EUROPYTHON 17-23 July 2023
PRAGUE & REMOTE

FROM JUPYTER NOTEBOOKS TO A PYTHON PACKAGE:

THE BEST OF BOTH WORLDS

Sin-seok SEO, Safran Tech, Safran SA

OUTLINE

1. Introduction
   - ✓ Jupyter Notebook
   - ✓ IDE (Integrated Dev. Env.)

2. Jupyter Notebook Data Science Workflow
   - ✓ Data loading
   - ✓ Preprocessing
   - ✓ Exploratory Data Analysis (EDA)
   - ✓ Prediction

3. To (Your Own) Python Package
   - ✓ What is a Python package?
   - ✓ How to create a (minimal) package
   - ✓ How to import and use
   - ✓ Live refactoring examples

4. Wrap-up / Some Tips

BEFORE WE BEGIN

- All the materials are available here
  - ✓ https://github.com/sesise0307/europython2023-package

- How many of you have experiences with Jupyter Notebooks?

- How many of you have experiences with IDEs, such as VS Code or PyCharm?

- How many of you have experiences with both?

INTRODUCTION

➢ Jupyter Notebook

- ✓ Web-based interactive application
- ✓ Provides ideal workflows for data science, scientific computing and machine learning
- ✓ Pros: REPL (Read-Eval-Print-Loop), interactivity, integration of code / output / documentation, visualization, rapid prototyping, result sharing, etc.
- ✓ Cons: lacks of debugging, code sharing, refactoring, version control, advanced editing, etc.

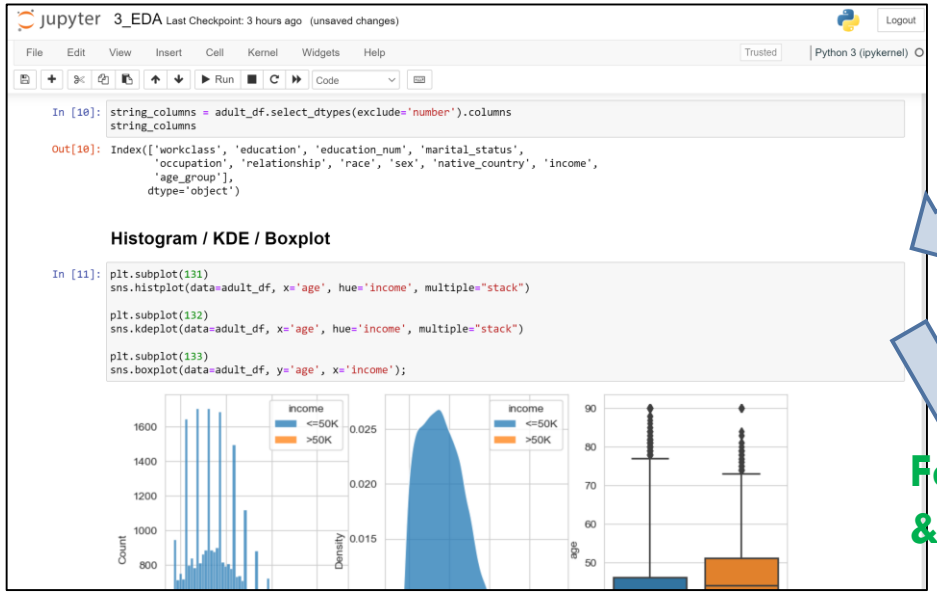
➢ IDE (Integrated Dev. Env.)

