Do LLM-Generated Skills Help? A Multi-Provider Ablation for Data-Science Agents

Anonymous ACL submission

Abstract

Agent platforms increasingly let users attach reusable `SKILL.md` files to future tasks. The idea is practical: write procedures, examples, and reference notes once, then reuse them across related tasks. Human-written skills can provide careful guidance, but writing them for many task families is costly. We ask whether LLM-generated skills offer a useful low-curation alternative for agents working on data-science tasks.

We evaluate LLM-generated skills on 56 deterministically verifiable tasks across data preparation, data extraction, statistical analysis, and reporting. We conduct an ablation study spanning 15 LLM-generated skill sets and nine agent model configurations from OpenAI, Anthropic, and Google, for 37,800 runs.

We find that LLM-generated skills do not reliably improve task success in this setting. Full skills and all ablated variants have confidence intervals that include zero, while Full skills increase input-token use by 7.54×. Supplemental audits do not point to No-Skill instability or task-irrelevant skill-formatted context as explanations for the flat result. Low-curation flat-file skills therefore need validation before they can be treated as performance-improving context.

1 Introduction

Product data scientists often ask agents based on large language models (LLMs) to help with recurring execution tasks: cleaning data, writing SQL, choosing statistical tests, computing effect sizes, and formatting reports. We refer to these systems as *data-science agents*, following recent benchmarks for data-science automation (Zhang et al., 2025; Nascimento et al., 2024).

In this paper, the term denotes LLM-based systems that execute data-science tasks. It does not require multi-turn planning, persistent memory, or tool use.

For such systems, model choice is only part of the design problem. The other question is what guidance, if any, should be placed in front of the model.

Recent agent platforms and benchmarks use reusable skill files, such as `SKILL.md`, to package task instructions, examples, and reference notes (Anthropic, 2025; Li et al., 2026). For data-science agents, the reuse pattern is tempting: write one skill for recurring task families such as data preparation, SQL generation, statistical analysis, and reporting, then prepend it to future tasks from the same family.

Skills can be authored by human domain experts or generated by LLMs. We focus on LLM-authored skills because they can be produced from high-level task-family specifications or public documentation without human expert authoring for each task family.

This low-curation path matters because skill files are meant to be reused. If they have to be hand-tuned for every task family, much of their practical appeal disappears. The central question is whether the cheaper version actually improves task execution.

Prior evidence gives reason for caution. SkillsBench (Li et al., 2026) (86 tasks, 7,308 trajectories) found that human-curated skills improve pass rates by 16.2 percentage points, while LLM-generated skills provide no aggregate benefit.

This aggregate result raises a component-level question: are all skill components unhelpful, or do useful and harmful components offset one another? If the latter, pruning could improve LLM-authored skills while preserving much of their low-curation advantage.

We study these questions across four execution-facing data-science task families: **data preparation** (null handling, deduplication, type coercion), **data extraction** (SQL query formulation), **statistical analysis** (hypothesis testing and effect size

computation), and **reporting** (structured JSON reporting). These task families represent common execution tasks for data-science agents, and their outputs can be checked by deterministic verifiers.

Our evaluation varies both sides of the skill pipeline. We use *generator* to mean an LLM configuration that writes skill files, and *agent model* to mean the LLM configuration that receives a task prompt and produces the scored output.

We generate 15 skill sets, five from each of three authoring providers, so that conclusions are not tied to a single generated artifact. We then evaluate each skill set across nine agent model configurations from three providers. For each skill set, we compare the No-Skill baseline against a full generated skill and three ablation variants.

We make three primary contributions:

- We evaluate low-curation LLM-generated skills as reusable artifacts across 15 regenerated skill sets, four data-science task families, and nine agent model configurations.
- We introduce a multi-provider ablation design that separates full-skill effects from the effects of the procedural core, worked examples, and reference notes.
- We observe no reliable improvement from either Full skills or ablated variants.

