



Economia, Finanza e Analisi dei Dati

Laurea Magistrale

Data Analysis for Economics

01 - Introduction

Marco Nieddu

Fall 2026

This course: what it is

- ▶ **Third edition** of a course on using code (Python) to collect, organize, and analyze data in economics
- ▶ **Course goal:** learn to go
 - ▶ from an **economic question** to a dataset that can answer it
 - ▶ from the dataset to **charts and tables** that make an argument, in the structure of an economic report
- ▶ **Skills you will develop:** finding data in the wild (APIs, web pages), cleaning and wrangling, visualization, *reading and checking* code
 - ▶ practical (and often neglected) skills for dissertation and career
 - ▶ a prerequisite for the metrics and policy evaluation courses of the next terms

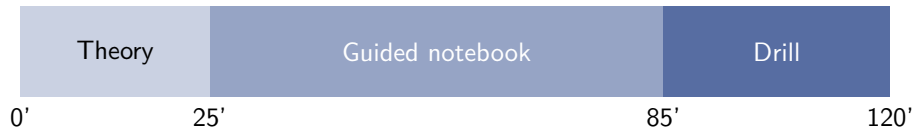
This course: what it is

- ▶ Besides coding, this is a **Topics in Applied Economics** course
- ▶ Four (?) topics. Each starts from theory and a short literature, states a few **claims**, and tests them on real data:
 1. The gender pay gap in Europe (Labor econ)
 2. Tax incidence: who paid for the 2022 fuel excise cut? (Public econ)
 3. Competition and pass-through in the fuel market (IO)
 4. Culture/religion and the economy (Political econ/Development)*
- ▶ Plus one class on **web data**: getting numbers out of the internet

This course: what it is *not*

- ▶ Econometrics is **not** the core of this course
 - ▶ We do descriptives and careful comparisons: control groups, indices, before/after, terciles, event studies
 - ▶ You will see how far you can get *before* running a regression, and where you cannot go without one
- ▶ It is not a course on Python syntax either
 - ▶ We learn the language by using it on data, with a small toolkit of recurring commands
- ▶ We fill the gaps left by traditional econometrics and methods classes
 - ▶ e.g., how to actually find datasets in the wild and clean them

How a class works



slides: the
question, the
theory, hypothe-
ses

you **run** pre-written cells, **predict** be-
fore running, **investigate** what a line
does, **modify** something

you **write** a short
script from stubs

Three ways you will meet code

Mode	File	What you do
Read and run	the notebook of the class	run cells, predict, modify one thing
Fill in	the drill script	complete the gaps, run the checker
Write from scratch	homework, project, exam	a plain script, start to end

- ▶ The three modes use the **same tool** (VS Code), so nothing changes on exam day
- ▶ If you get lost in a notebook: *Restart* and *Run All*, and you are back with me in thirty seconds

The toolkit

Every notebook introduces a few **tools**, numbered 1 to 26 across the course. Each answers one question:

- ▶ **Describe a cross-section:** ranking, map
- ▶ **Describe change:** trend, change since a baseline, convergence scatter, index rebased to 100
- ▶ **Isolate a policy:** decomposition, control product, margin, before/after, event study
- ▶ **Build the data:** API, HTML table, merge, loop and concat, pivot, text to dates
- ▶ **Space:** points and coordinate systems, spatial join, distance matrix
- ▶ **Groups:** terciles, group means

The big picture you should keep: *which tool answers which question*, not the syntax of each command.

Course preliminaries I: People

- ▶ Me: [Marco Nieddu](#)
 - ▶ Associate Professor in Public Economics (*Scienza delle finanze*)
 - ▶ I work on applied microeconomics (mostly education/development)
 - ▶ I'm an (applied) economist using data science tools for economics research, **not a data scientist!**
 - ▶ Office hours: by email (mgnieddu@unica.it)

Course preliminaries II: Course material

- ▶ Day 1: download and unzip DAEc2026.zip. The folder DAEc2026 is your **course folder**
- ▶ Every week: unzip the new topic and move its folder **inside** DAEc2026. A topic looks like this:

cl03_gender_pay_gap/	
01_gender_pay_gap.ipynb	the notebook of the class
drill_01.py	the drill (30 minutes)
data/	raw files as downloaded
output/	the figures you produce

