Registered Report Template: Preregistration, Deviations, and Claim Boundaries

A reproducible exemplar for registered analysis plans

Daniel Ari Friedman
Active Inference Institute
daniel@activeinference.institute
ORCID: 0000-0001-6232-9096
DOI: 10.5281/zenodo.21298892

2026-07-10

Contents

1 Abstract

This document is a **template**, not an empirical study. It demonstrates the registered-report workflow end to end: locking a preregistration, validating its completeness, executing the registered analysis plan against deterministic demonstration data, and reporting confirmatory and exploratory claims through an explicit deviation ledger. Every quantity reported here is produced by the tested code in `src/registered_report/` and regenerated by `scripts/generate_figures.py`; none is hand-entered or illustrative.

The demonstration binds a single confirmatory hypothesis (H1) to one registered outcome (`primary_score`) analysed by a two-sided label-permutation test. Run on a seeded two-group dataset (`seed = 20260709`, `n = 24` per group), the registered test yields an observed mean difference of `1.003` with a two-sided permutation p-value of `0.0005` (`0` of `2000` shuffles at least as extreme), significant at the preregistered `alpha = 0.05`. A deliberately introduced secondary endpoint and an alternative model are carried only as **documented deviations**, keeping the confirmatory claim boundary intact. The purpose is to give forks a working, auditable skeleton in which planned analyses, frozen hypotheses, deviations, and post-run claims are separable and machine-checkable.

2 Introduction

A registered report separates *what a study commits to* from *what the data later show*. Hypotheses, outcomes, exclusion rules, and the analysis plan are locked and time-stamped before results are observed; only then is the analysis run and interpreted. This ordering is the structural defence against the degrees-of-freedom problems that inflate false-positive rates in flexible analyses [Simmons et al., 2011] and motivates the broader preregistration and research-transparency movement [Nosek et al., 2018, Munafò et al., 2017, Chambers, 2013].

This template operationalises that discipline as tested code rather than prose convention. The `registered_report` package freezes a registration payload under a content hash, validates that the required preregistration sections are present and well formed, and — after execution — compares the analysis that was actually run against the plan that was locked. Any departure must appear in a **deviation ledger** with a rationale before it can be reported, and confirmatory claims are kept mechanically distinct from exploratory ones.

To make the workflow concrete and reproducible, the template ships a **deterministic demonstration study**. It contains no real empirical data: a seeded generator synthesises a two-group dataset, and the registered label-permutation test is executed against it. Because both the data and the permutation schedule are seeded, every figure and every number in this document is byte-stable and is regenerated on demand from `scripts/generate_figures.py`. Readers forking this template should replace the demonstration fixture in `data/example_registration.json` and the synthetic data generator with their own registration and real data collection, keeping the same lock-validate-execute-compare structure.

The remainder of this report states the preregistered hypotheses (Section 3), describes the frozen methods and analysis plan (Section 4), reports the confirmatory outcome bound to the executed analysis (Section 5), records deviations (Section 6), and discusses scope and boundaries (Section 7).

3 Preregistered hypotheses

The registration in `data/example_registration.json` declares its hypotheses, outcomes, exclusion rules, and analysis plan before any result is observed. It is frozen under a SHA-256 content hash (prefix `96a34a11d132`); the validator in `registered_report.protocol.validate_registration` confirms the payload carries every required section with no structural findings.

3.1 Confirmatory hypothesis

- **H1** — *The proposed condition improves the primary score.* This is the sole confirmatory hypothesis. It maps to the registered outcome `primary_score`, measured as a mean score difference and analysed with a two-sided label-permutation test at `alpha = 0.05`.

Each registered hypothesis is bound to exactly one registered outcome and one registered analysis. fig. 1 renders that binding directly from the frozen registration via `hypothesis_outcome_map`, so the figure cannot silently drift from the locked plan.