2 Related Work

A common way to adapt LLM behavior is to add external guidance at inference time, such as retrieved documents, examples, or task-specific instructions (Lewis et al., 2020; Brown et al., 2020). Prior work on prompting shows that these inputs can matter in subtle ways.

For example, label correctness in demonstrations can matter less than format and input distribution (Min et al., 2022), and example ordering can affect performance (Lu et al., 2022).

This work primarily examines localized prompt-level interventions. Our setting moves that question to a larger unit of intervention: a structured file that packages routing guidance, procedures, examples, and reference notes for repeated use.

Reusable skills have also been evaluated directly. SkillsBench (Li et al., 2026) found that focused human-curated skills improve performance, while LLM-generated skills provide no aggregate benefit. SkillLearnBench (Zhong et al., 2026) studies continual skill-learning methods and finds gains over no-skill baselines, but no method dominates across tasks and models. SkVM (Chen et al., 2026) treats skills as compiled capabilities across heterogeneous LLM backends.

These studies motivate testing reusable skills, but they do not identify which parts of generated skills help or hurt in data-science task families.

Our study focuses on execution-facing data-science tasks and asks whether a null aggregate result for generated skills hides component-level effects, using regenerated skills from multiple authoring providers and mechanical component ablations.

Two recent anonymous papers by overlapping authors, submitted to non-archival workshops, study nearby versions of this question. Anonymous (2026a) reports a component-level ablation of LLM-authored skills for data-analysis tasks in a smaller single-generation setting, focusing on whether procedures, examples, or reference notes explain the aggregate null result. Anonymous (2026b) studies LLM-generated skills for AI data-scientist workflows and includes a token-matched control for prompt-length effects. The present paper differs by regenerating 15 skill sets with LLMs from three providers, OpenAI, Google, and Anthropic, expanding the controlled evaluation to 37,800 model runs across nine agent model configurations, and treating provider variation and sensitivity checks as part of the primary analysis.

DS-1000 (Lai et al., 2023), DataSciBench (Zhang et al., 2025), and LLM4DS (Nascimento et al., 2024) evaluate LLM code generation for data-analysis tasks and show that strong models can solve many tasks from the prompt alone. This creates a demanding test for reusable skills: if the No-Skill baseline already covers many common procedures, skills have less room to help and may still add cost or conflict with task-specific instructions.

Recent data-preparation and table-reasoning systems point to a different form of support than static front-loaded instructions. PrepBench emphasizes ambiguous intents, imperfect data, interactive disambiguation, code generation, and process translation (Xu et al., 2026). AutoDCWorkflow generates data-cleaning procedures from a raw table and analysis purpose (Li et al., 2025).

Chain-of-Query uses multi-agent, clause-by-clause SQL generation for table understanding (Sui et al., 2025). These systems do not evaluate reusable skill files directly, but they suggest one limitation of flat prompt injection: data-science

agents may need task-specific orchestration, validation, and feedback.

3 Experimental Design

The experiment compares the No-Skill baseline with a controlled family of LLM-generated skill prompts. We first generate reusable skill sets without exposing the generator to task prompts, reference outputs, or verifier code. We then derive ablations from those same files, evaluate all conditions on the same data-science tasks and agent model configurations, score outputs with deterministic verifiers, and estimate paired pass-rate changes.

This design keeps the source skill fixed while varying which parts of the skill the agent model can see.

3.1 Generated Skill Sets

We generate skills using one high-capability reasoning configuration from each of OpenAI, Google, and Anthropic: GPT-5.5-ExtraHigh, Gemini-3.1-Pro-High, and Claude-Opus-4.6-Thinking. This gives the generated-skill condition a capable authoring setting, so evaluation results are not primarily a test of whether compact or less capable models can write instructions.

Each generation run produces one *skill set*: a matched bundle of four task-family skills, one for each task family. Each provider produces five skill sets, yielding 15 generated skill sets and 60 generated skill files.

