Skill registry · 2026-05-27

A register of every custom slash-command skill I have written for Claude Code, with measurement when I have it and an honest guess when I do not. 4 repos, 33 skills, 33 A/B-tested.

A/B-MEASURED SAVE RATES BY PORTFOLIO

+23% +78K tokens across 8 skills

AudiobookMaker (arm A 342K → arm B 264K)

+21% +27K tokens across 4 skills

mikkonumminen.dev (arm A 128K → arm B 101K)

+20% +170K tokens across 11 skills

Spacepotatis (arm A 853K → arm B 683K)

+9% +52K tokens across 10 skills

claude-skills (arm A 574K → arm B 523K)

Aggregate: +17% (+327K tokens across 33 skills). Save rate = (arm A − arm B) / arm A, summed per portfolio. Negative means the skill costs MORE per use than going cold, those skills encode rigor (audit thoroughness, protocol discipline, spec depth), not scout-savings.

> *This document separates measured numbers from estimated ones, shows the cases where a skill cost MORE than no skill at all, and claims no more than the data supports.*

HOW TO READ THIS

- **Per use, not annual.** Every number on the row is per one invocation of the skill. Multiply by uses-per-year yourself if you want to extrapolate.
- **Measured (green) vs estimated (italic gray).** Green numbers come from real runs: production transcripts for cost, A/B calibrations for save. Italic gray numbers are the author's pre-measurement guesses.
- **Cost and save are different regimes.** Cost is from production transcripts; save is from controlled A/B calibrations on representative tasks. They often live at different scales. Do not divide save by cost to get an "efficiency %": the percentages shown are anchored to the A/B baseline, not the production cost.
- **N=1 for most A/B saves.** One sub-agent ran the task with the skill, one without. Trust direction and rough magnitude, not two-significant-digit precision. The /review row is the exception: N=11 across four PR-size buckets.
- **Negative save (orange) is a real finding**, not a bug. The skill spent MORE tokens than the unstructured baseline because it encodes work the cold arm skipped (audit thoroughness, protocol discipline). The value is completeness, not compression.
- **Estimated save column is intentionally absent.** It would mechanically be 2× estimated cost (the 3× heuristic this project started with); zero independent information. The estimated cost column carries the actual author guesses.

Across-model pattern. 33 of these skills were A/B-tested on more than one model, as was 1 built-in command. The save rate is what changes: skills that save 50%+ on Sonnet typically settle at 20–40% on Opus. The skill arm does not get more expensive: the cold arm gets cheaper, because a stronger model needs less scaffolding to do the task well. A skill's measured value is not a fixed property; it is relative to the model underneath it. See the per-model save columns below.

Per-skill registry

*Every figure per single invocation. **Cost / use** is from production transcripts (model tagged on each row); **Est. cost** is the author's pre-measurement guess; **Save: X** is an A/B calibration on model X (arm A cold − arm B with-skill). Dash means no data exists for that cell yet. A † on a save cell flags a procedure deviation: see the appendix.*

SKILL	STATUS	COST / USE	EST. COST	SAVE: SONNET	SAVE: OPUS	SAVE: HAIKU
Claude Code built-in (reference) · not part of the custom portfolio						
/review CLAUDE CODE BUILT-IN (REFERENCE) Claude Code built-in PR code review	✓MEASURED last 2026-05-27	**10K** tokens / use *Opus production · median of 339, mean 48K, range 1K–981K, σ 134K*	–	**+44%** +22K	**+34%** +18K	**+52%** +30K

