Do LLM-Authored Agent Skills Help? A Component-Level Ablation in Data Analysis Tasks

Wei-Jung Huang*
Independent Researcher
United States
william.wj.huang@gmail.com

Abstract

Agent Skills, structured `SKILL.md` packages that bundle procedures, examples, and references, have become the primary mechanism for injecting domain knowledge into LLM-based agents. LLM-generated skills are cheaper and more scalable than human-curated ones, but prior evaluations have found no aggregate benefit from them. This raises a natural question: does the aggregate null conceal useful structure at the component level, or are LLM-authored skills genuinely unhelpful?

We investigate this question by decomposing LLM-authored skills into their individual sections (procedures, examples, and reference notes) and conducting an ablation study across four data-analysis task families, 56 tasks, nine models, and three providers (2,520 evaluations). At the aggregate level, none of the skill conditions significantly outperforms the No-Skill baseline (all $p>0.58$, mixed model); the total spread across all five conditions is 1.2 pp (65.5% to 66.7%). Stratifying by model capability reveals no significant interaction: all bootstrap confidence intervals include zero, and paired McNemar tests confirm that skills help and hurt on roughly equal numbers of task×model pairs. These results suggest that, in data analysis, single-shot LLM-authored skills provide no measurable benefit, whether baselines are high or low, regardless of which sections are included, which model is used, or how capable that model is.

Keywords: Agent Skills, SKILL.md, ablation study, prompt engineering, LLM evaluation, data analysis

1 Introduction

The `SKILL.md` format has emerged as a cross-platform standard for encoding reusable domain knowledge in LLM-based agents. Numerous agent platforms, including Claude Code, Gemini CLI, and Codex, now support Skill ingestion at inference time [7]. While human-curated, task-specific Skills can improve pass rates substantially (+16.2 pp in SkillsBench [7]), LLM-generated skills, authored by language models rather than domain experts, have shown no aggregate benefit [7]. Skill effects also vary widely across domains, from +51.9 pp in healthcare to +4.5 pp in software engineering [7]. This domain dependence raises a question: in domains where baseline performance is already moderate to high, do structured skills add any value at all?

LLM-authored skills are the focus of this study, for three reasons. First, they are increasingly prevalent: several agent platforms now auto-generate skills from user interactions, making LLM authoring the default path for many practitioners. Second, they are far cheaper and more scalable than manual expert curation. Third, and most importantly, the absence of an *aggregate* benefit does not mean every section is unhelpful. If some sections contribute positively while others introduce noise that cancels the benefit, targeted pruning could rescue LLM-authored skills without sacrificing their scalability.

This possibility motivates two questions. First, **when an LLM-authored Skill helps or hurts, which section is responsible?** Anthropic's Agent Skills documentation [1] organizes skill content into three components: discovery metadata (when to activate the skill), procedural instructions with examples (how to execute the task), and on-demand reference materials (supplementary heuristics and conventions). We adopt these content categories as our ablation factors, decomposing each skill into four sections that mirror this hierarchy: routing triggers (corresponding to Anthropic's discovery metadata), core procedures, worked examples, and reference notes. Evaluating the skill as a whole obscures which sections help and which cause harm; removing sections one at a time can disentangle these effects.

Second, **do these patterns hold across different domains, starting with data analysis?** A rigorous ablation study requires a domain where task outputs are deterministic and can be verified automatically, ruling out subjective or open-ended tasks. Data analysis satisfies this requirement: SQL queries return exact result sets, statistical computations produce numeric values checkable within tolerance, and structured reports conform to verifiable schemas. Data analysis is also relatively under-explored for skill evaluation compared to software engineering [7], yet it is one of the most common LLM use cases.

We address this gap with a component-level ablation of LLM-authored Skills in the data-analysis domain. Each of our four skills covers an entire task family (e.g., all data-cleaning tasks rather than a single task instance), mirroring how practitioners typically author skills. We systematically remove individual sections and measure each section's marginal

*This work was conducted independently and does not represent the views of the author's employer.

contribution across 56 tasks and nine model configurations spanning three providers (OpenAI, Google, Anthropic). Our contributions are:

1. **A within-task paired ablation design.** Each of the 56 tasks is evaluated under five conditions (No-Skill, Full Skill, and three partial-skill variants), so every task serves as its own control. This design isolates the effect of individual skill sections while absorbing task-level difficulty variation, across nine models from three providers (7,560 total runs).
2. **A robust null result across all skill conditions.** No skill condition outperforms the No-Skill baseline (all $p>0.58$, mixed model). The aggregate spread is 1.2 pp (65.5% to 66.7%), and paired McNemar tests show that skills help and hurt on roughly equal numbers of task×model pairs.
3. **Evidence that skill effects do not vary by model capability.** Stratified bootstrap analyses show no significant interaction between skill condition and model capability. All confidence intervals include zero for compact, frontier, and reasoning models, suggesting that the null result is uniform across the capability spectrum.