- ✓ Such as VS Code, PyCharm, Eclipse, Spyder, …
- ✓ Designed to maximize programmer productivity
- ✓ An application for software development providing
  - Code editing / Syntax highlighting / Code completion
  - Debugging / Building / Testing
  - Version control / Packaging
- ✓ One iteration might take a long journey (c.f., REPL)

BEST OF BOTH WORLDS

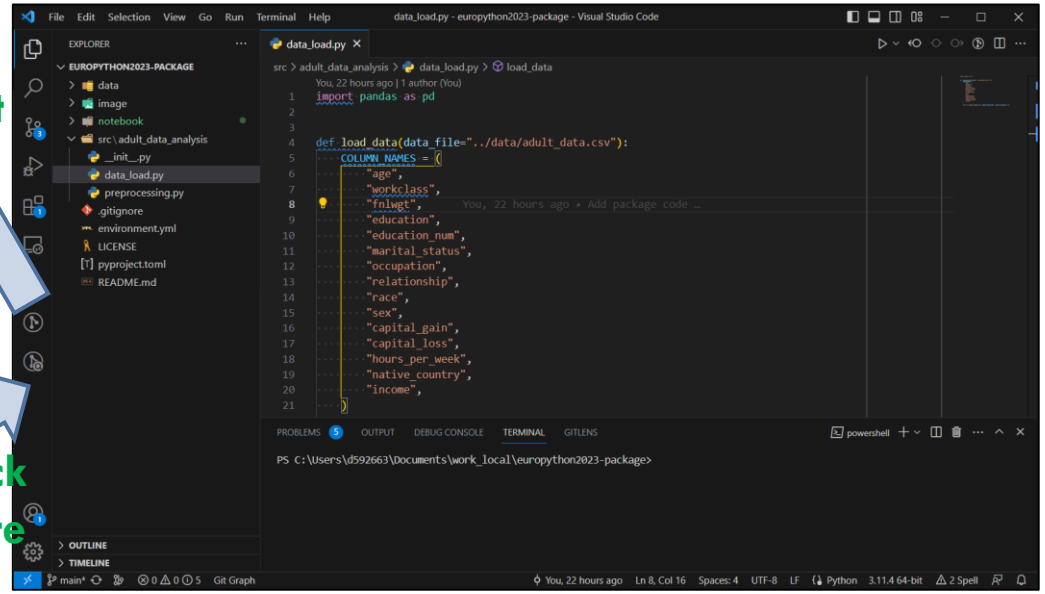
Jupyter Notebook:
REPL, Prototyping, Visualizations, Experiments, Documentation, …

IDE (Python Package):
Common code (function / class / module), Refactoring, Unit tests, Version control, Debugging, …

