

Other Sources of Brittleness in Eval Results

Which choices and conditions can change the conclusion?

The question for this lesson

If we change a reasonable evaluation choice, or repeat under comparable conditions, does the conclusion **survive**?

Judge noise and bias have their own lesson. Today: everything else that can change a conclusion.

Where brittleness enters

ITEMS 1–4

The question

Metrics, scoring rules and the decision we want to support

ITEMS 5–9

The evidence

Test cases, synthetic data, references and input presentation

ITEMS 10–16

The experiment

Configuration, execution, state and comparison conditions

ITEMS 17–18

The application

How far the result carries into real use and future conditions

Some changes reveal sensitivity. Others change the target, the system or the population.

The question

Metrics, scoring rules and the decision we want to support.

1 Metric choice

Same 200 answers, same judge, three questions:

If we ask about...	The winner is
The mean score	System A
The share rated 4 or above	System B
The worst case (5th percentile)	System B, clearly

The data does not change. The question does.

Playground: the Choice of Metric Lab

- 1 Open the Mean tab: System A wins.
- 2 Switch to the good rate and the worst case: System B wins.
- 3 Nothing in the data changed. Which question fits your product?

[Open the Choice of Metric Lab](#)

Mappings, weights and thresholds

2 Mapping and aggregation

Score mappings, weights and averaging per query or per user change the result.

3 Decision thresholds

A different definition of “good enough” can change which system passes.

Mappings and weights are utility assumptions. State them.

4 Goal mismatch

WHAT WE MEASURE

Tickets closed

Goes up

WHAT WE WANT

Problems resolved

Does not

A proxy can improve while the real goal does not.

How to handle measurement choices

Read a flip as a trade-off

A changed ranking is not automatically a defect

Say what you value

Mappings and weights are assumptions about utility

Decide before you look

Fix the decision rule and the guardrails first

Choose the question from the product objective, not from the results.

The evidence

Test cases, synthetic data, references and input presentation.

Dataset construction

5	Selection and filtering	Sources, exclusions, difficulty levels and date ranges define what gets tested.
6	Synthetic assumptions	The generator may create cleaner questions or miss real user behavior.
7	Shared origins	Variants of a few templates, documents or users dominate the evidence.

Many test rows can still represent only a few kinds of situation.

References and input presentation

- | | | |
|---|---------------------------|--|
| 8 | Reference quality | Incomplete, outdated or narrow labels can penalize valid answers. |
| 9 | Input presentation | Paraphrases, typos, context order and layout can change the system's answer. |
-

A consistent judge can faithfully score a flawed test.

How to check the evidence

Source	What to do
Synthetic data	Ask how it was generated, and for which claim
Shared origins	Resample or split by source, user, document or template
References	Audit them: consistent judging is not enough
Presentation probes	Keep the task's meaning before reading a change as sensitivity

Count independent situations, not rows.

The experiment

Configuration, execution, state and comparison conditions.

Variation in the system under test

10	Run-to-run variation	Different generations, retrieval results or agent paths change the output being judged.
11	Configuration	System prompts, examples, temperature and context limits alter the tested system.
12	Resource budgets	Timeouts, retries, tokens and tool-call limits change what it can achieve.

Stable judging does not imply stable product behavior.

State and external dependencies

- | | | |
|-----------|------------------------------|---|
| 13 | State and history | Conversation history, memory, caches and leftover actions affect the next case. |
| 14 | External dependencies | Search results, indexes, APIs, model versions and service load change between runs. |
-

The same code can run under different conditions.

Implementation and comparison design

- | | | |
|-----------|-----------------------------------|---|
| 15 | Evaluation implementation | Parsing, truncation, retries and dropped failures can change scores and denominators. |
| 16 | Boundaries and confounding | Prompts, retrieval, tools and budgets change together. Which change caused the gain? |
-

A product comparison does not isolate the contribution of each component.

How to check the experiment

Source	What to do
Run-to-run variation	Keep output generation separate from scoring
Configuration	Record a changed configuration as a different system
Resource budgets	Pick equal cost, equal time or deployed settings from the product question
State	Define reset boundaries between independent tests
The harness	Inspect parsing, truncation, joins and failure handling
Bundled changes	Hold other parts fixed to learn what caused a gain

The application

How far the result carries into real use and future conditions.

Transfer to real use and future conditions

- | | | |
|-----------|-----------------------------|--|
| 17 | Test-to-use mismatch | Single-turn tasks or simulated users miss important parts of real workflows. |
| 18 | Change over time | Users, tasks, policies and dependencies evolve. A valid result can become stale. |
-

Every result needs a boundary: where, when and for whom it applies.

When to re-evaluate

A valid baseline can become stale when any of these change:



Set re-evaluation triggers from the claim and its risk, not from a fixed calendar.

Reading a changed result

Not every difference is a failure.

What changed when the result changed?

What changed	For example	What it means
The question	Metric, weights or definition of success	A legitimate change of preference
The system or setting	Configuration, resources, population or environment	A different product or setting: record it
Nothing that should matter	Repeats under comparable conditions	Unstable measurement or behavior

The claim becomes brittle when it ignores the conditions it depends on.

Discussion: what changed?

- A** The new system wins on mean quality but loses on severe errors.
- B** The gain disappears when retries are switched off.
- C** The same evaluation, run again next week, picks the other winner.

Which kind of change is it? What must be fixed or documented?

From a list to an evaluation check

Understand

Which choices or conditions could change this conclusion?

Report

Which settings, failure rules and checks belong beside the score?

Manage

What should we fix, vary on purpose, version or re-evaluate?

[Open the Choice of Metric Lab](#)

Next: how the problems compound rather than cancel out.