2 Related Work

The `SKILL.md` specification formalizes the longstanding practice of adding domain instructions to LLM prompts [1]. Before standardization, similar techniques appeared under various names, including system prompts [10], retrieval-augmented generation [6], and few-shot prompting [2]. A related line of work examines how individual prompt elements affect LLM behavior: Min et al. [9] showed that label correctness in demonstrations matters far less than the format and input distribution, and Lu et al. [8] demonstrated that example ordering significantly impacts performance. Our work extends this direction from ad-hoc prompt elements to *structured knowledge packages*.

SkillsBench [7] provided the first large-scale evaluation of Skill efficacy across 86 tasks and 7,308 trajectories, finding that focused skills with two to three modules outperform comprehensive documentation and that LLM-generated skills provide no aggregate benefit. Our study differs by examining family-level skills in a single domain and systematically varying which sections are included. SkillLearnBench [13] subsequently benchmarked continual skill-learning methods, finding that all methods improve over the no-skill baseline but no single approach dominates across tasks and LLMs. SkVM [3] takes a complementary approach, treating skills as compiled code and proposing capability-based compilation across heterogeneous LLM backends.

On the benchmark side, DataSciBench [12] and DS-1000 [5] evaluate LLM code generation for data-analysis tasks and demonstrate that frontier models already achieve strong baseline performance on many data-analysis tasks. We use this observation to motivate our investigation: in task families where models already perform well, the marginal value of skill content may be near zero. While these benchmarks evaluate LLM capability on data-analysis tasks, none study whether structured Agent Skills improve performance in this domain. Our work addresses this gap.

3 Experimental Design

3.1 Skill Construction

We generated one skill per family (Data Cleaning, SQL Query, Experiment Analysis, and Structured Reporting; four skills total) using an LLM coding assistant (Gemini 2.5 Pro with extended thinking enabled). For each family, the assistant was given a prompt specifying the target domain (e.g., "data cleaning"), the four required sections, and the `SKILL.md` schema from Anthropic's documentation [1]. Each skill was produced in a single generation with no subsequent human editing, selection from multiple candidates, or iterative refinement. The ablation variants (Core-Only, Core+Examples, Core+Refs) were then produced by mechanically deleting sections from the Full skill, not by separate generation runs. We adopted this minimal-curation approach because it reflects a common and highly scalable workflow for skill creation: an LLM produces the content in one pass based on a domain description, and the output is used as-is. Using a frontier reasoning model for generation means our skills likely represent a quality ceiling for auto-generated content, so the structural issues we identify (e.g., generic Reference Notes conflicting with task instructions) are unlikely to be artifacts of a weak generator. Because our ablation tests which *sections* help or hurt rather than the absolute quality of any particular skill, the choice of generator model is unlikely to affect the directional findings, provided the output follows the same four-section schema.

Each skill targets a distinct data-analysis workflow: **Data Cleaning** (null handling, deduplication, type coercion), **SQL Query** (schema conventions, JOIN patterns), **Experiment Analysis** (hypothesis testing, effect sizes), and **Structured Reporting** (JSON schema generation). Each Skill contains four sections:

- *Routing* (~50 tokens): activation triggers that tell the agent when to apply the skill (e.g., "activate when the user asks to clean a CSV dataset")
- *Core Procedure* (~190 tokens): step-by-step workflow instructions (e.g., "1. Load data, 2. Identify nulls, 3. Apply imputation, 4. Validate output")
- *Worked Examples* (~225 tokens): concrete input→output demonstrations showing the expected behavior
- *Reference Notes* (~225 tokens): supplementary details such as syntax rules, edge-case handling, and domain conventions (e.g., "for numeric columns with <30% missing, use forward-fill")

These four sections correspond to the content types in Anthropic's documentation [1], but differ in delivery format. Anthropic's architecture stores reference materials in *separate files* loaded only on demand, so they enter the context window only when relevant. Our experiment deliberately uses single flat files instead: by ensuring that all sections, including Reference Notes, are always present in the context window, we can cleanly measure the effect of each section without confounding from selective loading. This format also reflects how most LLM-generated skills work in practice, since the LLM produces one continuous markdown document and cannot create multi-file directory structures without explicit tooling. We discuss implications for progressive-disclosure architectures in §6.

