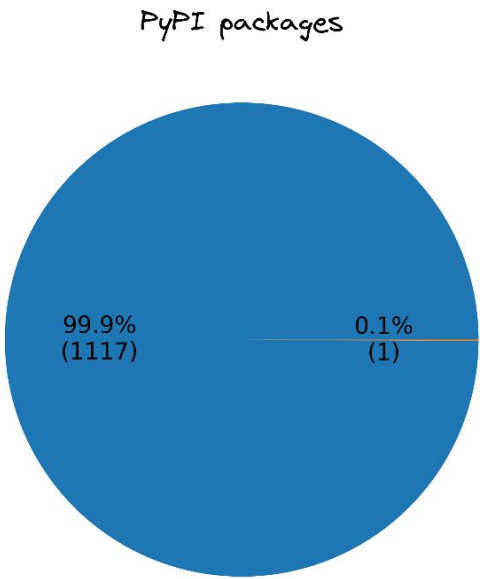
setup.py

```python
from setuptools import setup

setup(
 name="MyLib",
 install_requires=[
   "pandas",
   "click>=1.2"
],
 extras_require={
   "http": ["requests"],
   "chinese": ["jieba"]
}
 packages=["mylib"]
)
```

With dependency declarations
Without dependency declarations

Who declares dependencies?





setup.py

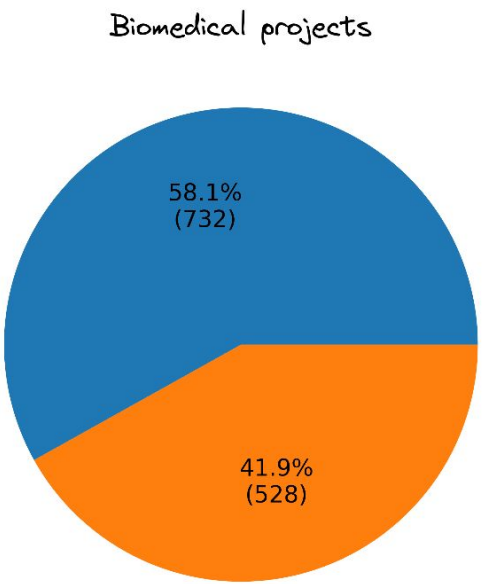
```
from setuptools import setup

setup(
 name="MyLib",
 install_requires=[
   "pandas",
   "click>=1.2"
],
 extras_require={
   "http": ["requests"],
   "chinese": ["jieba"]
}
 packages=["mylib"]
)
```

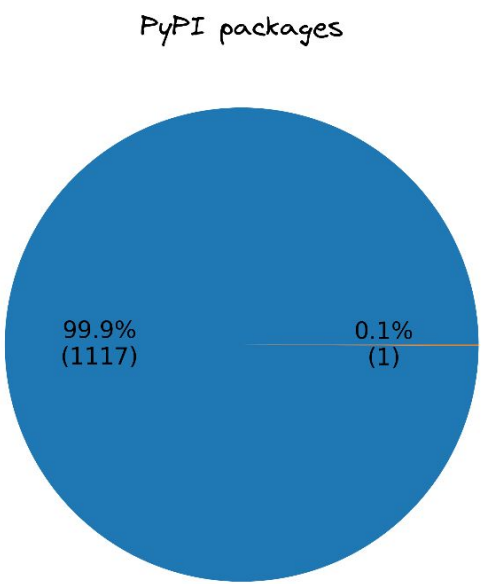
With dependency declarations
Without dependency declarations

Who declares dependencies?

setup.py

```
from setuptools import setup

setup(
 name="MyLib",
 packages=["mylib"]
)
```



