PROGRAMMING WITH PYTHON

Using ASHE-Census Synthetic Data

Dr Kathyrn R Fair

HOUSEKEEPING

- The session will be recorded and made available upon request

  If you don't want your image/voice recorded, please make use of the chat

- Schedule:

  09:30 – 10:45 Welcome and Introduction, Guided Code Walkthrough

  10:45 – 11:00 Break

  11:00 – 12:30 Using AI assistance, Group practical and problem solving, Peer Review

SESSION STRUCTURE

- Python setup and coding standards
- Data exploration and visualisation
- Indicator extraction and cleaning
- Derived variables and proxy outcomes
- AI coding assistance
- Final group problem + peer review

INSTALLING PYTHON & SPYDER

- Recommended: Install Anaconda (www.anaconda.com)
- Includes Python, Spyder IDE, and libraries
- Alternatively: install Python and use pip to install Spyder
- Use conda or pip for package management
- Create virtual environments for dependency isolation

PYTHON LIBRARIES

A **library** is a collection of pre-written code that adds extra tools or shortcuts to Python

Popular libraries we'll use today:

- 📊 **pandas** – Work with tables (like Excel or R dataframes). Load, clean, filter, and summarise data.
- 🔢 **numpy** – Fast calculations on numbers, arrays, and matrices. Often used behind the scenes by pandas.
- 📈 **matplotlib** – Create charts and figures. Very flexible, like a plotting toolkit.
- 🎨 **seaborn** – Easier, cleaner plotting. Built on matplotlib but designed for statistics.

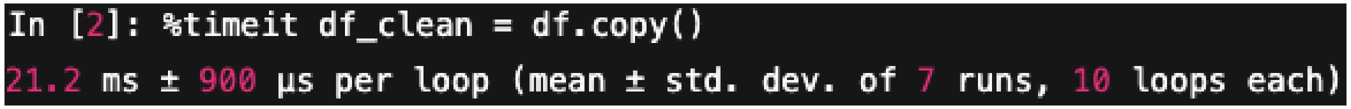
R equivalents: pandas = dplyr, numpy = matrixStats, matplotlib & seaborn = ggplot2.

CLEAN AND REPRODUCIBLE CODE

- **Organise by step:** load → clean → analyse → export
- **Comment why,** not just what
- **Avoid repetition:** use loops or functions
- **Test as you go:** use Spyder's console + Variable Explorer
- **Stick to core tools:** pandas for data, seaborn for plots
- **Write for others:** use clear variable names (hourly_pay, not hp)

TIMING CODE AND WORKING EFFICIENTLY

- Use %timeit in Spyder to test performance of operations



```
In [2]: %timeit df_clean = df.copy()
21.2 ms ± 900 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

- Helpful for understanding bottlenecks in large datasets
- For longer scripts, use time.time() to benchmark sections
- Optimise only where performance matters

FROM PYTHON TO R – WHEN AND WHY

- R is powerful for modeling: survey design, imputation, mixed models
- Export clean data from Python with df.to_csv(...)
- Check for missing codes (-99) and label categories clearly
- You may want to use Python for cleaning, R for modeling

USING AI FOR PYTHON HELP

- ✅ Use it for syntax, cleaning snippets, or refactoring
- ❌ Don’t use it to run full analyses without review
- **Be specific:** describe your data and your goal clearly
- Use AI tools only with **non-sensitive** data or documentation
- **Review AI code like student work:** Is it correct? Clear? Scalable?