SKILL	STATUS	COST / USE	EST. COST	SAVE: SONNET	SAVE: OPUS	SAVE: HAIKU
AudiobookMaker · github.com/MikkoNumminen/AudiobookMaker · 8 skills						
ci-failure-triage When CI fails on a release tag or PR, walk the failure modes in the right order against a recipe library built from the project's fix(ci) commit history.	✓MEASURED last 2026-05-23	**25K** tokens / use *A/B arm-B*	*1K* *tokens / use*	**+43%** +19K	**+10%** +3K	**+14%** +6K
copyright-scan Scan a git diff (staged by default, or a named revision range / file set) for accidental third-party copyright leaks before they land on origin.	✓MEASURED last 2026-05-12	**14K** tokens / use *Opus production*	*3K* *tokens / use*	**+49%** +32K	**−20%** −9K	**+57%** +35K
pronunciation-corpus-add Append a Finnish pronunciation failure (a word the Chatterbox Grandmom voice mispronounces) to the project's pronunciation corpus at docs…	✓MEASURED last 2026-05-23	**17K** tokens / use *A/B arm-B*	–	**+12%** +2K	**+39%** +17K	**+7%** +2K
release-bundle-audit Audit the AudiobookMaker PyInstaller release bundle for unused dependencies, dead-code data files, and accidental ML-stack pollution; then propose spec-only fixes on a `chore/release-bundle-size` branch.	✓MEASURED last 2026-05-23	**87K** tokens / use *A/B arm-B*	*3K* *tokens / use*	**−12%** −9K	**−1%** −546	**+20%** +11K
release-cut Cut a new AudiobookMaker release (bump APP_VERSION, tag vX.Y.Z, push, verify CI lands SHA-256 in release notes AND sidecar). Use this ski…	✓MEASURED last 2026-05-12	**32K** tokens / use *Opus production · median of 6, mean 31K, range 18K–49K, σ 10K*	–	**+17%** +6K	**+48%** +31K	**+29%** +16K
voice-pack-finnish Build a Finnish voice pack from a source audio file end-to-end: chunked analyze with ECAPA diarization, transcript validation per speaker, branching to LoRA training (long sources) or few-shot ref-clip packaging (short sources), and ear-check by synth.	✓MEASURED last 2026-05-23	**27K** tokens / use *A/B arm-B*	*7K* *tokens / use*	**+52%** +29K	**+20%** +9K	**+32%** +15K
work-session Start, pause, or finish a TODO.md work session as one of the 4 permanent Claude sessions in AudiobookMaker. Use this whenever the user sa…	✓MEASURED last 2026-05-23	**23K** tokens / use *A/B arm-B*	*2K* *tokens / use*	**−10%** −2K	**−10%** −3K	**−12%** −3K
worktree-launch Start a new parallel Claude session safely.	✓MEASURED last 2026-05-23	**23K** tokens / use *A/B arm-B*	*800* *tokens / use*	**+1%** +126	**+24%** +8K	**+23%** +9K

SKILL	STATUS	COST / USE	EST. COST	SAVE: SONNET	SAVE: OPUS	SAVE: HAIKU
Spacepotatis · github.com/MikkoNumminen/Spacepotatis · 11 skills · 1 redirect						
balance-review Diff uncommitted changes to game data (weapons, enemies, waves, missions, perks, augments, loot pools, solar systems) and report DPS, TTK, energy-cost-per-DPS, augment-folded effective DPS, loot-pool roster shifts, and drop-rate deltas vs HEAD.	✓MEASURED last 2026-05-22	**33K** tokens / use *A/B arm-B*	*14K* *tokens / use*	**+28%** +13K	**−12%** −4K	**+5%** +2K
content-audit Pre-commit content invariants check: orphan refs (enemy / weapon / sprite / pod / loot-pool / mission system / story trigger), missing sprite generators, perk drop-weight sanity, mission prereq DAG, story integrity (voice + music files, trigger refs), storyTriggers helper coverage.	✓MEASURED last 2026-05-03	**53K** tokens / use *Opus production*	*15K* *tokens / use*	**+22%** +22K	**−25%** −20K	**−2%** −912
equipment Create, modify, or remove a weapon or piece of equipment (augment / reactor / shield / armor), including visual changes to how it appears in combat or in the loadout UI.	✓MEASURED last 2026-05-03	**157K** tokens / use *Opus production · median of 3, mean 268K, range 27K–619K, σ 254K*	*4K* *tokens / use*	**−5%** −5K	**+48%** +41K	**−3%** −1K
modular-architecture-audit Orchestrates the multi-phase modular-architecture audit + refactor.	✓MEASURED last 2026-05-22	**129K** tokens / use *A/B arm-B*	*5K* *tokens / use*	**+26%** +46K	**−4%** −4K	**+29%** +13K
new-enemy Scaffold a new enemy: enemies.json entry, BootScene placeholder sprite generator, optional test wave, and integrity test verification.	✓MEASURED last 2026-05-22	**68K** tokens / use *A/B arm-B*	*6K* *tokens / use*	**+9%** +7K	**+70%** +82K	**+57%** +35K
new-migration Add a Postgres schema migration end-to-end: dated SQL file in db/migrations/, Database interface update in src/lib/db.ts, prod applicatio…	✓MEASURED last 2026-05-22	**21K** tokens / use *A/B arm-B*	*7K* *tokens / use*	**+16%** +4K	**+36%** +19K	**−33%** −11K
new-mission Scaffold a new combat mission across missions.json, waves.json, the galaxy planet binding, and a smoke test: in one shot.	✓MEASURED last 2026-05-22	**50K** tokens / use *A/B arm-B*	*8K* *tokens / use*	**+37%** +29K	**+15%** +8K	**+19%** +8K