- ▶ Slides before class
- ▶ Keep the course folder **out** of Dropbox /Onedrive

Course preliminaries II: Course material

Some notes/books I borrow from:

- ▶ Pedro H.C. Sant'Anna (Emory University), "Data Science for Economics", [link](#)
- ▶ Jared Hutchins (University of Illinois), "Data Science for Applied Economics", [link](#)
- ▶ Grant McDermott (Amazon), "Data Science for Economists", [link](#)
- ▶ Arthur Turrell, "Coding for Economists", [link](#)
- ▶ Florian Heiss and Daniel Brunner, "Using Python for Introductory Econometrics", [link](#)
- ▶ Gábor Békés & Gábor Kézdi, "Data Analysis for Business, Economics, and Politics". Cambridge UP, 2021

The question you are (rightly) asking

“An AI writes pandas better than I ever will.

Why sit through a course on it?”

- ▶ Fair question. For *syntax*, the answer is: it does
- ▶ But writing code stopped being the hard part. **Knowing what to compute, and whether the result is right, did not**
- ▶ This course is about the second thing.

A growing literature

- ▶ Large literature already on a key question:
is AI going to substitute or complement workers effort/ability?
 - ▶ **Brynjolfsson et al. (2025), Dell'Acqua et al. (2026), Noy and Zhang (2023), Kanazawa et al. (2026)**
- ▶ Who is going to catch up, and who is going to lose
- ▶ Is the frontier of possibilities expanding?
- ▶ Very open questions. No definitive answers yet.

The bottleneck moved

Ten years ago the slow part of data work was *typing the code*. Now it is:

1. asking a question worth answering
 2. specifying it precisely: which data, which unit, which comparison
 3. **reading and checking** the code and the numbers that come back
 4. interpreting them with economics
- Reading code fluently is now the core skill: **you cannot check what you cannot read + you cannot ask what you cannot imagine**

How we use AI in this course

Where	AI	Rule
Homework, project	✓	allowed and encouraged; be ready to explain every line
Drills (in class)	✗	30 minutes of your own reading and writing
Sit-in exam	✗	course material allowed, AI not

- ▶ Rationale: incentives to use AI as a **complement** (a learning tool), not a substitute
- ▶ In the project presentation you may be asked about any line of your code

How to ask an AI for code, and stay in control

- ▶ **Be specific:** “drop duplicate rows and keep the first occurrence”, not “clean the data”
- ▶ **Give context:** dataset, variable names, folder structure
- ▶ **Break it down:** one step at a time, not the whole analysis at once
- ▶ **Ask for explanations:** what each line does, not only the code
- ▶ **Check assumptions:** if it assumes something about your data, verify it on the data
- ▶ **Iterate:** rephrase, add constraints
- ▶ **Keep it simple:** if it proposes functions, lambdas or shortcuts you do not understand, ask for simpler, explicit code
- ▶ **Keep control:** run it, read the output, compare with what you expected. *You* are the one who signs

Exam/Grading

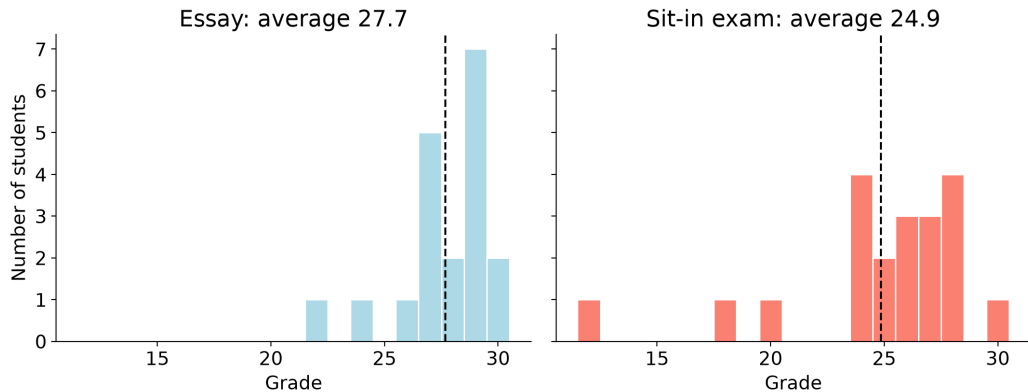
	Component	Weight	When	Rules
1	Individual project	50%	November 15	Open book (AI ✓)
2	Sit-in exam (PC)	50%	January (and later)	Closed book (AI ✗)

OR

	Component	Weight	When	Rules
1	Sit-in exam (PC)	100%	January session	Closed book (AI ✗)

Unlike last year, there is **no November exam**: the sit-in exam takes place in the January session (and in the later ones).