Sections are self-contained with no cross-references, enabling mechanical deletion for ablation. The distinction between Core Procedure and Reference Notes is important for interpreting our results: Core Procedure provides task-aligned guidance, while Reference Notes supply generic heuristics that may or may not match a given task's requirements.

3.2 Ablation Conditions

Our ablation treats Worked Examples and Reference Notes as two independent binary factors (present or absent), with Routing and Core Procedure always included as a shared base. Routing and Core Procedure are held constant rather than ablated for two reasons: first, without activation triggers and procedural steps, the remaining sections (examples, references) lack the context needed to function as a coherent skill; second, Routing triggers are redundant in our setup because each skill is only paired with tasks from its matching family, so the triggers would always fire. This 2×2 design yields four skill conditions, plus a No-Skill baseline:

1. **No-Skill**: task prompt only, no skill content
2. **Core-Only**: Routing + Core Procedure (base only)
3. **Core+Examples**: base + Worked Examples
4. **Core+Refs**: base + Reference Notes
5. **Full**: all four sections (base + Examples + Refs)

Skill content is injected unconditionally as a user-message prefix using a fixed template (`[SKILL START] ... [SKILL END]`, followed by the task prompt) across all conditions and providers. This direct injection is deliberate: our goal is to measure the causal effect of skill *content*, not the accuracy of a routing system that decides whether to apply a skill. Routing accuracy is a separate research question; by injecting unconditionally, we ensure that any observed differences between conditions are attributable to the skill sections themselves. The template does not include an explicit instruction-priority directive (e.g., "task instructions override skill content"), meaning the model must resolve any conflict between skill heuristics and task-specific instructions on its own. This mirrors the default behavior of current skill-injection systems but means that any observed degradation from Reference Notes may be partly mitigable via prompt engineering. We return to this point in §4.2.

3.3 Task Construction and External Validation

We evaluate 56 tasks (14 per skill family), split evenly to mitigate self-authorship bias:

- **Self-authored** (28 tasks, 7 per family): task prompts and gold-standard outputs were created with the assistance of an LLM coding agent (distinct from the nine models under evaluation), spanning a range of difficulty within each family. Gold outputs were validated using automated scripts that recompute expected results from source data.
- **Externally sourced** (28 tasks, 7 per family): task prompts and gold outputs were written against public datasets and APIs,[1] also with LLM coding assistance.

All task prompts and gold outputs were finalized before generating the skills, preventing any post-hoc alignment between tasks and skill content.

3.4 Verification

All outcomes are binary pass/fail, evaluated by fully automated, deterministic verifiers with no human judgment involved. Each task family uses a verification strategy matched to its output type:

- **Data Cleaning**: the model's output CSV is parsed into a DataFrame and compared against the gold CSV. Exact-mode tasks check row-by-row value equality (case-sensitive, column-order-invariant). Structural-mode tasks check for expected columns, row counts, absence of nulls, and date-format compliance.
- **SQL Query**: the model's SQL is executed against an in-memory SQLite database seeded with task-specific test data. The result set is compared to the gold query output with order-insensitive row matching.
- **Experiment Analysis**: the model's numeric outputs (test statistics, p-values, effect sizes) are extracted and compared against gold values with a tolerance of ϵ=0.01. Both absolute and relative tolerance are applied to accommodate rounding differences.
- **Structured Reporting**: the model's JSON output is validated against a task-specific schema (required keys, value types, nested structure). Field values are checked against gold entries where specified.

[1]SQL tasks use schemas from Spider [11] and Chinook/Northwind-style databases; Data Cleaning tasks use the Melbourne Housing, Titanic, UCI Wine Quality, Auto MPG, and student performance datasets; Experiment Analysis tasks use Fisher Iris [4], mtcars, PlantGrowth, ToothGrowth, and the sleep dataset; Structured Reporting tasks use the GitHub Events, OpenWeatherMap, and REST Countries APIs.

