

# Big Data Experiments as Data Pipelines

## Conceptual Questions

Please write a three to ten sentence explanation for **ONE** of the two questions below.

1. **Estimands at scale (ITT vs “product impact”).** In a platform A/B test, the treatment may (i) fail to deliver to some users, (ii) deliver but users may not engage, and (iii) outcomes may be measured through fragile logs.
  - Define the ITT estimand in this setting.
  - Give one reason ITT is the default for decision-making in production experiments.
  - Explain one case where ITT is not the estimand a research audience wants, and what additional assumptions you would need to target an alternative estimand.
2. **Measurement as part of the design.** Large experiments rely on instrumentation and event logs (missing events, changing definitions, bots).
  - Give two concrete examples of how logging changes can create “fake treatment effects”.
  - Explain why randomization does not protect you from measurement drift.
  - Propose one monitoring strategy that would detect instrumentation problems early (what would you plot or test?).

## Applied Exercises

Use the code in the week’s code tutorial and the lecture slides to answer the following questions.  
bigskip

3. **Add a retention-style outcome and estimate its ATE.** Extend the pipeline so that, in addition to the existing outcomes, you compute a user-level retention / activity measure.
  - From the user-day logs, create `days_active` = the number of days with `active == 1` for each user (ignore missing days).
  - Create `retained_any` = 1 if `days_active`  $\geq 1$ , else 0.
  - Add both outcomes to the analysis-ready dataset and estimate the ATE using:
    - (a) difference in means, and
    - (b) regression adjustment with `factor(block)` and cluster-robust SEs clustered at `cluster_id`.
  - Save your results as `outputs/tables/ate_retention.csv`.
4. **Simulate noncompliance and compare ITT vs TOT (IV).** Modify the experiment so that treatment assignment does not always translate into treatment receipt.
  - Create a variable `received` such that:
    - all controls have `received = 0`,
    - treated units have `received = 1` with probability  $p < 1$  (choose and report your  $p$ ),
    - (optional) let  $p$  depend on `platform` or `baseline_activity`.

- Redefine the outcome generation so that the treatment effect operates through **received** rather than **treat**.
  - Compute:
    - (a) ITT: regress the outcome on **treat** (your original approach),
    - (b) TOT / LATE using IV: treat **treat** as an instrument for **received**.
  - Report the ITT and TOT estimates side-by-side and explain (2–5 sentences) why TOT is typically larger in magnitude than ITT in your simulation.
  - Save your results as `outputs/tables/itt_vs_tot.csv`.
5. **Multiple outcomes + multiple testing (BH/FDR).** In big experiments, it is easy to “find significance” by testing many outcomes.
- Create 10 outcome variables at the user level:
    - 1 outcome with a real treatment effect (use one of your existing outcomes),
    - 9 placebo outcomes with **no** treatment effect (simulate them so they depend on baseline covariates but not **treat**).
  - For each outcome  $k$ , estimate the treatment effect with robust SEs and extract a p-value.
  - Apply Benjamini–Hochberg (BH) correction to control the false discovery rate.
  - Create a table with columns: outcome name,  $\hat{\tau}$ , p-value, BH-adjusted p-value, and an indicator for whether it is significant at  $q = 0.05$ .
  - Save your results as `outputs/tables/multiple_testing.csv`.

## Challenge Question (Optional — if you finish early)

Choose **ONE** option.

- (a) **Randomization inference.** Implement a randomization-inference (permutation) p-value for the treatment effect on **converted**.
  - Keep the outcomes fixed.
  - Permute **treat** within blocks (or explain why you permuted globally).
  - Compute the null distribution of the difference-in-means estimator and report a two-sided p-value.
  - Plot the null distribution and mark the observed estimate.
- (b) **Interference / spillover simulation.** Simulate spillovers within `cluster_id` so that control users’ outcomes depend on the fraction of treated users in their cluster.
  - Define an exposure variable, e.g., `exposure = share treated in cluster`.
  - Modify outcome generation so that outcomes depend on both **treat** and **exposure**.
  - Show (briefly) how naive user-level analysis can mis-estimate the direct effect when interference exists.