05: How to tell if a model is good

Benchmarks, user impressions, and testing models on your own work

Why this matters

- Every model launch comes with a chart showing it winning.
- Most of those wins don't carry over to your repo or tool.
- Scores mislead in specific ways you can learn to spot.
- Choosing a model for a team needs evidence from your own work.

"Good" at what, for whom, at what cost?

Axis	The question
Capability	Can it do the task at all?
Reliability	Every time, or one time in three?
Fit	In *your* language, framework, repo size, and tool?
Cost and speed	Dollars, quota, and wall-clock time per finished task
Behavior	Stays in scope, admits uncertainty, writes reviewable code?

Most public benchmarks measure the first row. Daily work is mostly the other four.

A score describes a setup, not a model

```
  model      weights + version
+ harness    tools, system prompt, retries
+ settings   reasoning effort, token budget, temperature
+ task set   which version; contamination; label errors
+ grader     unit tests, rubric, LLM judge, humans
+ attempts   pass@1, pass@k, best-of-N
= the reported score
```

Change any line and the number moves.

Part 1

Reading other people's numbers

Public benchmarks wear out

Problem	Example
Contamination	Shown only the issue text, models named the file to fix up to 76% of the time
Saturation	Labs reported mid-90s on GPQA Diamond; an index dropped it, Sep 2026
Label errors	An estimated 6.49% of MMLU questions contain errors
Goodhart's law	"When a measure becomes a target, it ceases to be a good measure."

A frontier benchmark stays useful for a year or two. A good one is fresh, held-out, realistic, verified, and documents its harness.

Same model, same task: 99% or 51%

A model solves a task 80% of the time. Three independent attempts:

Metric	Meaning	Result
pass@3	At least one of 3 succeeds	$1 - 0.2^3 \approx 99\%$
pass^3	All 3 succeed	$0.8^3 \approx 51\%$

pass@k suits check-and-retry work. pass^k suits anything that must work every run.

The harness moves the score

Model	Vendor's own CLI	Neutral harness (Terminus-2)
GPT-5.5	24.7% (Codex)	21.2%
Gemini 3.1	12.2% (gemini-cli)	7.3%

Harbor-Index, Terminal-Bench team, June 2026. No gap reached statistical significance.

Compare models in the harness and settings you will actually use.

Reading a benchmark claim

"Model X scores 72% on Benchmark Y." Ask:

- Which benchmark version, and which harness?
- What settings: effort, token budget, number of attempts?
- Who ran it: the lab, the maintainers, or an evaluator?
- Were competitors run the same way?
- What's the error bar? A 2-point lead at ±2.6 is a tie.
- What does it cost per task, and what was left out?

Independent evaluators give a second opinion

Evaluator	What it does	Watch for
Artificial Analysis	Standardized runs; a composite index	Index changes every few months
Epoch AI	Its own runs, with error bars; trends	OpenAI funds FrontierMath (disclosed)
METR	Long-task time horizons; productivity RCTs	Close lab ties; few top-end tasks
Vals AI	Private and held-out domain evals	You can't inspect them
Scale Labs	SWE-bench Pro, with private splits	Also sells training data to labs
LiveBench	Regularly refreshed questions	Short, less agentic tasks
Arena	Human preference votes	Style bias; private testing

Independent doesn't mean neutral. Cross-check two or three, never one.

Part 2

Evidence closer to your work

Productivity studies: 55.8% faster to 19% slower

Study	Finding
Peng et al., 2023: one greenfield task	55.8% faster
Paradis et al., 2024: 96 Google engineers	~21% faster
Cui, Demirer et al., 2025: 4,867 developers	+26% completed tasks
METR, 2025: 16 maintainers, their own repos	19% slower; believed 20% faster
METR, Feb 2026: 57 developers	"Unreliable signal"; intervals include zero
METR on Anthropic, Nov 2026	Measured ~2x; self-reported ~12x

Self-reports run ahead of measurements. Treat surveys as a ceiling.

