Lectures on causal inference and experimental methods

Macartan Humphreys

Subsection 1

Getting started

Plan

- General aims and structure
- Expectations
- Pointers for exercises
- Quick `declaredesign` intro

Aims

- Deep understanding of key ideas in causal inference
- Transportable tools for understanding how to evaluate and improve design
- Applied skills for design and analysis
- Exposure to open science practices

Syllabus

https://macartan.github.io/ci/syllabus.pdf

The topics

Day 1: Intro

- 1.1 Course outline, tools,
- 1.2 Introduction to Declare design

Day 2: Causality

- 2.1 Fundamental problems and basic solutions
- 2.2 General inquiries and causal identification

The topics

Day 3: Estimation and Inference

- 3.1 Frequentist
- 3.2 Bayesian

Day 4:

- 4.1 Experimental Design
- 4.2 Design evaluation (incl power) @secdiagnosis

Day 5:

- 5.1 Topics and techniques @ref(citopics)
- 5.2 Open science @ref(openscience)

Expectations

- 5 tasks
- (Required) Work in four "exercise teams": 1 team per session $\times 4$
- (Optional) Prepare a research design or short paper, perhaps building on existing work. Typically this contains:
  - a problem statement
  - a description of a method to address the problem
  - analytic or simulation based results describing properties of the solution
  - a discussion of implications for practice. A passing paper will illustrate subtle features of a method; a good paper will identify unknow properties of a method; en excellent paper will develop a new method.
- Plus general reading and participation.

Exercise team job

Teams should prepare 15 - 20 minute presentations on set puzzles. Typically the task is to:

- Take a puzzle, theorem, claim
- Declare and diagnose a design that shows the claim operating (e.g. some estimator produces unbiased estimates under some condition)
- Modify the design to show behavior when conditions are violated
- Share a report with the class. Best in self-contained documents for easy third party viewing. e.g. `.html` via `.qmd` or `.Rmd`

See example in `git`.

Good coding rules

- https://bookdown.org/content/d1e53ac9-28ce-472f-bc2c-f499f18264a3/code.html
- https://www.r-bloggers.com/2018/09/r-code-best-practices/

Good coding rules

- Metadata first
- Call packages at the beginning: use `pacman`
- Put options at the top
- Call all data files once, at the top. Best to call directly from a public archive, when possible.
- Use functions and define them at the top: comment them; useful sometimes to illustrate what they do
- Replicate first, re-analyze second. Use sections.
- Have subsections named after specific tables, figures or analyses

Aim

Nothing local, everything relative: so please do not include hardcoded paths to your computer

- First best: if someone has access to your `.Rmd`/`.qmd` file they can hit render or compile and the whole thing reproduces first time.
- But: often you need ancillary files for data and code. That's OK but aims should still be that with a self contained folder someone can open a `master.Rmd` file, hit compile and get everything. I usually have an `input` and an `output` subfolder.

Collaborative coding / writing

- Do not get in the business of passing attachments around
- Share self contained folders; folders contain a small set of live documents plus an archive. Old versions of documents are in archive. Only one version of the most recent document is in a main folder.
- Data is self contained folder (`in`) and is never edited directly
- Update to github frequently