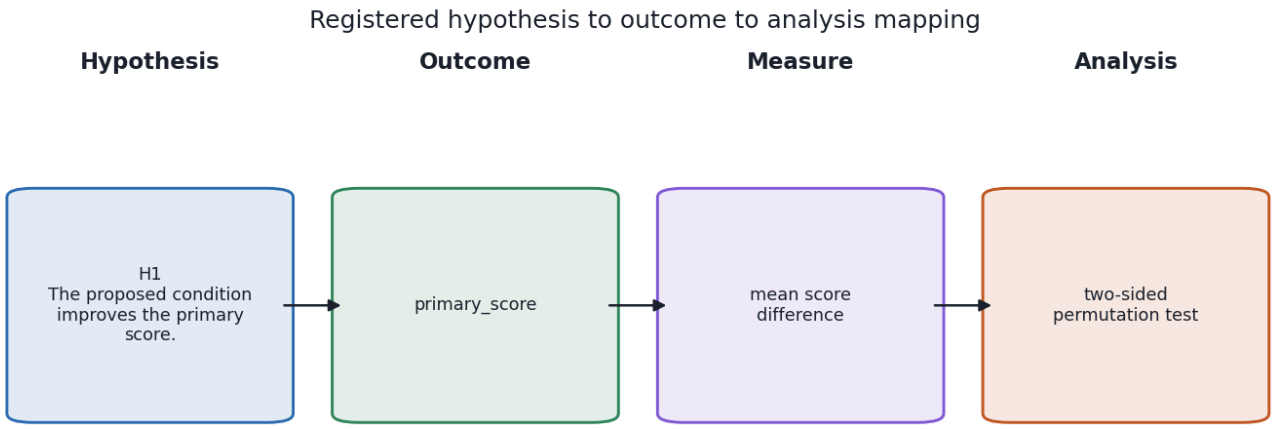

Figure 1: Registered hypothesis-to-outcome-to-analysis mapping for the demonstration registration. Each row is generated by `hypothesis_outcome_map(registration)` in `src/registered_report/demo_study.py`: hypothesis H1 links to the `primary_score` outcome, the `mean score difference` measure, and the `two-sided permutation test` analysis. Only registered hypotheses appear; exploratory endpoints are excluded by construction.

3.2 Registered outcome and exclusion rule

- **Primary outcome** `primary_score` — mean score difference between the proposed and reference conditions, evaluated by a two-sided permutation test.
- **Exclusion rule** — records missing the primary outcome are excluded *before* group assignment is inspected, so exclusion decisions cannot depend on the observed effect.

No secondary outcome is preregistered. Any additional endpoint introduced during analysis is therefore exploratory and must be recorded in the deviation ledger (Section 6) rather than presented as a confirmatory finding.

4 Methods and analysis plan

The method is the registered-report workflow itself, implemented as tested code. It has four locked stages that complete before any result is observed, followed by two interpretation stages that run only after the deviation ledger is built. fig. 2 shows this ordering and the lock boundary it protects.