Which option to choose?



Grades of the last two years.

1. Individual project

- ▶ Output: 5-10 page essay on a topic of your choice (economics-related)
- ▶ Includes finding data, collecting/cleaning, and performing a brief analysis, following this structure:
 1. Abstract (whole story in 150 words)
 2. Research question (why important)
 3. Brief review of literature (what we know)
 4. Data description (which data/why/how they look)
 5. Results: 3-4 objects to summarize findings (+ comments)
 6. Scripts (and source files) to **reproduce** figures/tables/analyses
- ▶ This is exactly the structure of each topic in class

1. Individual project (cont'd)

- ▶ To get the grade, you must also give a short (10-15 min) presentation on your project at a later but not final stage
 - ▶ For me, needed to ensure substantial human contribution (I don't grade GPT)
 - ▶ For you, a chance to receive feedback and improve the project before submission
- ▶ **Important dates:**
 - ▶ Deadline to present the research idea: 15 October
 - ▶ Presentations: **TBA** (week 6)
 - ▶ Deadline to submit the final project: 15 November

2. Sit-in exam

The exam is a mini-version of the research project

- ▶ You are expected to import some data, analyze them, and write a very short comment
- ▶ Task simple enough to be completed in three hours
- ▶ **Closed-book exam** (no AI), but you may look at the course material
- ▶ The drills are the training for it: same tool, same kind of task, same time pressure
- ▶ Mock exam in week 6 shows you exactly what to expect

Course roadmap

- ▶ Lectures: Mon/Tue/Wed 12-14, six weeks

Week	Topic
W1	Intro and setup, Python basics
W2	Topic 1 (gender pay gap): theory, mechanisms
W3	Topic 2 (tax incidence): theory, average pass-through
W4	Topic 3 (competition): Local competition and heterogeneity
W5	Topic 4 (culture and the economy): theory, scraping
W6	Presentations, Mock exam

Data: Eurostat (W2), SISEN (W3), World Bank and MIMIT (W3), MIMIT and ISTAT (W4),

Getting started I

- ▶ **VS Code**: the editor. Opens notebooks and plain scripts in the same window, runs cells, shows your variables, and has a terminal panel built in
- ▶ **uv**: the installer. Reads the list of packages in the course folder and installs **exactly the same versions** on every laptop, Windows or Mac, in a minute or two

Advantages:

- ▶ one window for everything: notebooks in class, scripts in the drills and in the exam
- ▶ light: only the packages this course uses, stored inside the course folder
- ▶ a fallback if your laptop refuses: Google Colab has the same packages

Getting started II: VS Code and the course folder

1. Install **VS Code** from <https://code.visualstudio.com> and open it
2. Unzip DAEC2026.zip and move the whole folder into your home folder (/Users/you or C:\Users\you): **not** on the Desktop, in Documents, Dropbox, OneDrive or iCloud
3. In VS Code: File, Open Folder, pick DAEC2026. Mark the repository as “trusted”. Search for the extensions and **Install** the suggested ones: **Python, Jupyter, Python Environments, Data Wrangler**
4. Menu **Terminal**, then **New Terminal**: a command panel opens at the bottom of the window.

If VS Code offers to install uv or Python, click **No**: the next steps do it.

Getting started III: uv

5. Paste the line for your system and press Enter. macOS:

```
curl -LsSf https://astral.sh/uv/install.sh | sh
```

Windows:

```
powershell -ExecutionPolicy Bypass -c "irm https://astral.sh/uv/install.ps1 | iex"
```

6. Quit VS Code and open it again, then open a new terminal panel. Paste each line and press Enter:

```
uv sync
```

```
uv run check_setup.py
```

7. When VS Code asks which Python to use, pick **daec (3.14.7)** (see Getting started V)

Getting started IV: you are good to go when you see this

Folder: /Users/you/DAEc2026

Python: 3.14.7 at /Users/you/DAEc2026/.venv/bin/python

ok pandas 3.0.6

ok geopandas 1.1.4

...