Skill generation is conducted in isolated web-enabled runs. Each generator receives the same prompt, can use public web search, and writes a plan and session log before producing the four final skill files. These intermediate files document the run and support protocol checks; they are not used during evaluation.

The generator is not provided with previously generated skill sets, task prompts, reference outputs, or verifier implementations. This reduces direct cross-run contamination while allowing access to publicly available information.

The prompt specifies the target task family, the `SKILL.md` schema, and the required section structure. Generated skills are not manually edited, ranked, or selected after creation. This preserves the low-curation setting under study: a practitioner asks an LLM to write reusable task-family guidance, then uses the generated file as-is.

3.2 Skill Format and Ablation Conditions

Each generated skill contains four components, implemented as sections in the skill file:

- *Routing*: activation triggers specifying when to apply the skill.
- *Core Procedure*: step-by-step task-family instructions.
- *Worked Examples*: concrete input→output demonstrations.
- *Reference Notes*: supplementary heuristics and conventions.

These components correspond to the content types in the Agent Skills documentation (Anthropic, 2025), but our delivery format is different. Anthropic's architecture stores reference materials in separate files loaded only on demand. Our experiment instead uses single flat files, so every skill component is visible to the agent model unless it is removed by ablation.

This choice makes the component comparison direct: estimated effects reflect visible component content rather than selective loading. The format also matches a common low-curation workflow, where the LLM produces one continuous markdown document rather than a multi-file skill package.

We construct ablations by mechanically deleting components from the full file, not by running separate generation procedures. The design asks whether the procedural core changes performance relative to the No-Skill baseline, and whether worked examples or reference notes add measurable value once the core task-family guidance is present.

Components are written to be self-contained, so deletion does not leave dangling instructions. Routing and Core Procedure are held fixed because examples and reference notes are difficult to interpret without the base procedure.

Worked Examples and Reference Notes are varied as two binary factors, producing four variants:

1. **Core-Only**: Routing + Core Procedure.
2. **Core+Examples**: Routing + Core Procedure + Worked Examples.
3. **Core+Refs**: Routing + Core Procedure + Reference Notes.
4. **Full**: all four components.

We compare all skill conditions against **No-Skill**, which contains only the task prompt. Skill content

is injected as a user-message prefix using the same wrapper across models and providers.

To test whether results can be explained by extra skill-formatted context rather than data-science content, we add a supplemental **Length-Control** condition. For each Full skill, the control uses the same injection wrapper and four-section structure but replaces the data-science content with coherent, task-irrelevant guidance for arranging a tabletop sketching kit. The control is not intended to help data preparation, data extraction, statistical analysis, or reporting.

The Length-Control prompts are closely matched to Full prompts in provider-reported input tokens: they are 17.6 input tokens shorter on average, a -0.93% relative difference. We therefore use the condition as a targeted audit for an extra-context explanation, not as a claim that prompt length can never matter.

To limit cost, the token-matched Length-Control condition covers all 56 tasks and all 15 skill sets, but uses three selected agent model configurations spanning the three providers and model types: GPT-5.4 Mini (compact), Gemini 3.1 Pro (low reasoning), and Claude Opus 4.7 (high reasoning). This adds 2,520 model outputs and preserves variation across tasks and generated skill artifacts while reducing model coverage.

3.3 Task Construction and External Validation

The controlled evaluation contains 56 tasks, with 14 tasks per task family:

- **Data Preparation**: CSV cleaning, including null imputation, deduplication, type coercion, and date standardization.
- **Data Extraction**: SQL query formulation against relational databases.
- **Statistical Analysis**: hypothesis tests, effect sizes, confidence intervals, and multiple-comparison adjustments.
- **Reporting**: structured JSON and tabular reports from semi-structured inputs.

The tasks target the verifiable execution layer of data-science work rather than the full product decision-making lifecycle. They therefore exclude upstream problem framing, metric selection, and stakeholder tradeoffs, which are important but harder to score with deterministic verifiers.