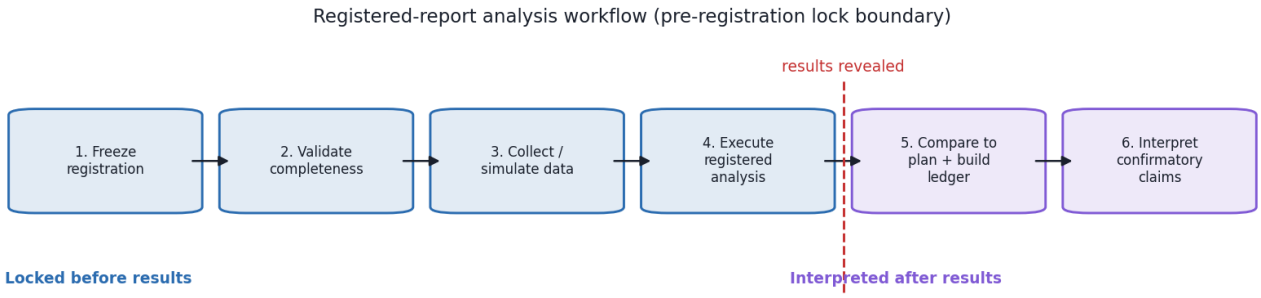

Figure 2: Registered-report analysis workflow, rendered from `analysis_plan_stages()` in `src/registered_report/demo_study.py`. The four blue stages — freeze registration, validate completeness, collect/simulate data, and execute the registered analysis — lock before results are revealed (dashed boundary). The two purple stages — compare to plan and build the deviation ledger, then interpret confirmatory claims — run only afterwards. The `locks_before_results` flag on each stage is what enforces the boundary in code.

4.1 Frozen registration

`freeze_registration` deep-copies the registration, strips any pre-existing hash, and stamps a canonical SHA-256 hash over the sorted-key JSON encoding. `validate_registration` then checks that `title`, `version`, `hypotheses`, `outcomes`, `exclusion_rules`, and `analysis_plan` are present; that each hypothesis has a unique `id` and a `claim`; that each outcome names a `measure` and `analysis`; that `analysis_plan.primary_model` and `seed` are set; and that registered-report stage and ethics-review metadata are supplied. For the demonstration registration this validation returns no findings.

4.2 Registered analysis plan

The analysis plan is fixed before execution:

- **Primary model** — a two-sided label-permutation test on the group mean difference.
- **Significance threshold** — `alpha = 0.05`.
- **Seed** — `20260709`, used for both data synthesis and the permutation schedule so the result is reproducible.
- **Permutations** — `2000` label shuffles; the two-sided p-value uses the add-one correction `(k + 1) / (permutations + 1)` and is therefore strictly positive.

4.3 Deterministic demonstration data

Because this is a template, the "data collection" stage is a seeded synthetic generator (`generate_demo_dataset`), **not** a real study. It draws `n = 24` control observations from `Normal(0, 1)` and `n = 24` treatment observations from `Normal(0.8, 1)` using a single seeded generator, so the entire dataset is a pure function of the plan seed. `run_registered_analysis` reads the seed and `alpha` back out of the frozen plan and executes the registered permutation test against this dataset, guaranteeing the executed analysis honours the locked plan. The executed summary is written to `output/data/demo_analysis.json` and is the single source of truth for every number in Section 5.

Forks replace this generator with genuine data collection while keeping the same `run_registered_analysis` contract, so the confirmatory analysis remains bound to the frozen plan rather than to post-hoc choices.

5 Results

Results are interpreted only after the registered analysis has run and the deviation ledger (Section 6) has classified any departures from the plan. Every value below is read from `output/data/demo_analysis.json`, which is regenerated by `scripts/generate_figures.py` from the tested `run_registered_analysis` function.

5.1 Confirmatory outcome (H1, `primary_score`)

On the deterministic demonstration data, the registered two-sided permutation test returns:

Quantity	Value
Control mean	`-0.178991`
Treatment mean	`0.823948`
Observed mean difference	`1.002940`
Permutations	2000
Shuffles with absolute difference >= observed	0
Two-sided permutation p-value	`0.0005`
Preregistered `alpha`	`0.05`
Confirmatory decision	significant

The observed difference of `1.003` lies far in the right tail of the permutation null distribution: none of the 2000 label shuffles produced a mean difference as extreme, so the add-one-corrected p-value is `0.0005`, below the preregistered `alpha = 0.05`. H1 is therefore **supported** as a confirmatory result. fig. 3 shows the observed statistic against the null it was tested against.