SETUP OK

Then reload VS Code: Ctrl+Shift+P (Cmd+Shift+P on a Mac), then Reload Window. And this is what the drill checker looks like, after every drill:

drill_01.py

PASS Task 1 - df_age has 2023 only and three columns

FAIL Task 2 - wide has one row per country

hint: wide = df_age.pivot(index='geo', columns='age', values='OBS_VALUE')

Getting started V: make sure VS Code uses the course Python

VS Code has no Python of its own. It must use **daec (3.14.7)**, the course environment in `.venv`.

- ▶ **Scripts:** the label at the bottom right of the window must read `daec (3.14.7)`.
If it shows anything else, click it and choose `daec (3.14.7)`
- ▶ **Notebooks:** click *Select Kernel* at the top right, then *Python Environments*, then `daec (3.14.7)`
- ▶ If VS Code offers to install `ipykernel`, do **not** click Install: it means the wrong Python is selected

Other Pythons appear in the same lists, including `cpython-3.14.7`: never choose it. If `daec (3.14.7)` is missing, reload the window first.

Getting started VI: your first script

1. In the file list on the left, right-click c100, choose *New File* and call it `hello.py` (the name must end in `.py`)
2. Type this line in the empty file:

```
print("hello world")
```

3. Save it: `Cmd+S` on a Mac, `Ctrl+S` on Windows
4. Click the **play button** at the top right of the editor

If `hello world` appears in the terminal panel at the bottom, you are good to go.

The VS Code window, in one picture

1. Files

cl03_gender_pay_gap/
01_gender_pay_gap.ipynb
drill_01.py
data/
output/

2. The notebook or the script

```
df2023 = df[df["TIME_PERIOD"] == 2023].copy()
```

▶ Run Cell

```
plt.barh(df2023["geo"], df2023["OBS_VALUE"])
```

▶ Run Cell

output and chart appear here (notebook), or in a side panel with the variable explorer (script)

3. Terminal panel (menu Terminal, New Terminal)

```
uv sync
```

```
uv run check_drill.py cl03_gender_pay_gap/drill_01.py
```

Why are we using Python?

We are going to focus on one language: **Python**. Why?

1. **General purpose**: used for data, scraping, APIs, machine learning, even apps.
Downside: econometrics support weaker than R and Stata
2. **Open source**: free, maintained by a community, a package for almost everything.
Downside: packages do not always work well together, hence the pinned environment
3. **Easy to use**: high-level, interpreted, readable. Downside: slow on very large data
4. **The language AI assistants know best**: which cuts both ways, see the previous section

Popularity: [PYPL index](#), [r4stats](#)

Addendum: Data visualization

- ▶ When it is beautiful
 - ▶ <https://www.reddit.com/r/dataisbeautiful/>
- ▶ When it is not
 - ▶ <https://www.reddit.com/r/dataisugly/>

References for the AI slides

- ▶ Brynjolfsson, E., Li, D., Raymond, L. (2025). Generative AI at Work. *The Quarterly Journal of Economics*, 140(2), 889–942. [link](#)
- ▶ Dell’Acqua, F., McFowland III, E., Mollick, E., Lifshitz, H., Kellogg, K. C., Rajendran, S., Kraymer, L., Candelon, F., Lakhani, K. R. (2026). Navigating the Jagged Technological Frontier: Field Experimental Evidence of the Effects of Artificial Intelligence on Knowledge Worker Productivity and Quality. *Organization Science*, 37(2), 403–423. [link](#)
- ▶ Kanazawa, K., Kawaguchi, D., Shigeoka, H., Watanabe, Y. (2026). AI, Skill, and Productivity: The Case of Taxi Drivers. *Management Science*, 72(2), 1376–1388. [link](#)
- ▶ Noy, S., Zhang, W. (2023). Experimental Evidence on the Productivity Effects of Generative Artificial Intelligence. *Science*, 381(6654), 187–192. [link](#)