Megacell v13.13.9
Section-by-Section Code Walkthrough

Evan Wesley & Vivi The Physics Slayer!

September 5, 2025

Contents

1 High-level Overview

Goal. Test whether observed signals align with a preregistered *registry/ledger* of simple fractions (esp. dyadics $1/2^k$) using multiple endpoints:

- Per-block and per-cluster Monte Carlo *PB tails* (exceedance probabilities);
- Family-wise error rate (FWER) over the preregistered endpoint family;
- Bayes factors contrasting an MDL*-weighted dyadic spike prior against a slab (Jeffreys-like $\mathrm{Beta}(\frac{1}{2}, \frac{1}{2})$);
- E-values and meta-combinations (Tippett min-p, Simes p);
- Stability: leave-one-out (dataset) and jackknife (block);
- Exploratory distance statistic T.

Reproducibility. We fix RNG seeds, echo ledger file SHA256s, compute a canonical *registry hash*, provide a code fingerprint, and emit an artifact bundle (CSV/JSON + README) under `results/run_YYYYMMDD-HHMMSS/`.

2 Imports, Utilities, and Artifacts

We rely on standard Python libraries (filesystem, hashing, RNG) and `numpy`. Artifact helpers create an audit trail:

- `ensure_outdir`: creates `results/run_.../`, plus `sections/` and `ledger/` subfolders.
- `write_text/json/csv`: consistent encodings and headers.
- `Tee`: optional console log teeing.
- `dump_environment`: stores Python/NumPy versions and system basics.

Why: Professional, machine-readable receipts for replication and audit.

3 Registry (Ledger) Parsing and Hashing

What. The registry is a set of rational numbers in $(0, 1)$.

- `parse_fracs_from_text`: extracts fractions a/b and decimals strictly in $(0, 1)$ (tokenized to avoid overlaps).
- `try_load_ledger`: reads candidate files, parses to `Fraction` list, records per-file SHA256s.
- `working_ledger`: merges a small fallback set and file-derived fractions; filters to strict $(0, 1)$; deduplicates.
- `sha256_of_ledger`: canonical hash by sorting reduced a/b lines and hashing the newline-joined string.

Why. Deterministic, content-addressable preregistration anchor that is platform agnostic.

4 MDL* Complexity and Nearest Fractions

Definitions. We define bit-length with conventions $\mathrm{bitlen}(0) = 1$, $\mathrm{bitlen}(1) = 0$, otherwise the usual. For a reduced a/b:

$$\mathrm{MDL}^*(a/b) = \mathrm{bitlen}(a) + \mathrm{bitlen}(b).$$

`nearest_fraction` finds the closest candidate fraction to p; ties are broken by smaller MDL^*. A cap `MAX_MDL_NEAREST` hides ultra-complex rationals in displays.

Why. MDL^* is a simple, data-independent complexity prior: smaller numerators/denominators are favored (Occam). It later induces spike weights $\propto 2^{-\mathrm{MDL}^*}$.

5 Dyadics and Blocks

Dyadic pools. ALL $= \{1/2^k : k \in [2, 16]\}$, and TINY $= \{1/2^k : k \in [k_{\min}, 16]\}$ with default $k_{\min} = 8$ (i.e., $\leq 1/256$). These define spike supports.

Blocks. Each block stores (n, k_X) for the XOR signal, and auxiliary A/B proportions for sanity checks; datasets map blocks to clusters.

Why dyadics. They are highly compressible, interpretable rationals and serve as canonical simple targets.

6 Wilson Intervals and CI-coverage

Wilson CI. We use the Wilson interval for a binomial proportion with guardrails (never negative under the square root; clamped to $[0, 1]$). For $z \in \{1.96, 2.24, 2.58\}$ we test coverage of pool values.

Why Wilson. Good coverage properties, smooth near boundaries, and standard for proportions.

7 Monte Carlo PB Tails (Blocks and Clusters)

Blocks. Let H be the observed number of blocks whose Wilson CI covers any pool value. Under the null, for each block draw $p \sim \mathrm{Beta}(a, b)$ (default $a = b = \frac{1}{2}$) and $k \sim \mathrm{Binom}(n, p)$; recompute $H^\star$. The PB tail is $\mathbb{P}(H^\star \geq H)$.

Clusters. Mark a dataset 1 if any of its blocks hits; sum over datasets to get C and compute $\mathbb{P}(C^\star \geq C)$ under the same null.

Why $\mathrm{Beta}(\frac{1}{2}, \frac{1}{2})$**.** Symmetric, Jeffreys-like slab; exchangeable across blocks; analytically convenient in Bayes factors.

Display floors. Monte Carlo zeros are floored to $\leq 1/\mathrm{sims}$; we report SE on the displayed p to avoid "± 0" outputs.

8 Bayes Factors: Spike vs Slab and Mixtures

Spike. A discrete prior over the pool with weights $w_i \propto 2^{-\mathrm{MDL}^*(p_i)}$; we precompute $\log p_i$ and $\log(1 - p_i)$.

Slab. $\mathrm{Beta}(\frac{1}{2}, \frac{1}{2})$.

Numerics. All likelihood sums and products are computed in the log domain using $\log\text{-}\sum\exp$ to avoid underflow. The choose-term cancels in BF ratios.