Exact-match verifiers risk conflating genuine reasoning failures with surface-level formatting differences. We programmatically identified all condition-flip cases (where a task passes under one condition but fails under another) and reviewed the associated error messages. In each case, the failures reflect substantive errors (e.g., wrong imputation method or wrong case convention due to competing instructions) rather than trivial formatting mismatches.

3.5 Models

We select nine model configurations spanning three major providers (Table 1), chosen to cover three axes of variation. First, *model capability*: we include compact models (GPT-4o-mini, Gemini Flash, Claude Haiku) and frontier models (GPT-4o, Gemini Pro, Claude Sonnet) to test whether stronger models utilize skills differently. Second, *provider diversity*: sampling from OpenAI, Google, and Anthropic, with each provider contributing at least one compact and one frontier model, guards against provider-specific artifacts. Third, *extended thinking*: both Gemini 2.5 Flash and Claude Sonnet 4 are tested with extended thinking disabled and enabled, and o3-mini provides an always-on thinking baseline. For the tier-stratified analysis (§4.3), we classify the three extended-thinking configurations (o3-mini, Gemini Flash$^+$, Claude Sonnet$^+$) as a separate *reasoning* tier, yielding a balanced 3×3 design (three models per tier).

Model	Provider	Thinking	Capability
GPT-4o-mini	OpenAI	No	Compact
GPT-4o	OpenAI	No	Frontier
o3-mini	OpenAI	Yes	Reasoning
Gemini 2.5 Flash	Google	Disabled	Compact
Gemini 2.5 Flash$^+$	Google	Enabled	Reasoning
Gemini 2.5 Pro	Google	Yes	Frontier
Claude Haiku 4.5	Anthropic	No	Compact
Claude Sonnet 4	Anthropic	Disabled	Frontier
Claude Sonnet 4$^+$	Anthropic	Enabled	Reasoning

Table 1. Model configurations. Thinking: Yes = always-on reasoning; Disabled/Enabled = toggleable extended thinking ($^+$). Temperature is 0 for non-reasoning models and provider default otherwise.[2]

Each of the $56 \times 5 \times 9 = 2{,}520$ cells is run three times, yielding 7,560 total runs. We use three repetitions because, even at temperature 0 (greedy decoding, the most deterministic setting available), API providers exhibit minor non-determinism across calls due to infrastructure-level factors such as batching and quantization. Of the 2,520 cells, 93.4% are unanimous (all three repetitions produce the same outcome, whether all-pass or all-fail) and only 6.6% show a split verdict (2-of-3 or 1-of-3 pass), confirming that cross-repetition variance is low. We use majority-vote (2-of-3 pass) as the cell-level outcome.

[2]Exact API model identifiers: `gpt-4o-mini`, `gpt-4o-2024-11-20`, `o3-mini`, `gemini-2.5-flash`, `gemini-2.5-pro`, `claude-haiku-4-5-20251001`, `claude-sonnet-4-20250514`. The $^+$ variants use the same identifiers with extended thinking enabled.

3.6 Analysis

The 2×2 factorial design (§3.2) supports two types of inference. *Additive*: what is the marginal value of adding a section to the base (e.g., Core-Only → Core+Examples)? *Subtractive*: what happens when a section is removed from the Full Skill (e.g., Full → Core+Examples removes Reference Notes)?

We define *task difficulty* as the No-Skill pass rate averaged across all nine models: a task is "easy" if most models pass it without any skill, and "hard" if most fail. We define *skill utility* as the difference in pass rate between a given skill condition and the No-Skill baseline. Because every task is evaluated under all five conditions, each task serves as its own control.

We analyze the results in four steps:

1. **Aggregate mixed-effects model.** We fit a linear mixed-effects model with condition as a fixed effect and crossed random intercepts for task and model to the 2,520 majority-vote outcomes (Table 3). The crossed random intercepts allow each task and each model to have its own baseline pass rate, so the condition effects are estimated after accounting for the fact that some tasks are inherently harder and some models are inherently stronger.[3] The model predicts pass rate directly (rather than log-odds), which makes the coefficients interpretable as percentage-point differences.
2. **Model-capability interaction model.** To test whether skill effects differ by model capability, we fit a second model adding a condition × model-capability interaction term (compact vs. frontier vs. reasoning), retaining crossed random effects.
3. **Bootstrap confidence intervals.** Pairwise CIs are computed via bootstrap resampling (10,000 iterations) over the 56 task-level paired differences, using the percentile method.
4. **McNemar paired tests.** For each skill condition, we count the number of (model, task) pairs where the skill flips the outcome from fail to pass versus pass to fail, and test whether these counts differ using McNemar's test.