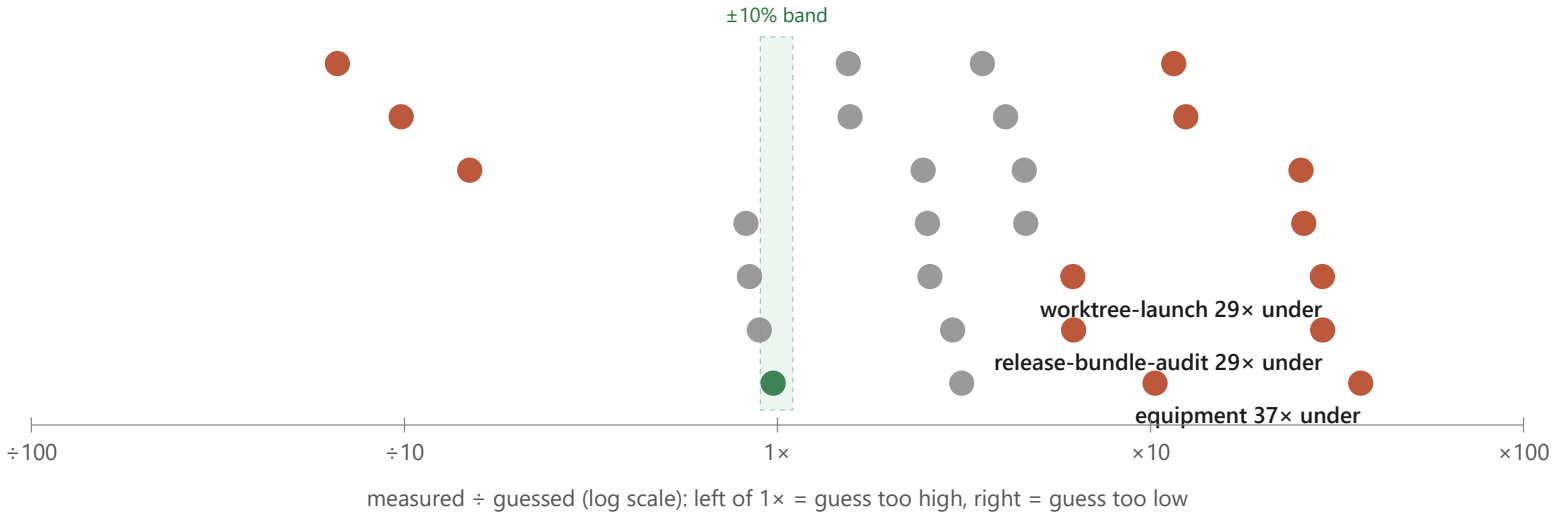
SKILL	STATUS	COST / USE	EST. COST	SAVE: SONNET	SAVE: OPUS	SAVE: HAIKU
new-perk Scaffold a new mission-only perk: perks.ts entry, BootScene icon generator, HUD chip wiring, and (for actives) a switch case in PerkContr…	✓MEASURED last 2026-05-22	**56K** tokens / use *A/B arm-B*	*9K* *tokens / use*	**+32%** +27K	**+25%** +12K	**+30%** +16K
new-solar-system Add a new solar system to the galaxy: solarSystems.json entry (incl. galaxy bed), SolarSystemId union extension, optional unlock gating, …	✓MEASURED last 2026-05-22	**60K** tokens / use *A/B arm-B*	*13K* *tokens / use*	**+9%** +6K	**+73%** +93K	**+16%** +8K
new-story Add, modify, or remove story content: cinematic popups, voiceovers, music beds, body text, written synopses, and auto-trigger wiring.	✓MEASURED last 2026-04-30	**56K** tokens / use *Opus production*	*5K* *tokens / use*	**+29%** +14K	**+34%** +29K	**+28%** +16K
new-weapon *Superseded by /equipment.*	REDIRECT	–	–	–	–	–
save-roundtrip-audit Pre-commit save-pipeline invariants check: walks every StateSnapshot field through the 8 layers of the save round-trip (snapshot interfac…	✓MEASURED last 2026-05-03	**19K** tokens / use *Opus production*	*12K* *tokens / use*	**+10%** +7K	**+15%** +13K	**+16%** +11K
claude-skills · github.com/MikkoNumminen/claude-skills · 10 skills						
ai-codegen-smell-audit Read-only audit for specific failure modes that recur in LLM-generated code: defensive guards on impossible cases, generic names in domain code, swallowed errors, single-use helpers, mirror tests, and so on.	✓MEASURED last 2026-05-17	**7K** tokens / use *Haiku production · median of 4, mean 45K, range 2K–165K, σ 69K*	*75K* *tokens / use*	**+48%** +76K	**+10%** +11K	**+1%** +496
audit Run a multi-phase robustness audit of the current codebase.	✓MEASURED last 2026-05-20	**28K** tokens / use *Opus production · median of 6, mean 153K, range 4K–670K, σ 239K*	*425K* *tokens / use*	**+2%** +3K	**−7%** −15K	**+23%** +21K
mikko-audit-suite Run every `mikko-*` audit skill that's relevant to the current codebase, in a sensible order, and produce one index document linking to each audit's report.	✓MEASURED last 2026-05-23	**24K** tokens / use *A/B arm-B*	–	**−21%** −4K	**+5%** +2K	**+8%** +3K