Tasks are split evenly between self-authored and externally grounded examples. The self-authored subset contains 28 tasks drafted with LLM assistance and then validated by scripts. The externally grounded subset contains 28 tasks based on public datasets, public schemas from prior benchmarks, or public API formats.

These tasks use compact synthetic fixtures or prompt-level adaptations rather than redistributed full datasets; Appendix B lists the public datasets and schemas that inspired them. All prompts, fixtures, and reference outputs are stored with the task set, so scoring does not depend on network access.

Task prompts and reference outputs were finalized before skill generation to avoid post-hoc alignment between task requirements 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 verifier matched to its output type:

- **Data Preparation**: the model's output CSV is parsed and compared against the reference table. Exact-mode tasks check row-level values with task-specific row-order and numeric-tolerance policies. Structural-mode tasks check expected columns, row counts, absence of invalid values, and required format constraints.
- **Data Extraction**: the model's SQL is executed against in-memory SQLite fixtures. The result set is compared to the reference output with task-specific row-order policies.
- **Statistical Analysis**: numeric outputs, such as test statistics, p-values, effect sizes, and confidence intervals, are extracted and compared against reference values using task-specific absolute and relative tolerances, including checks near significance boundaries.
- **Reporting**: JSON or table outputs are validated against task-specific schemas, including required keys or columns, value types, nested structure, and selected field values where specified.

Verifier errors are recorded separately from model failures. This distinction matters because malformed artifacts or environment issues should not be counted as task failures. The task set includes validation scripts that execute SQL references, recompute statistical reference outputs, check structured-reporting schemas, and run negative controls before model outputs are scored.

3.5 Agent Models

We select nine agent model configurations spanning three providers (Table 1), varying along three dimensions. First, *model type*: we include compact configurations and frontier configurations with low or high reasoning budgets. Second, *provider diversity*: sampling from OpenAI, Google, and Anthropic guards against provider-specific artifacts.

Third, *inference-time effort*: for each provider, we include frontier configurations at lower and higher reasoning settings where the provider exposes such controls.

For stratified analysis, we use three pre-specified model types: compact, low reasoning, and high reasoning. These labels normalize provider-specific effort controls into comparison strata. They should not be read as a universal ranking of model quality across providers.

Provider	Model label	Type
OpenAI	GPT-5.4 Mini	Compact
OpenAI	GPT-5.5	Low reasoning
OpenAI	GPT-5.5	High reasoning
Google	Gemini 3.1 Flash Lite	Compact
Google	Gemini 3.1 Pro	Low reasoning
Google	Gemini 3.1 Pro	High reasoning
Anthropic	Claude Haiku 4.5	Compact
Anthropic	Claude Opus 4.7	Low reasoning
Anthropic	Claude Opus 4.7	High reasoning

Table 1: Agent model configurations.[1]

Because proprietary providers can rename, retire, or change hosted models, we treat these entries as evaluation configurations rather than stable public product identifiers. Appendix A reports the configured API identifiers, and the released harness records request parameters and preflight status.

3.6 Analysis Methods

The experiment is paired by design. For each task, agent model configuration, and generated skill set, we observe a No-Skill output and the outputs produced with each skill condition. Each task-model-skill-set cell can therefore serve as its own comparison unit: the analysis asks whether adding skill content changes the pass/fail outcome relative to the matching No-Skill baseline.

We use three quantities throughout the results. *Task difficulty* refers to how often a task is solved without skill content. *Skill effect* refers to the paired pass-rate difference between a skill condition and its reference condition. *Baseline variation* refers to pass/fail variation among repeated No-Skill calls for the same task and agent model configuration.

These quantities separate three issues that raw pass rates would otherwise mix together: some tasks are easier than others, some agent model configurations are stronger than others, and different generated skill sets contain different guidance.

The main comparisons answer three questions:

- Do Full skills change performance relative to the matched No-Skill baseline?
- Which skill components account for any change: the procedural core, worked examples, reference notes, or their combination?
- Do effects differ across task families, model types, or skill-generator providers?