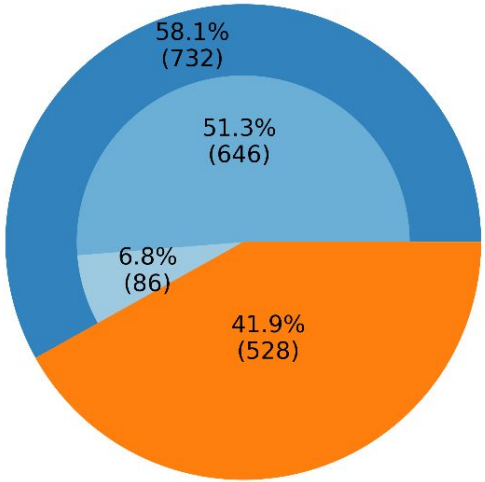


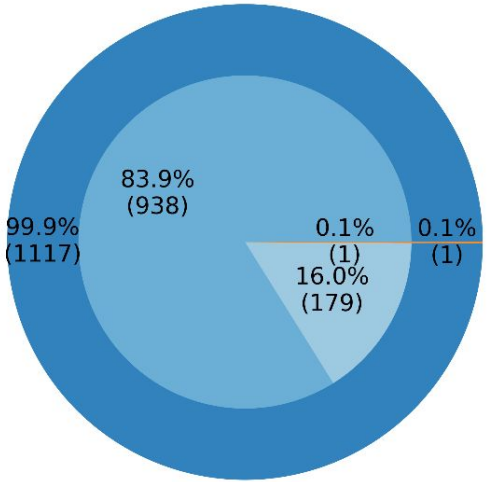
With dependency declarations
Without dependency declarations

Who declares dependencies?

setup.py

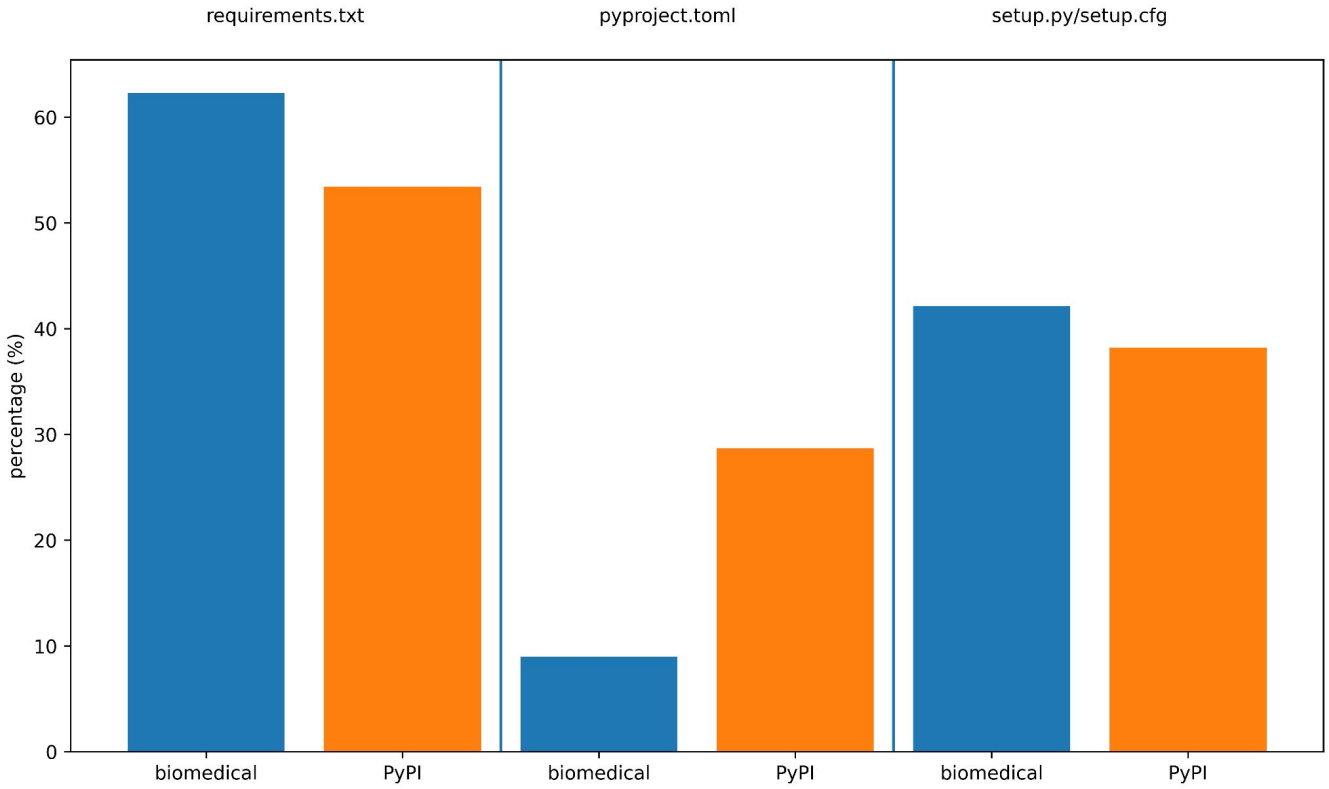
```
from setuptools import setup

setup(
 name="MyLib",
 packages=["mylib"]
)
```