SKILL	STATUS	COST / USE	EST. COST	SAVE: SONNET	SAVE: OPUS	SAVE: HAIKU
mikko-help List every installed `mikko-*` skill with its description (or with its `barney:` field when `--barney` is passed).	✓MEASURED last 2026-05-20	**17K** tokens / use *Opus production · median of 9, mean 143K, range 8K–528K, σ 185K*	*7K* *tokens / use*	**−22%** −5K	**−21%** −7K	**−62%** −12K
mikko-install Install, update, list, or uninstall `mikko-*` skills from the cloned `claude-skills` source repo into the user-wide skill directory (`~/....	✓MEASURED last 2026-05-21	**12K** tokens / use *Opus production · median of 3, mean 16K, range 12K–23K, σ 5K*	*4K* *tokens / use*	**−35%** −6K	**−2%** −503	**−14%** −3K
react-anti-patterns-audit Read-only audit for six concrete React anti-patterns that recur in component code (key=index in lists, useEffect for derived state, dep-array lies, state mutation instead of replace, missing effect cleanup, multiple sources of truth).	✓MEASURED last 2026-05-23	**22K** tokens / use *A/B arm-B*	*23K* *tokens / use*	**−33%** −5K	**−28%** −7K	**−21%** −4K
readme-drift-sync Detect drift between what the repo actually contains and what `README.md` claims, then rewrite ONLY the drifted sections in the README's …	✓MEASURED last 2026-05-23	**63K** tokens / use *A/B arm-B*	*40K* *tokens / use*	**−112%** −33K	**−28%** −12K	**−6%** −3K
security-audit Orchestrates the multi-phase security audit + remediation.	✓MEASURED last 2026-05-22	**112K** tokens / use *A/B arm-B*	*750K* *tokens / use*	**−4%** −4K	**−8%** −10K	**+3%** +2K
session-cost Measure the token cost of a single Claude Code session: main thread plus every sub-agent it dispatched.	✓MEASURED last 2026-05-21	**409K** tokens / use *Opus production*	*0* *tokens / use*	**+14%** +5K	**+4%** +1K	**−89%** −26K
skill-usage Measure actual Claude Code skill usage from local transcript JSONL files.	✓MEASURED last 2026-05-21	**10K** tokens / use *Opus production · median of 5, mean 15K, range 6K–39K, σ 12K*	*4K* *tokens / use*	**+53%** +26K	**−16%** −6K	**+57%** +36K

SKILL	STATUS	COST / USE	EST. COST	SAVE: SONNET	SAVE: OPUS	SAVE: HAIKU
mikkonumminen.dev · github.com/MikkoNumminen/mikkonumminen.dev · 4 skills						
md-to-pdf Render a markdown report (or any HTML) to a styled PDF using the local Chrome's `--print-to-pdf` flag.	✓MEASURED last 2026-05-23	**21K** tokens / use *A/B arm-B*	*25K* *tokens / use*	**+41%** +15K	**+22%** +8K	**+54%** +32K
skill-localUpdate Refresh every local artifact the site renders about your portfolio skills: the JSON the /contact terminal fetches at runtime, the measurement-overlaid annual totals, and the downloadable skills-registry PDF, in one sequenced chain.	✓MEASURED last 2026-05-23	**21K** tokens / use *A/B arm-B*	–	**+52%** +23K	**+39%** +19K	**+33%** +15K
skill-registry Scan every sibling repo in the workspace for `.claude/skills/*/SKILL.md` files and emit a consolidated registry as a structured JSON docu…	✓MEASURED last 2026-05-20	**116K** tokens / use *Sonnet production · median of 4, mean 111K, range 83K–128K, σ 17K*	*130K* *tokens / use*	**−38%** −6K	**−39%** −9K	**−32%** −6K
sync-readmes Audit project data against sibling repos' READMEs and open a PR with drift corrections.	✓MEASURED last 2026-05-18	**119K** tokens / use *Opus production*	*141K* *tokens / use*	**−11%** −4K	**−1%** −680	**−24%** −11K