	No-Skill	Full	Core-Only	Core+Ex	Core+Refs
GPT-4o-mini	55.4%	55.4%	55.4%	50.0%	**57.1%**
GPT-4o	**62.5%**	**62.5%**	57.1%	58.9%	58.9%
o3-mini	78.6%	78.6%	76.8%	**80.4%**	**80.4%**
Gemini 2.5 Flash	53.6%	57.1%	57.1%	**60.7%**	55.4%
Gemini 2.5 Flash$^+$	57.1%	55.4%	**58.9%**	57.1%	55.4%
Gemini 2.5 Pro	67.9%	66.1%	69.6%	**75.0%**	67.9%
Claude Haiku 4.5	**66.1%**	64.3%	64.3%	64.3%	64.3%
Claude Sonnet 4	**82.1%**	78.6%	80.4%	80.4%	80.4%
Claude Sonnet 4$^+$	**75.0%**	73.2%	71.4%	73.2%	69.6%
Average	***66.5%***	*65.7%*	*65.7%*	***66.7%***	*65.5%*

Table 2. Pass rates by model and condition (56 tasks × 3 reps each, majority-vote). Bold indicates best condition per model; underline indicates worst. Per-model standard deviations across tasks range from 0.41 to 0.50, reflecting the binary nature of the outcome.

4 Results

4.1 Aggregate: No Condition Dominates

Table 2 presents pass rates by model and condition. The aggregate spread across all five conditions is just 1.2 pp (65.5% to 66.7%). The mixed-effects model (Table 3) confirms that no condition reaches statistical significance at α=0.05; all four skill conditions fall well short (all $p \geq 0.587$). Paired McNemar tests provide independent confirmation: for each skill condition, the number of (model, task) pairs where the skill flips the outcome from fail to pass is approximately equal to the number flipped from pass to fail (e.g., Core+Examples: 20 helped vs. 19 hurt, p=1.000). Meanwhile, skill content substantially increases prompt length: Full skills consume ~4.7× the input tokens of No-Skill (~1,150 vs. ~240 tokens), providing no measurable benefit.

Condition	$\hat{\beta}$	p
Full	−0.008	0.664
Core-Only	−0.008	0.664
Core+Ex	+0.002	0.914
Core+Refs	−0.010	0.587

Table 3. Linear mixed-model fixed effects (n=2,520 majority-vote outcomes, reference: No-Skill). $\hat{\beta}$: effect in pass-rate points. No condition reaches significance (α=0.05).[4]

[3]We use majority-vote outcomes to avoid pseudo-replication from treating three repetitions as independent observations; 93.4% of cells show unanimity (3/3 agree), so this discards very little information. We use random intercepts rather than random slopes because the number of variance-covariance parameters for condition slopes would exceed what 56 task groups and 9 model groups can stably estimate; a random-slopes variant fails to converge.
[4]A logistic GEE with exchangeable correlation and a non-parametric permutation test (10,000 iterations) yield substantively identical conclusions: all p>0.45 across all conditions. We report the linear model for interpretability, as its coefficients directly express percentage-point differences.

To understand why the aggregate is so flat, we examine how task difficulty (§3.6) distributes across families. Table 4 shows pass rates grouped by task family; because each family targets a different data-analysis workflow, families also differ substantially in baseline difficulty.

Family	No-Skill	Full	C-Only	C+Ex	C+Refs	Range (SD)
SQL Query	88.9	91.3	89.7	90.5	89.7	0–100 (26.1)
Struct. Report	90.5	92.1	92.1	91.3	92.1	0–100 (26.8)
Data Cleaning	57.1	50.8	52.4	54.8	52.4	0–100 (32.6)
Exp. Analysis	29.4	28.6	28.6	30.2	27.8	0–67 (28.4)

Table 4. Pass rates (%) by task family, averaged across all models. Range and standard deviation (SD) report within-family task difficulty variation (per-task No-Skill pass rates across all models).