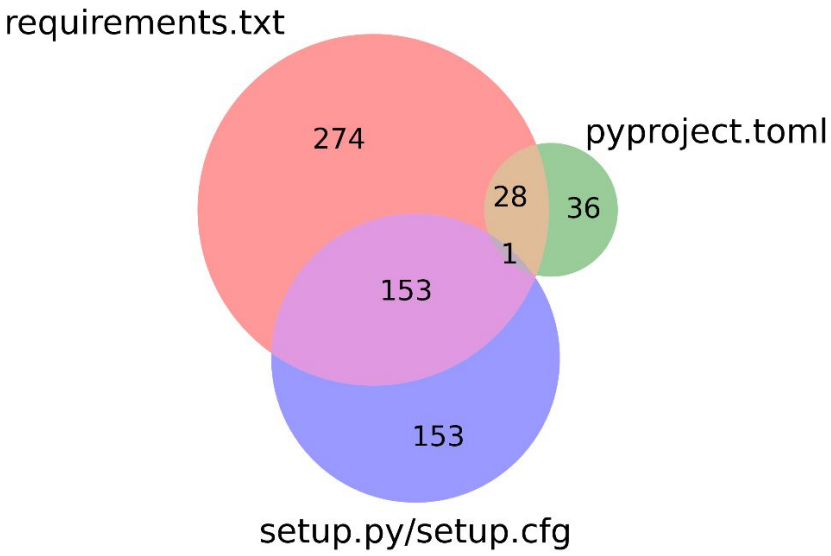
- With dependency declaration files
- Without dependency declaration files
- With dependency declared in the files
- Without dependency declared in the files

How they declare dependencies?

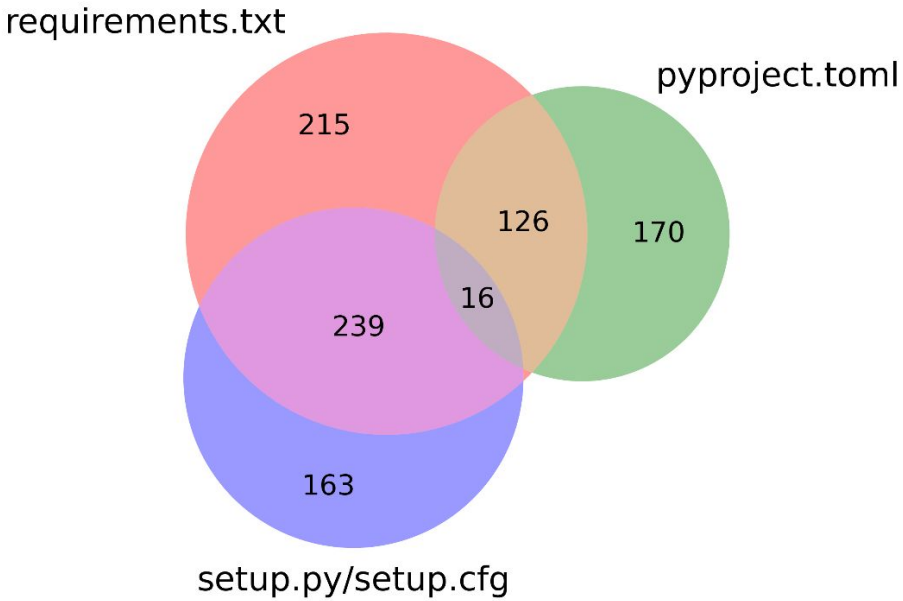


How they declare dependencies?

Biomedical projects



PyPI packages



Why?

Why?

Biomedical projects

- They are not necessarily importable

Why?

Biomedical projects

- They are not necessarily importable

PyPI packages

- Written to be *reproducible*

Why?

Biomedical projects

- They are not necessarily importable
- Follow the simple option: requirements.txt

PyPI packages

- Written to be *reproducible*

Why?

Biomedical projects

- They are not necessarily importable
- Follow the simple option: requirements.txt

PyPI packages

- Written to be *reproducible*
- Follow the newest trends - pyproject.toml

Conclusion

	Top PyPi packages	Biomedical packages
Structure	👌	🥲
Dependency declaration	👌	😭

You can experiment with Python metadata to get a better understanding of the ecosystem.

Next Steps

For us

Explore how well are the dependencies declared.

For you

Adopt the structure per project.

Use configuration tools like poetry or uv.

Declare your dependencies. Check it with FawltyDeps.

Check www.pyopensci.org

Crunch the data yourself!

Takeaways

```
Now is better than never.
Although never is often better than *right* now.
```

FawltyDeps:
experiment branch

Analysis repository:
FawltyDeps-analysis

Previous PyData Talk:
Introduce FawltyDeps

Experiment blog post

MODUS CREATE

THANK YOU!

Let’s make something great together.