Build a golden-task set

A golden task is a real, solved piece of your work, frozen for replay.

```yaml
id: billing-proration-0142
base_commit: 3f9c2e1          # commit before the original fix
category: bugfix              # ... migration | review | trap
prompt: |
  Customers who upgrade plans mid-cycle are sometimes charged
  one cent too much. Find and fix the bug. Add a regression test.
grading:
  hidden_tests: evals/hidden/test_proration_0142.py
  must_not_touch: [migrations/, src/api/public/]
```

Score with a rubric, blind

Criterion	Points
Hard fail (edited tests, false success claim, forbidden path, destructive command)	Score is 0
Correctness: hidden tests pass	0–4
Scope discipline: about as small as the reference	0–2
Code quality: a reviewer would approve it	0–2
Tests: a meaningful regression test	0–1
Communication: accurate summary, real uncertainty flagged	0–1

Strip model names and shuffle before anyone scores. Unblind after.

Small evals are noisy

Standard error of a pass rate $\approx \sqrt{(p(1-p)/n)}$

- 20 tasks at 60%: ±11 points, so ±22 at 95% confidence.
- 55% vs 65% on 20 tasks: you can't tell them apart.
- Pair the comparison: same tasks, count only the disagreements.
- If A wins 9 of 10 disagreements, that means something.
- Repeat runs to separate randomness from task difficulty.

Without an error bar, you can't tell how much a score means.

Keep the eval honest

- New model or version? Rerun the whole set before switching.
- Harness update? Rerun a five-task smoke subset.
- Pin versions and dated model snapshots, not "latest."
- Retire tasks whose fixes are now public.
- Hide tests, diff the test directory, read passing transcripts.
- Keep the eval in the repo and review changes like code.

Part 3

When benchmarks and users disagree

Benchmark-maxing runs from defensible to dishonest

Optimizing for scores at the expense of what users experience:

1. Training on benchmark-shaped data
2. Contamination, where test items leak into training
3. Tuning a variant to please one grader or audience
4. Reward hacking that shows up in behavior
5. Cherry-picking the harness, settings, version, or benchmarks
6. Unlabeled best-of-N; publishing only the best private variant

From Opus 5 to Opus 5.5

When (2026)	What happened
Jul 24	Opus 5 ships with strong numbers and leads the Artificial Analysis index
Late Jul	Reception is mixed: verbose, over-engineers, spins at high effort
System card	Blames some weak results on "unproductive self-verification"
Sep 22	Opus 5.5: "major improvements to the way Opus 5.5 writes and communicates"
Sep 26	Zvi Mowshowitz: "the most consistently positive set of reactions I have ever seen"

Maxed on what's measured, with no cheating needed

- No public evidence shows Opus 5 trained on benchmark answers.
- One researcher alleged genre-specific training. That's a single claim.
- The benchmarks scored task completion.
- Users also cared about communication, scope, and judgment.
- Benchmarks barely score those. A few newer ones do.

Vibes are data. Collect them on purpose.

1. Keep a log: date, model, harness version, task, result.
2. Decide what you're testing before you look.
3. Blind the comparison by stripping the model names.
4. Give both models the same tasks.
5. Wait out the honeymoon and re-judge after two weeks.
6. Trust specific, repeated reports, like "over-engineers small fixes."

Bryan's take

> **Bryan's take:** Opus 5 felt benchmark-maxed to me. It posted great numbers and kind of sucked in daily use, and people didn't like it. Opus 5.5 is the one people really liked.

Try it

Individual (2–3 hours)

- Turn 10 merged PRs into task files with hidden tests.
- Run two configurations three times each; score blind with the rubric.
- Write five sentences: winner, margin, task types, cost, error bar.

Class or team (one session plus homework)

- Each person contributes two golden tasks from real work.
- Each group runs one configuration; a separate group grades blind.
- Compare your ranking with the public leaderboard's.

Discussion

1. A vendor beats a rival by 3 points on a coding benchmark. What do you ask first?
2. pass@k or pass^k: which fits autocomplete, a PR-writing agent, a nightly job?
3. METR's developers felt faster but were slower. How would your team guard against that?
4. Your eval favors Model A; your senior engineers' vibes favor Model B. Now what?

Read more

- Module: 05 How to tell if a model is good
- Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity (Becker et al., 2025)
- Harbor-Index (Terminal-Bench team, June 2026)
- Adding Error Bars to Evals (Miller, 2024)