Calibration honesty: where my guesses landed

Each dot is one skill: position on the x-axis shows how wrong the guess was (measured ÷ guessed). **Green dots** sit inside the ±10% band: the guess was effectively right (1 skill). **Orange dots** missed by 5× or more (13 of 28). **Gray dots** missed by less than 5×: between the ±10% band and the 5× threshold (14 of 28). **21 of 28 guesses were too low**. The fix is not "guess better next time": intuition about token cost is unreliable in a way better intuition will not fix. The fix is to keep measuring.



Per-skill guess and measured numbers are in the spine table above (Est. cost and Cost / use columns). This chart shows the ratio between them. 28 skills total: every measured row that has a prior editorial estimate on record.

Findings

Estimates were systematically wrong, and systematically low.

Before measuring anything, I wrote a guess for what each skill would cost. Of 28 skills with a guess on record, 13 were off by 5× or more, and the misses ran almost entirely one direction: I guessed too low. The takeaway is not "guess better next time." Intuition about token cost is unreliable in a way better intuition will not fix. The takeaway is to measure, and to keep measuring.

A recipe's value shrinks as the model gets more capable.

The same skill, A/B-tested on Sonnet, Opus, and Haiku, does not save the same amount on each. Skills that save 50%+ on Sonnet typically settle at 20–40% on Opus. The skill arm does not get more expensive: the cold arm gets cheaper, because a stronger model needs less scaffolding to do the task well. A skill's measured value is not a fixed property; it is relative to the model underneath it.

Some skills cost more than no skill at all, and that is the point.

Several skills show a negative save: the run that followed the skill spent more tokens than the run that went in cold. These are not failures. They are audit and lifecycle skills that do thorough work the unstructured run simply skipped. Their value is completeness and discipline, not token compression. A registry that only celebrated savings would have to hide them; this one counts them.

The measurement tooling itself had a 23× error.

The parser that produced the cost numbers grouped a whole Claude Code session as one "use." A single session can contain dozens of calls to the same skill, so for /review it reported the cost of 336 invocations divided by 14, overstating per-use cost roughly 23×. A carefully collected number from a faulty instrument is still a wrong number. The measurement infrastructure needs the same scrutiny as the thing it measures, and catching this cost only attention, not compute: re-parsing transcripts is free.

Cost and save are measured in different regimes; do not divide them.

Cost per use comes from real production transcripts. Save per use comes from controlled A/B calibrations on representative tasks. They measure related but different things, often at different scales, so dividing save by cost to get an "efficiency %" produces a number that means nothing. The percentages here are anchored to the A/B baseline, stated explicitly on each row where the gap is large. Two real numbers can still be the wrong pair to compare.

Cost distributions are heavily skewed; the median and the mean tell different stories.

A skill's cost per use is not a single number. For /review it ranges from about 1K to nearly 1M tokens, because cost tracks the size of the work, a one-line PR and a 4,000-line PR are not the same job. A few large runs pull the mean to 48K, while the typical run: the median, costs 10K. This is not a defect to fix; it is a true property of the data. The honest response is to headline the median and show the spread, not to pretend one average describes every use.

The measured aggregate save is far below the heuristic that started this.

This project began with a rule of thumb: a skill saves about twice what it costs, a ~67% saving. Measurement put the portfolio aggregate at about 17%. The heuristic was roughly three times too optimistic. That gap, more than any single skill's number, is why the document exists: the way to know what tooling is worth is to measure it, not to model it.