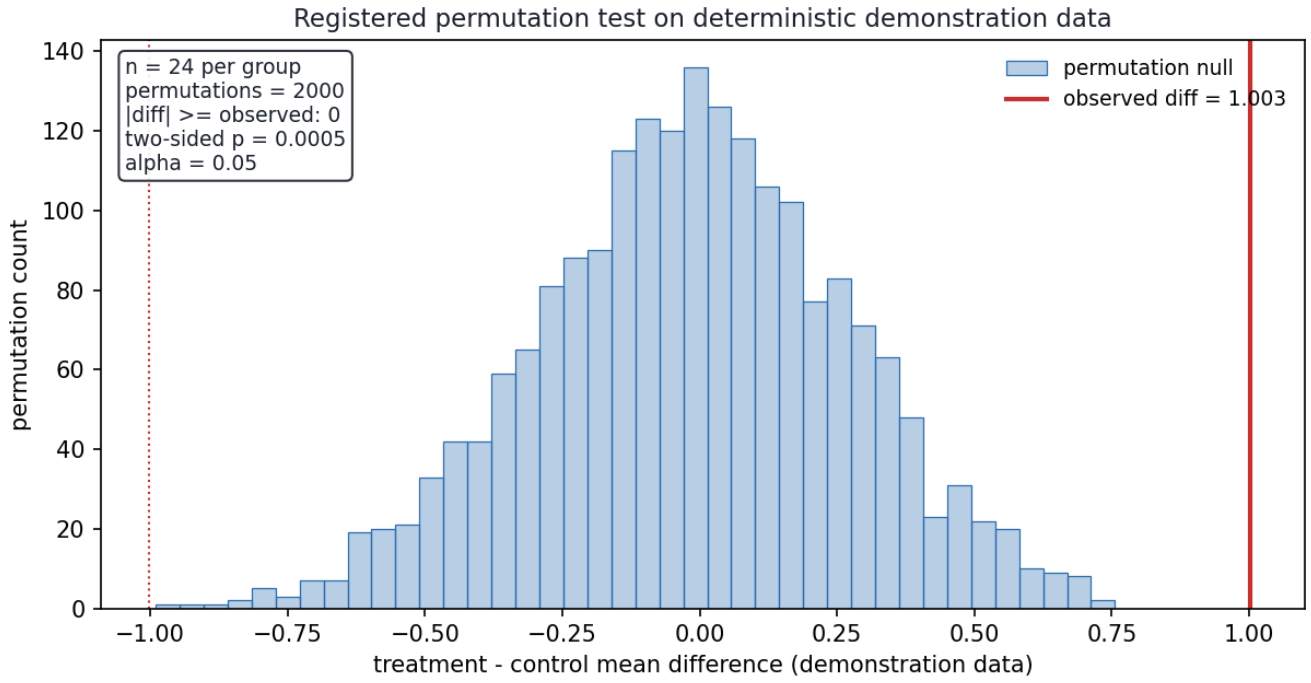

Figure 3: Registered permutation test on the deterministic demonstration data, rendered by `plot_permutation_result` from `permutation_null_distribution` and `run_permutation_test`. The histogram is the label-permutation null distribution of the treatment-minus-control mean difference (2000 seeded shuffles); the solid red line marks the observed difference of `1.003` and the dotted line its mirror image. The annotation reports `n = 24` per group, `0` shuffles at least as extreme, a two-sided p-value of `0.0005`, and `alpha = 0.05`. The observed statistic sits outside the entire null, which is why the primary confirmatory claim holds.

5.2 Claim boundary

The registration preregisters exactly one confirmatory outcome. The review packet built by `build_review_packet` records `primary_score` as the sole confirmatory outcome and returns an integrity score of `1.0` for a plan-faithful execution. Any other endpoint examined during analysis is exploratory and is reported under Section 6, never here. This separation is enforced in code, not merely asserted in prose.

6 Deviation register

A registered report must state, explicitly, every way the executed analysis departed from the frozen plan. This template treats that register as a first-class artifact: `build_deviation_ledger` produces one row per executed outcome and model, classifying each as `ok` (registered), `warning` (an unregistered element carried with a documented rationale), or `error` (an unregistered element with no rationale). `compare_analysis_to_registration` downgrades a documented model change from `error` to `warning` and refuses to accept any deviation whose `rationale` is empty.