We use paired effects for these questions because the same task, agent model configuration, and generated skill set appear under the compared conditions.

Paired effects. For task t, agent model configuration m, generated skill set s, and condition c, let $Y_{tmsc} \in \{0, 1\}$ denote verifier success. The main paired quantity is

$$\Delta_{tmsc} = Y_{tmsc} - Y_{tms,\text{NoSkill}}.$$

This compares a skill-conditioned output to the No-Skill output submitted in the same skill-set batch. We report the mean of Δ_{tmsc} in percentage points. Full versus No-Skill is the primary comparison. Core-Only versus No-Skill asks whether the procedural core alone changes performance. Full versus Core-Only asks whether the optional components add value beyond that core. The remaining ablation comparisons separate the effects of worked examples and reference notes.

Uncertainty. We report 95% percentile intervals from a stratified clustered bootstrap with 10,000 samples. Each bootstrap sample resamples tasks with replacement within each task family and resamples skill sets with replacement within each skill-generator provider.

The nine agent model configurations are kept fixed because they are planned evaluation targets, not a random sample of possible models. We use the same bootstrap for aggregate effects, task-family slices, model-type slices, and provider slices.

[1] Model labels are readable names for configured API calls. Exact API identifiers, effort settings, and preflight status are recorded in the evaluation harness. Model types are normalized labels for planned comparison strata.

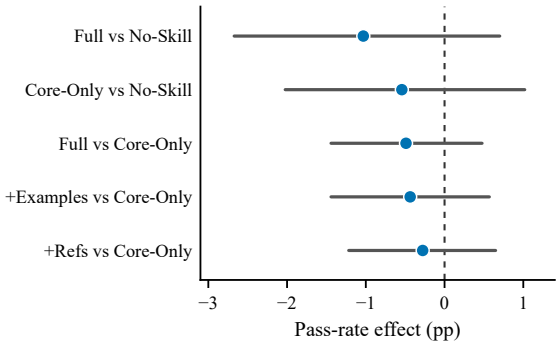

Figure 1: Paired pass-rate effects.

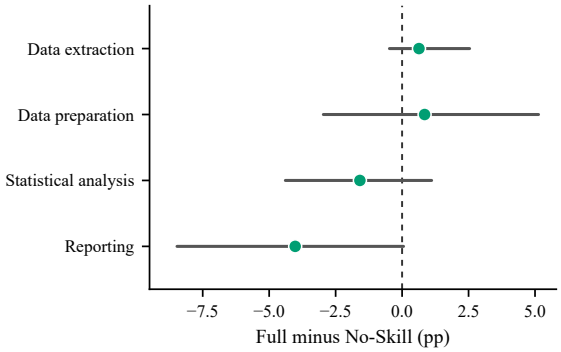

Figure 2: Full-skill effects by task family.

4 Results

We first report the aggregate paired effects, then examine component ablations, task-family variation, and variation by model type and skill-generator provider. We then present two token-related analyses: a token-matched control for context length and a token-overhead summary.

We close with a No-Skill stochasticity check, since it qualifies the one-shot evaluation design but does not change the primary estimand.

4.1 Aggregate Results: No Generated Skill Condition Reliably Improves Performance

Table 2 gives the main paired comparisons. Full skills do not produce a reliable aggregate change over No-Skill: the estimated difference is -1.03 percentage points, with a clustered 95% CI of [-2.67, +0.70]. Collapsing the 15 repeated No-Skill rows to one model-task mean gives the same estimate, -1.03 percentage points, with a similar CI of [-2.59, +0.53]. In other words, the flat result is not an artifact of which repeated No-Skill row is used for pairing.

This estimate averages over 15 regenerated skill sets from three authoring providers, with each skill-conditioned output paired against the No-Skill baseline for the same task, agent model configuration, and skill-set batch. It measures the reliability of low-curation generated skills as reusable artifacts in this controlled evaluation, not the best possible hand-edited, task-specific, or selectively loaded skill.