Mixture optimization. For spike+slab, we maximize mixture weight $\varepsilon \in (0, 1)$ via a stable ternary search over $[10^{-9}, 1 - 10^{-9}]$.

9 Ancillary Tests

A/B sign tests. Exact two-sided binomial tests for counts above/below 1/2 across blocks; sanity only.

Per-block p **for E-values.** If a block hits, we simulate a per-block p under the slab; values are floored to 1/sims before multiplication. An optional conservative early-exit (off by default) can short-circuit when hitting the smallest dyadic is provably impossible in a tiny-k scan.

Nearest-dyadic z **distances.** For each block's XOR rate $\hat{p} = k_X/n$, we find the nearest dyadic and compute $|\hat{p} - p_d|/\mathrm{SE}(\hat{p})$.

10 Reporting Sections (stdout)

1) **Candidates:** For each block, prints A/B nearest *registry* fraction (MDL*-aware) and XOR CI vs nearest *dyadic*.
2) **A/B sign:** Two-sided binomial tests around 1/2.
3) **PB tails:** For each z and pool (ALL, TINY), block and cluster PB tails with floors and SE; prereg primary (e.g., $z = 2.24$, TINY) flagged.
4) **FWER:** Monte Carlo family-wise exceedance over the prereg family.
5) **Nearest-dyadic z:** Standardized distances as an effect-size readout.
6) **Bayes:** Spike-only and spike+slab lnBF for ALL/TINY and the optimized ε^*.
7) **Predictive hold-out:** Tune ε^* on train datasets; evaluate mixture lnBF on the held-out dataset; emit notes when ε^* hits boundaries (0 or 1).
8) **E-values:** Observed hits, Tippett min-p, Simes p, product E-value.
9) T **statistic (exploratory):** $T = \sum |\hat{p} - p_d|/\text{SE}$; null distribution via MC. Note: excluding 1/2 from the dyadic set inflates null T near 0.5.
10) **Cluster LOO:** Cluster PB tails at the primary endpoint with one dataset dropped.
11) **Jackknife:** Block-level influence diagnostics at the primary endpoint.

11 Self-tests (Sanity Invariants)

Quick assertions catch regressions:

- Bit-length conventions and MDL* ordering (dyadic not more complex than 1/3).
- Wilson symmetry around 0.5.
- MDL*-aware nearest-fraction tie-breaks.
- Spike prior normalization (weights sum to 1 in the probability domain).

12 Run Block and Artifacts

The run block merges the ledger, prints file SHA256s and the preregistry hash, executes all sections, then writes artifacts:

- `config.json`, `environment.json`;
- `ledger/registry_fractions.txt`, `ledger/registry_hash.txt`, `ledger/file_hashes.json`;
- `sections/*.csv|json` for each section;
- `script_snapshot.py` (best-effort source capture);
- `README.md` summarizing the bundle.

13 Configuration Knobs

Important parameters:

- RNG seeds and simulation budgets (`SEED`, `MC_SIMS_*`);
- Prereg endpoints $\mathcal{Z} = \{1.96, 2.24, 2.58\}$; TINY threshold $k_{\min}$; display cap `MAX_MDL_NEAREST`;
- `LEDGER_PATHS` search order;
- Env flags: `ASCII_ONLY=1` (prints "¡=" instead of "≤"); `FAST_PVALS=1` (enables conservative early-exit in per-block p).

14 Interpretation Checklist

For reviewers:

- Primary evidence: $z = 2.24$ with TINY dyadics, blocks *and* clusters.
- Multiplicity: confirm FWER.
- Bayes: spike+slab lnBF with ε^*.
- Meta: min-p, Simes p, product E-value.
- Stability: dataset LOO and block jackknife.
- Repro: verify registry hash and per-file SHA256s.

15 Design Tradeoffs and Alternatives

- Wilson vs. Clopper–Pearson: Wilson is smoother with good coverage; CP is conservative and wider near extremes.
- Slab Beta$(\frac{1}{2}, \frac{1}{2})$ vs. Beta$(1, 1)$: symmetric Jeffreys-like prior is standard here; alternatives only slightly shift tails.
- Dyadic spike vs. broader rational pools: framework generalizes; dyadics chosen for interpretability and compressibility.
- MDL* choice: bit-lengths of numerator/denominator; a power-of-two "bonus" could strictly prefer dyadics, but we keep it neutral to avoid post-hoc bias.

16 Footnotes and Caveats

- Display floors: printing $\leq 1/\text{sims}$ prevents impossible zeros; SE is computed on the displayed p.
- T statistic: explicitly exploratory; excluding 1/2 from the dyadic set inflates null magnitudes for draws near 0.5.
- Source snapshot: may fall back to a placeholder if runtime introspection is limited.

17 Extending the Framework

Add new endpoints by following the pattern: (a) define an observed summary; (b) implement a null simulator; (c) export CSV/JSON; (d) document in the artifact README. Swap or augment spike pools via a new fraction list. Expose parameters through CLI (e.g., `argparse`) and optionally tee stdout to `stdout.txt`.

Contact. For questions or replication, see the repository README and the generated artifact README in each run folder.