6.1 Demonstration deviations

To exercise the machinery, the demonstration executes a deliberately plan-divergent analysis: it adds a `secondary_score` endpoint and swaps the registered `permutation_test` for a `linear_model`, each accompanied by a rationale. The resulting ledger is:

Order	Kind	Target	Registered	Documented	Severity
0	outcome	`primary_score`	yes	—	`ok`
1	outcome	`secondary_score`	no	yes	`warning`
2	model	`linear_model`	no	yes	`warning`

Because both departures are documented, the review packet remains valid: it reports `primary_score` as the only confirmatory outcome, `secondary_score` as exploratory, and an integrity score of `0.9` (a single `0.10` warning penalty against a perfect plan). Had either deviation lacked a rationale, the same code would have raised a blocking `deviation_without_rationale` or `unregistered_outcome` error, and the packet would be invalid. fig. 4 plots this register along the registered-to-executed axis.

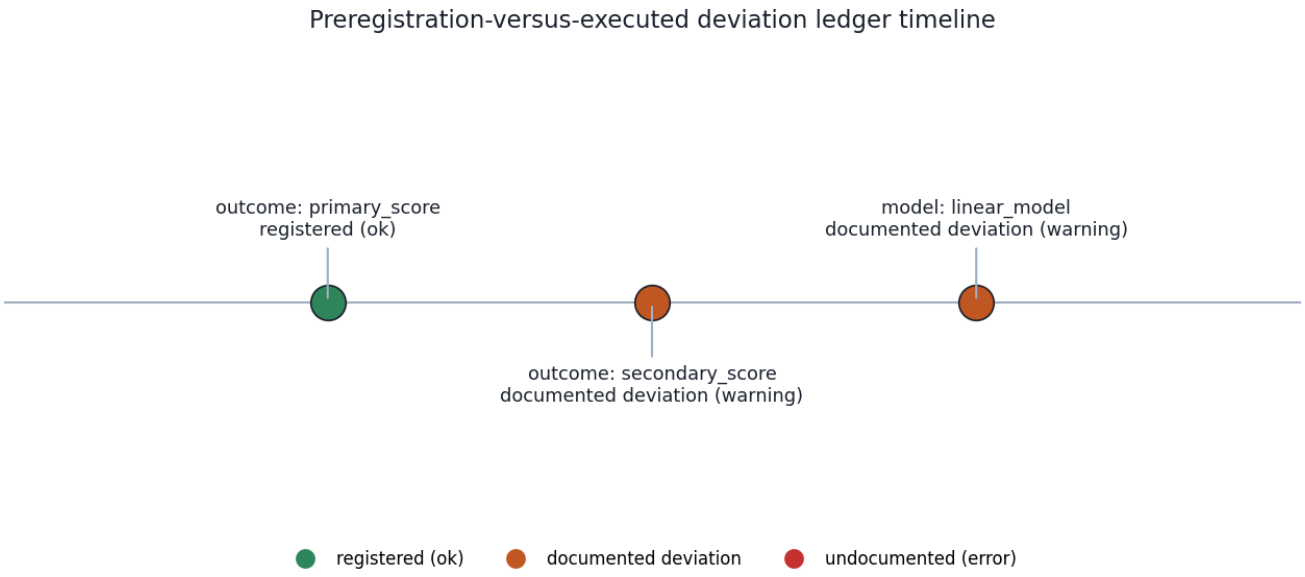

Figure 4: Preregistration-versus-executed deviation ledger timeline, rendered by `plot_deviation_timeline` from `build_deviation_ledger` and `deviation_timeline`. Each marker is a ledger row coloured by severity: the registered `primary_score` outcome is green (`ok`), while the added `secondary_score` endpoint and the `linear_model` substitution are amber (`warning`, documented deviations). No marker is red because every departure carries a rationale. An undocumented departure would appear in red and invalidate the review packet.