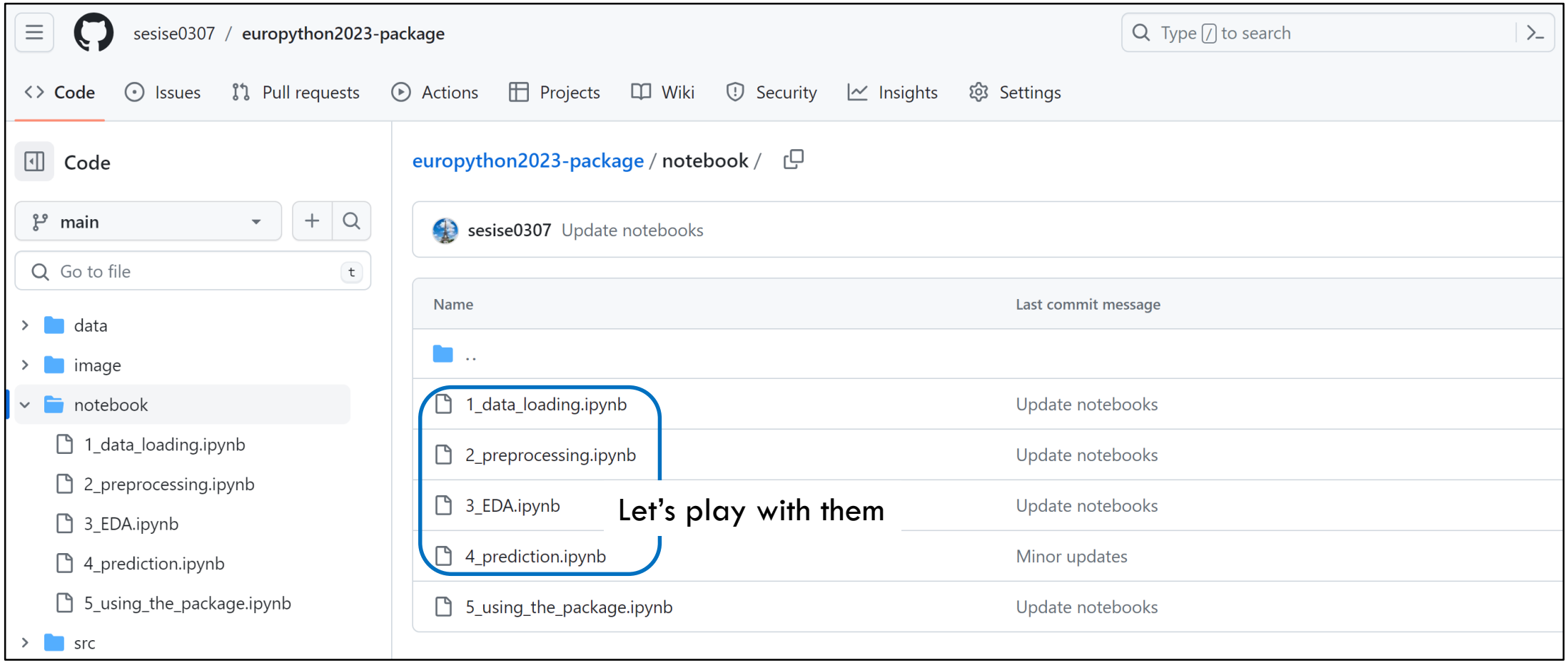


Import & Use

Feedback & Update



JUPYTER NOTEBOOK DATA SCIENCE WORKFLOW



TO (YOUR OWN) PYTHON PACKAGE

➢What is a Python Package?

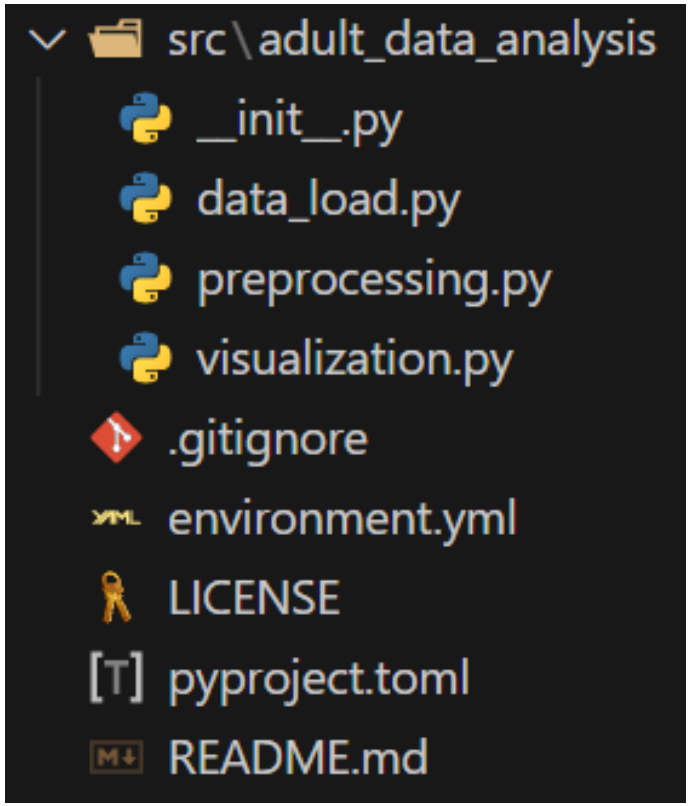
- ✓Any directory with an *__init__.py* file
- ✓A package can contain modules (python files) and sub-packages (sub-directories)
- ✓We can say that it is a collection of modules