Baseline difficulty varies dramatically across families, compressing the range where skills could matter. SQL tasks and Structured Reporting sit near ceiling (88.9% and 90.5% No-Skill baseline), leaving little room for improvement. This likely reflects both the relative simplicity of many tasks in these families and the extensive representation of SQL and JSON generation in LLM training corpora. Experiment Analysis tasks sit near floor (29.4%), where even the best condition adds only +0.8 pp. These tasks require multi-step statistical reasoning (hypothesis tests, effect-size computations), and the low baseline suggests that models struggle with them regardless of skill condition. Data Cleaning (57.1%) is the most informative family, sitting in the moderate-difficulty range, yet even here skills show a slight negative direction (−2.3 to −6.3 pp).

Difficulty also varies substantially *within* each family. Of the 56 tasks, 17 fall in the informative 30 to 80% baseline range (9 Data Cleaning and 7 Experiment Analysis tasks, plus 1 Structured Reporting task). The remaining 39 tasks are at ceiling (27 tasks above 80%) or floor (12 tasks below

30%). On the 17 informative tasks, all four skill conditions show a slight negative direction (−2.0 to −5.2 pp), though none reaches significance given the sample size. This pattern suggests that skills are, if anything, marginally harmful on tasks where they could plausibly make a difference.

4.2 Component-Level Patterns

Although no component-level comparison reaches significance, the partial-skill conditions are remarkably close to each other and to the No-Skill baseline:

- **Reference Notes show a slight negative direction**: Core+Refs (65.5%) falls below No-Skill (66.5%) by −1.0 pp, and below Core-Only (65.7%) by −0.2 pp.
- **Worked Examples are approximately neutral**: Core+Examples (66.7%) is the closest condition to No-Skill (66.5%), differing by just +0.2 pp.
- **The Full Skill bundles both**: Full (65.7%) matches Core-Only, suggesting that adding both Examples and Reference Notes to the base procedure produces no net change.

The near-zero aggregate differences are not an artifact of cancellation across models: Table 2 shows that 4 of 9 models achieve their best pass rate under No-Skill, and no single skill condition is best for more than 3 models. The pattern is consistent with skills providing no systematic benefit in this domain.

Although skills do not help on average, they are not entirely without effect: among the 14 task×model pairs where Core-Only passes but Core+Refs fails (the largest condition-flip category), Data Cleaning tasks account for 8 of 14 flips, and Google-provider models (Gemini Flash, Flash$^+$, and Pro) account for 10 of 14. One representative case (a Data Cleaning task on Gemini 2.5 Pro) reveals the mechanism:

> **The task** instructs: "fill missing ages with *median*, missing ports with 'Unknown'." **The Reference Notes** instead prescribe: "For numeric columns with <30% missing, apply *forward-fill*. For string columns, fill with 'UNKNOWN'." Under Core-Only, the model follows the task instructions and passes. Under Core+Refs, the model adopts the skill's generic heuristics (forward-fill instead of median, uppercase "UNKNOWN" instead of title-case "Unknown"), producing an output that fails verification.

This failure mode is concrete and actionable: when skills contain generic heuristics that conflict with task-specific instructions, the model tends to follow the skill content. We note that this may partly reflect the quality of our auto-generated Reference Notes (which are over-prescriptive by design) rather than an inherent problem with reference material as a section type; expert-curated or task-specific reference notes might not exhibit the same pattern. Adding an explicit instruction-priority directive (e.g., "task instructions override skill content") to the prompt template could also mitigate this, but we deliberately omitted it to test the default behavior of current skill-injection systems (§3.2).

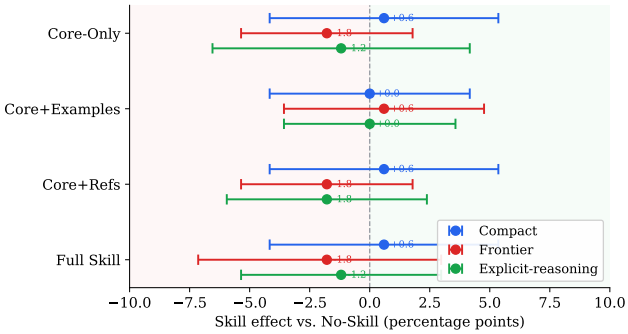

Figure 1. Skill effects (vs. No-Skill) with 95% bootstrap CIs, stratified by model capability. All effects cluster near zero; no condition produces a significant benefit or harm for any capability level.