> **WHAT THIS DOCUMENT DOES NOT CLAIM**
>
> - *No production cost numbers. This is development-time tooling on my own machine.*
> - *No comparison to other engineers' setups. I only have transcripts for one engineer.*
> - *No per-feature ROI claim. This is per-skill cost, not per-feature value.*
> - *No assertion that any of this scales linearly to a team. It might. I have not measured.*
> - *No claim that the modeled savings would survive an actual A/B test against well-prompted unstructured chats. They might; I have not tested.*

Appendix: methodology

How the numbers in the spine table are produced.

Per single use: the only unit

Every number is per one invocation of the skill. No annual figures, no "tokens per year." If you want a yearly figure, multiply by however many times you will actually run the skill.

How "measured cost / use" is produced

Two sources, both real. **Transcript measurement**: every Claude Code session writes a JSON-Lines transcript to `~/.claude/projects/<dir>/<sessionId>.jsonl`, with each assistant message carrying an `attributionSkill` field when a skill is active. `scripts/build-review-stats.mjs` walks those files, groups by `(skill, sessionId, promptId)` (one user message = one invocation), sums `input_tokens + output_tokens + cache_creation_input_tokens` (cache reads excluded: paid upstream), dedupes by `requestId`, and reports per-use stats across whatever invocations landed in the 90-day window. The dominant model that spent tokens on the skill becomes its `cost_model` tag.

A/B calibration arm-B: when a skill has no production transcripts, an A/B run still spends real tokens, a sub-agent followed the `SKILL.md` end-to-end on a representative task. Arm-B IS a real cost-per-use measurement, just from a controlled run instead of in-the-wild use.

What counts as one "use": the invocation-boundary correction

The upstream `skill-usage` parser groups assistant messages by `(skill, sessionId)`: every message in one session counts as part of one invocation. That overstates tokens-per-use whenever a session contains multiple uses of the same skill. For `/review` the distortion is 23×: 14 sessions contained 336 distinct calls, so the parser's "1.15M / use" was really "cost of 336 uses divided by 14 instead of 336." `build-review-stats.mjs` corrects this by walking the `parentUuid` → originating-user-message chain on every transcript-measured row and using each user message's `promptId` as the invocation ID.

Median, not mean, on the cost headline

Where a row's cost is the average of multiple invocations, the headline is the **median**: the cost distribution is heavily right-skewed (one or two large invocations pull the mean well above the typical use), so the median is the honest "what does one use cost" figure. The cost cell tags "median of N" when applicable.

How "measured save / use" is produced

A calibration A/B test. Two sub-agents solve the same task in fresh sandboxed worktrees: arm A cold (no `SKILL.md` access), arm B following the skill. Save = arm-A tokens − arm-B tokens. **N = 1 per skill, single data point** on Sonnet primary; the multi-model columns repeat the A/B with a sub-agent dispatched as Opus or Haiku. Trust direction and magnitude, not two-significant-digit precision.

Exception: `/review`. N=11 A/Bs bucketed by PR size. Per-bucket medians: small (0–199 lines) 44%, medium (200–799) 26%, large (800–2499) 15%, extra-large (2500+) −10%. 63% of production /review calls are on small PRs, so the row's headline save uses the small-bucket median; the across-bucket aggregate (35%, 17K saved) sits in `calibration_aggregate_*` on the JSON. Per-bucket and per-PR data live on `calibration_ab_buckets` + `calibration_ab_runs`.

Different regimes: do not divide save by cost

Cost is from production transcripts; save is from A/B calibrations on (often smaller) representative tasks. The percentages shown on save cells are `save ÷ A/B arm A`, not `save ÷ production cost`. Several rows have an explicit scale-ratio hint (e.g. "A/B scale ~5× production" or "production scale ~11× A/B") when the two regimes diverge by 2× or more.

How "estimated cost / use" is produced

The number an author wrote down before any measurement existed, parsed from the skill's `SKILL.md` frontmatter or the repo's `README.md`. There is no math behind these. They are intuition snapshots from the moment the skill was scaffolded. See the calibration honesty chart for how reliable that intuition turned out to be.

Reproducibility

From inside `mikkonumminen.dev/` on a machine with Claude Code installed:

1. `/mikko-skill-usage`: writes `.claude/agent-verdicts/SKILL-USAGE-LATEST.json` from local transcripts.
2. `/skill-localUpdate`: re-walks the portfolio inventory, applies the measurement overlay, re-runs `build-review-stats.mjs`, renders this PDF.

The data the renderer reads is committed at `public/data/skills-registry.json`. The dated history lives in `.claude/agent-verdicts/SKILL-REGISTRY-*.json`. The transcript files themselves stay on the author's machine because they contain code and context from private repos.