➢Packages Make It Easy

- ✓To reuse and share code
- ✓To install with *pip* or *easy_install*
- ✓To specify as a dependency for another package
- ✓For other users to download and run tests
- ✓To add and distribute documentation

HOW TO CREATE A (MINIMAL) PACKAGE

- Pick a package name
  - ✓ All lowercase / Underscore-separated
  - ✓ In our case: *adult_data_analysis*
- Create a package root directory
  - ✓ Normally same to the package name
  - ✓ You can put extra files here such as *README.md* or *pyproject.toml*
- Create a package source directory
  - ✓ *src/package_name*[1]
  - ✓ Create a *__init__.py* file
  - ✓ Add python module files and put your code



[1]src-layout vs. flat-layout: https://setuptools.pypa.io/en/latest/userguide/package_discovery.html#automatic-discovery

HOW TO IMPORT AND USE YOUR PACKAGE

➢Option 1: Adding your local path

```
import sys

sys.path.append("../src")
```

```
import adult_data_analysis as ada
```

\+ Simple

– Path is relative to the notebook path

– Have to add the two lines of code in each notebook

– Not an usual way of importing packages

➢Option 2: Installing using `*pip*` (recommended)

✓Add a package metadata file

- *setup.py* | *setup.py* + *setup.cfg* | *pyproject.toml*
- Starting with PEP 621, the Python community selected *pyproject.toml* as a standard way

✓Install the package using `*pip*`

- `*pip install <path>*`

✓Then, we can just import it

```
import adult_data_analysis as ada
```

PYPROJECT.TOML EXAMPLE

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

[project]
name = "adult-data-analysis"
version = "0.0.1"
authors = [
    { name="Sin-seok SEO", email="sesise@gmail.com" },
]
description = "Adult data analysis package"
readme = "README.md"
dependencies = [
    "matplotlib",
    "pandas",
    "seaborn",
]

[tool.setuptools.packages.find]
where = ["src"]
```

DYNAMIC UPDATES OF A PACKAGE

- Jupyter *autoreload* magic command

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

  - ✓Level 0: Disable
  - ✓Level 1: Reload only modules imported with %aimport
  - ✓Level 2: Reload all modules (except those excluded by %aimport)
  - ✓Level 3: Same as 2/all, but also adds any new objects in the module

- Editable installation
  - ✓`*pip install --editable* <path>` → editable or develop mode
  - ✓C.f., `*pip install* <path>` → static installation / reinstallation required

- Let's check it out with the notebook *5_using_the_package.ipynb*

WRAP UP / SOME TIPS

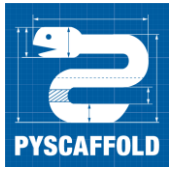
➢We Can Take Full Advantages of

- ✓ Jupyter Notebooks and IDEs
- ✓ By using a Python package
- ✓ And some simple tricks (*autoreload*, editable install)
- ✓ This will help you boost your productivity

➢Next Step

- ✓ Publish your awesome package to the Python Package Index (PyPI) and share with the world
- ✓ Refer to this page as a starting point:
  - https://packaging.python.org/en/latest/tutorials/packaging-projects/

➢PyScaffold



- ✓ Check this really nice package generator for bootstrapping high-quality Python packages

```
putup my_project
```

This will create a new folder called `my_project` containing a *perfect project template* with everything you need for getting things done.

➢VS Code is Great and Free

- ✓ Developer friendly and very customizable IDE
- ✓ Lots of features / extensions
- ✓ Very responsive
- ✓ Monthly updates

Thank you for your attention!

Q&A

REFERENCES

- ✓ https://setuptools.pypa.io/en/latest/userguide/quickstart.html
- ✓ https://python-packaging.readthedocs.io/
- ✓ https://setuptools.pypa.io/en/latest/userguide/pyproject_config.html
- ✓ *https://setuptools.pypa.io/en/latest/userguide/package_discovery.html*
- ✓ *https://packaging.python.org/en/latest/guides/single-sourcing-package-version/*