4.3 Skill Effects Do Not Vary by Model Capability

One natural hypothesis is that the aggregate null conceals opposing effects at different capability levels: perhaps compact models benefit from structured guidance while frontier models are harmed. To test this, we classify models into three tiers based on capability and inference mode: compact (GPT-4o-mini, Gemini Flash, Claude Haiku), frontier (GPT-4o, Gemini Pro, Claude Sonnet), and reasoning (o3-mini, Gemini Flash$^+$, Claude Sonnet$^+$), with three models per tier. We then stratify the bootstrap analysis by tier.

Figure 1 shows no strong evidence for this hypothesis. Compact models show effects centered near zero for all conditions (largest: Core-Only +0.6 pp, 95% CI [−4.2, +5.4]). Frontier models show a slight negative direction (largest: Core-Only and Core+Refs −1.8 pp, CI [−5.4, +1.8]), and reasoning models show a similar pattern (largest: Core+Refs −1.8 pp, CI [−6.0, +2.4]). All CIs include zero across all three tiers. The slight positive direction for compact models and slight negative direction for frontier and reasoning models average to the near-zero aggregate, with no tier reaching significance.

Notably, individual models within each tier show considerable heterogeneity. Among compact models, Gemini 2.5 Flash gains +7.1 pp from Core+Examples (53.6% → 60.7%), but GPT-4o-mini loses −5.4 pp from the same condition (55.4% → 50.0%). This within-tier variation, rather than a systematic tier-level pattern, explains why no tier shows a significant effect. The null is not an artifact of averaging over opposing tier-level trends; it is consistent with broad indifference to skill content across the capability spectrum.

5 Discussion

Our results converge on a single finding: in the data-analysis domain, single-shot LLM-authored skills provide no measurable benefit. This null holds across all five conditions, all nine models, all four task families, and all three model-capability tiers. We consider three explanations for why skills fail to help.

First, for the two highest-performing families, models are already proficient. SQL Query (88.9% No-Skill baseline) and Structured Reporting (90.5%) leave little room for improvement, consistent with the heavy representation of SQL and JSON generation in LLM training corpora. For the two harder families, different mechanisms appear to be at work. Data Cleaning (57.1%) sits at moderate difficulty, where skills actively introduce conflicting heuristics: as shown in §4.2, generic rules like "use forward-fill for columns with less than 30% missing" override task-specific imputation requirements, producing errors that would not occur without the skill. Experiment Analysis (29.4%) sits near floor, with the best skill condition adding only +0.8 pp. At such low baselines, we cannot distinguish whether skills fail because their content is mismatched to these tasks or because the underlying capability gap is too large for any prompt-level intervention to bridge. In short, skills are redundant where models are strong, actively harmful where they conflict with task instructions, and insufficient to bridge capability gaps at low baselines.

Second, our skills target entire task families rather than individual tasks. SkillsBench found that focused skills with two to three modules outperform comprehensive documentation [7]; our family-level skills are closer to the comprehensive end of this spectrum, covering all data-cleaning or all experiment-analysis tasks with a single set of instructions. This breadth necessarily sacrifices specificity: a family-level skill cannot anticipate the particular imputation method or statistical test that each task requires. Task-specific skills that address the exact requirements of each task might show larger effects.

Third, the context overhead of skill content may offset any benefit. Full skills consume ∼4.7× the input tokens of No-Skill, and the additional content can introduce conflicting heuristics (§4.2). The observation that 4 of 9 models achieve their best pass rate under No-Skill suggests that the cost of processing skill content can exceed its value. In domains where baseline performance is lower, such as healthcare (+51.9 pp skill effect [7]) or manufacturing (+41.9 pp), the informational value of skill content may outweigh this overhead.

Two additional observations warrant brief mention. Enabling extended thinking raises the baseline for Gemini Flash (+3.6 pp) but lowers it for Claude Sonnet (−7.1 pp), though with only two data points this is suggestive. When split by task source, self-authored tasks (baseline 66.7%) and externally sourced tasks (baseline 66.3%) show nearly identical skill effects (−0.7 pp and −0.5 pp respectively), confirming that the null is not an artifact of task provenance.

6 Limitations

Our study has three main categories of limitations. The first concerns *scope*. Although 17 of 56 tasks fall in the informative 30 to 80% baseline range, providing reasonable coverage, the remaining 39 tasks are at ceiling or floor, limiting detection power on those tasks. Our 95% bootstrap CIs span ±4 to 6 pp, meaning we can rule out moderate-sized effects but not small ones (1 to 3 pp). Scaling to hundreds of tasks with more informative baselines would further tighten these bounds.