6.2 Reading the register

The register is the mechanism that keeps Section 5 honest: the confirmatory claim is restricted to the registered `primary_score` outcome, and the exploratory `secondary_score` endpoint is visible, labelled, and excluded from

confirmatory interpretation. Forks should extend this ledger — never edit the frozen registration — whenever the executed analysis must diverge from plan.

7 Discussion

7.1 What the demonstration does and does not show

The demonstration shows that the registered analysis, run against seeded data with a genuine effect (`Normal(0.8, 1)` treatment versus `Normal(0, 1)` control), recovers a significant confirmatory result: an observed mean difference of `1.003` with a two-sided permutation p-value of `0.0005`. It does **not** show anything about any real-world phenomenon. The data are synthetic and the effect is injected by construction; the number exists to prove that the locked plan, when executed, produces an auditable statistic — not to support a scientific claim. This is the honest reading of a template: the value is a property of the code path, not evidence about the world.

7.2 Why the boundaries are enforced in code

Prose conventions for "confirmatory versus exploratory" are easy to blur once results are in view. Here the boundary is a function contract: `compare_analysis_to_registration` refuses unregistered outcomes without a documented deviation, `build_deviation_ledger` assigns a severity to every executed element, and `build_review_packet` partitions confirmatory from exploratory outcomes deterministically. The content hash on the frozen registration makes silent edits to the locked plan detectable (`hash_mismatch`). Together these turn registered-report discipline into checks that fail loudly rather than guidance that can be quietly ignored.

7.3 Limitations and scope

- The demonstration uses a single hypothesis and a single confirmatory outcome; real registrations will declare several, and the same helpers scale to them.
- A permutation test is used because it is exact, deterministic, and dependency-free; forks may register any primary model by naming it in the plan.
- Ethics-review and registered-report-stage metadata in the fixture are synthetic (`status: exempt`, `authority : synthetic-fixture-review`) and must be replaced with real values before any submission.

7.4 Using this template

Fork with `scripts/audit/copy_exemplar.py`, replace `data/example_registration.json` with your own preregistration, swap the synthetic generator in `src/registered_report/demo_study.py` for real data collection, record stage and ethics metadata, and rerun the tests and figure generation. Keep confirmatory claims tied to registered outcomes, record every departure in the deviation ledger, and never let post-run result prose rewrite the preregistered intent.

8 References

The bibliography lives in `manuscript/references.bib` and is read by Pandoc during PDF rendering. Every `[@key]` citation in the manuscript is resolved against that file, so the reference list below is generated from the cited entries rather than maintained by hand.

To validate that `references.bib` is syntactically clean and has the required fields per entry type, run from the repository root:

```
uv run python -m infrastructure.reference.citation.cli validate \
    projects/templates/template_registered_report/manuscript/references.bib --strict
```

References

Christopher D. Chambers. Registered reports: A new publishing initiative at cortex. *Cortex*, 49(3):609–610, 2013. doi: 10.1016/j.cortex.2012.12.016.

Marcus R. Munafò, Brian A. Nosek, Dorothy V. M. Bishop, Katherine S. Button, Christopher D. Chambers, Nathalie Percie du Sert, Uri Simonsohn, Eric-Jan Wagenmakers, Jennifer J. Ware, and John P. A. Ioannidis. A manifesto for reproducible science. *Nature Human Behaviour*, 1(1):0021, 2017. doi: 10.1038/s41562-016-0021.

Brian A. Nosek, Charles R. Ebersole, Alexander C. DeHaven, and David T. Mellor. The preregistration revolution. *Proceedings of the National Academy of Sciences*, 115(11):2600–2606, 2018. doi: 10.1073/pnas.1708274114.

Joseph P. Simmons, Leif D. Nelson, and Uri Simonsohn. False-positive psychology: Undisclosed flexibility in data collection and analysis allows presenting anything as significant. *Psychological Science*, 22(11):1359–1366, 2011. doi: 10.1177/0956797611417632.