4.2 Component-Level Analysis

The component-level results do not support a reliable directional claim. Full versus Core-Only, Core+Examples versus Core-Only, Core+Refs versus Core-Only, Full versus Core+Examples, and Full versus Core+Refs all have clustered intervals that include zero (Table 2). These results do not show that worked examples or reference notes add measurable value in the aggregate, but they also do not isolate a clear aggregate harm from either component.

The component analysis narrows the interpretation. The generated skills are not failing because one optional component is clearly harmful across the whole evaluation. The safer reading is more mundane: generated skills and their ablations produce similar single-shot pass rates once task, agent model configuration, and generated skill set are paired.

4.3 Effects Across Task Families

The aggregate result can hide different behavior across task families, so we examine Full versus No-Skill separately for data preparation, data extraction, statistical analysis, and reporting. This analysis is secondary because each family contains only 14 tasks. We use it to interpret the aggregate result rather than to introduce new confirmatory claims.

The pass-rate pattern is not uniform across families. Data extraction is near ceiling, with No-Skill passing 99.10% of paired rows and Full passing 99.74%. Data preparation is also high in this run, with 78.15% for No-Skill and 78.99% for Full. Statistical analysis is lower, with 59.47% for No-Skill and 57.88% for Full.

Reporting has the closest task-family contrast: Full trails No-Skill by -4.02 percentage points, with a clustered 95% CI of [-8.47, +0.05]. Because this interval still includes zero, we do not treat the direction as confirmed.

We also inspected paired cells where No-Skill passed and Full failed. This failure-mode analysis

Comparison	Reference pass	Treatment pass	Effect	95% CI
Full vs No-Skill	74.02%	72.99%	-1.03 pp	[-2.67, +0.70]
Core-Only vs No-Skill	74.02%	73.48%	-0.54 pp	[-2.02, +1.02]
Full vs Core-Only	73.48%	72.99%	-0.49 pp	[-1.44, +0.48]
Core+Examples vs Core-Only	73.48%	73.04%	-0.44 pp	[-1.44, +0.57]
Core+Refs vs Core-Only	73.48%	73.20%	-0.28 pp	[-1.22, +0.65]
Full vs Core+Examples	73.04%	72.99%	-0.05 pp	[-0.93, +0.75]
Full vs Core+Refs	73.20%	72.99%	-0.21 pp	[-1.20, +0.75]

Table 2: Main paired effects.

identifies which errors occur when skill injection changes a passing output into a failing one.

Reporting failures are mostly not invalid JSON. In 134 of 136 cases, the output satisfies the schema but mismatches content fields checked by the verifier. Examples include `duration_days` off by one day, rounded weather averages differing by 0.1, and canonical timeline strings changed inside otherwise valid JSON.

Statistical-analysis failures look different: 102 of 120 cases are numeric statistic mismatches. Data-preparation failures are mainly numeric value mismatches (48 of 88), with smaller counts of column, row, and categorical mismatches. These labels come from verifier outputs rather than human coding, but they suggest that the aggregate result is not driven by one failure category.

4.4 Effects Across Agent Models and Skill-Generator Providers

We do not find reliable aggregate heterogeneity by model type or skill-generator provider for Full versus No-Skill. For model types, the clustered intervals include zero for compact, low-reasoning, and high-reasoning configurations. For skill-generator providers, the intervals include zero for skills generated with OpenAI, Google, and Anthropic configurations.

These analyses are descriptive. The model types bundle provider, model family, training data, tokenizer, API implementation, effort controls, cost, context behavior, and sampling policy. Regression can estimate differences between these bundles, but it cannot identify which unobserved property explains any difference. Since the intervals include zero, we do not interpret directional differences among model types or skill-generator providers.

4.5 Token-Matched Control

A remaining alternative explanation is that skill files change behavior because they add a long, skill-formatted prefix, not because they contain useful data-science guidance. The Length-Control condition tests this explanation on the selected model subset. Pass rates are nearly identical: No-Skill passes 77.06% of paired rows, Length-Control passes 76.98%, and Full passes 76.90%.