The second concerns *generalizability*. Our setup evaluates single-turn prompt-response pairs in one domain (data analysis), where two of the four task families exhibit high No-Skill baselines (SQL 88.9%, Structured Reporting 90.5%). Agentic workflows with multi-turn tool use, and domains with uniformly lower baselines (healthcare, cybersecurity), may show larger skill effects. Our findings apply only to LLM-authored, family-level skills and should be validated with expert-curated, task-specific skills.

The third concerns *potential confounds*. Skill conditions increase prompt length (from ∼240 tokens for No-Skill to ∼1,150 for Full), so any degradation may partly reflect longer prompts rather than content harm. Because we use a single generated skill per family, section effects are confounded with the idiosyncrasies of those four specific skills; a different generation run might produce less over-prescriptive reference notes. Our always-on flat-file injection differs from Anthropic's recommended progressive-disclosure architecture [1], so the null may not generalize to selective-loading systems that inject content only when relevant.

7 Conclusion

We presented the first component-level ablation of LLM-authored Agent Skills in data analysis. Across 56 tasks, nine models, and three providers (2,520 majority-vote outcomes), no skill condition significantly outperforms the No-Skill baseline. The null is robust: it holds across all four skill components, all three model-capability tiers, and all four task families. Paired McNemar tests confirm that skills help and hurt on roughly equal numbers of task×model pairs, and bootstrap CIs bound the maximum possible benefit to less than ±4 pp at the aggregate level (±6 pp within individual tiers).

These results suggest that single-shot LLM-authored skills fail through different mechanisms at different baseline levels: they are redundant where models already perform well (SQL, Structured Reporting), introduce conflicting heuristics at moderate baselines (Data Cleaning), and cannot compensate for capability gaps at low baselines (Experiment Analysis).

This finding does not contradict evidence that expert-curated skills help substantially in specialized domains [7]; rather, it suggests that baseline difficulty alone does not determine skill utility. Natural extensions include: (1) evaluating domains with lower baseline performance where skill content may add genuine value, (2) comparing family-level skills against task-specific skills to test whether granularity matters, (3) testing progressive-disclosure delivery to determine whether selective loading improves outcomes, and (4) comparing alternative skill-generation strategies (iterative refinement, multi-candidate selection) against our single-shot baseline.

References

[1] Anthropic. 2025. Agent Skills: Overview. https://platform.claude.com/docs/en/agents-and-tools/agent-skills/overview Accessed: 2026-04-28.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, Vol. 33. Curran Associates, Inc., Red Hook, NY, USA, 1877–1901.

[3] Le Chen, Erhu Feng, Yubin Xia, and Haibo Chen. 2026. SkVM: Revisiting Language VM for Skills across Heterogenous LLMs and Harnesses. arXiv:2604.03088

[4] Ronald A. Fisher. 1936. The use of multiple measurements in taxonomic problems. *Annals of Eugenics* 7, 2 (1936), 179–188.

[5] Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Scott Wen-tau Yih, Daniel Fried, Sida Wang, and Tao Yu. 2022. DS-1000: A Natural and Reliable Benchmark for Data Science Code Generation. arXiv:2211.11501

[6] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.

[7] Xiangyi Li et al. 2026. SkillsBench: Benchmarking How Well Agent Skills Work Across Diverse Tasks. arXiv:2602.12670 https://arxiv.org/abs/2602.12670

[8] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity. arXiv:2104.08786

[9] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Arber, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the Role of Demonstrations: What Makes In-Context Learning Work? arXiv:2202.12837

[10] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774

[11] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 3911–3921.

[12] Dan Zhang, Sining Zhoubian, Min Cai, et al. 2025. DataSciBench: An LLM Agent Benchmark for Data Science. arXiv:2502.13897

[13] Shanshan Zhong, Yi Lu, Jingjie Ning, Yibing Wan, Lihan Feng, Yuyi Ao, Leonardo F. R. Ribeiro, Markus Dreyer, Sean Ammirati, and Chenyan Xiong. 2026. SkillLearnBench: Benchmarking Continual Learning Methods for Agent Skill Generation on Real-World Tasks. arXiv:2604.20087