The paired estimates provide no evidence for a reliable length-control effect. Length-Control versus No-Skill is -0.08 percentage points, with a clustered 95% CI of [-2.34, +2.22]. Full versus Length-Control is also -0.08 percentage points, with a clustered 95% CI of [-2.14, +2.06].

On the same subset, Full versus No-Skill is -0.16 percentage points, with a clustered 95% CI of [-2.42, +2.26]. Thus, the audit does not support an explanation in which additional skill-formatted context reliably changes performance, nor does it show that generated data-science content improves over task-irrelevant skill-formatted content.

This matches the main aggregate pattern: flat-file context injection does not produce a measurable pass-rate gain in this setting. We treat Length-Control as a sensitivity audit rather than a main comparison because it uses three selected agent model configurations rather than the full set of nine.

4.6 Token Overhead

Skill injection changes the resource profile of a model call even when it does not change pass rate. We measure token usage from provider-reported accounting fields. Input tokens are the most direct measure of prompt-side overhead. Output tokens, reasoning tokens, and dollar estimates also depend on provider-specific response behavior and pricing, so we treat dollar values as approximate monitoring estimates.

No-Skill prompts average 324 input tokens, while Full prompts average 2,441 input tokens. Full skill injection therefore adds 2,117 input tokens per paired call, a 7.54× input-token multiplier. Median output length changes less, from 326 tokens under No-Skill to 404 under Full.

The approximate cost per call rises from $0.0086

to $0.0127 under the evaluation runner's pricing assumptions. Since the paired Full effect is -1.03 percentage points with a clustered 95% CI of [-2.67, +0.70], the added prompt-side overhead is not accompanied by a measured aggregate accuracy gain in this setting.

4.7 No-Skill Stochasticity Check

One-shot API evaluations are not perfectly stable. The No-Skill rows provide a baseline-only estimate of this variation without additional model calls. This repetition was not part of a planned stochasticity experiment; it arose because each of the 15 generated skill-set directories contains a No-Skill condition.

For a fixed model-task cell, all 15 No-Skill rows have the same prompt hash because they contain only the task prompt and no skill text. They are therefore comparable repeated calls of the same No-Skill prompt.

Across 7,560 No-Skill outputs, corresponding to 504 model-task cells with 15 repeats per cell, 440 cells (87.30%) are pass/fail unanimous. The remaining 64 cells (12.70%, bootstrap 95% CI 9.92% to 15.67%) contain at least one pass/fail flip. A stricter instability measure, requiring at least three passes and three failures among the 15 repeats, marks 36 cells (7.14%) as unstable.

At the level most relevant to one-shot evaluation, a single No-Skill response disagrees with the 15-run cell majority in 2.63% of calls (bootstrap 95% CI 2.28% to 3.00%). Aggregate pass rates across the 15 No-Skill copies range from 71.63% to 74.40%.

The repeated calls show modest aggregate stochasticity, with uneven variation across task types. Statistical-analysis tasks have more pass/fail flips (32 of 126 cells) than data-extraction tasks (1 of 126), and flip rates differ across agent model configurations.

We use the repeated No-Skill rows as a sensitivity check, not as a replacement for the main one-shot comparisons. The audit does not measure stochasticity for skill-conditioned prompts, because those prompts also vary in skill content across generated skill sets.

Together, the Token-Matched Control and No-Skill Stochasticity Check address two measurement explanations for the flat aggregate result: added prompt length and unstable No-Skill baselines. Neither check supports those explanations, although they do not eliminate every form of stochasticity or prompt-context confounding.

5 Conclusion

This paper tested a practical hope: that an LLM-generated skill can be placed in front of a data-science task and improve a single-shot agent without further curation. In our controlled evaluation, that hope did not materialize. Across 15 regenerated skill sets, nine agent model configurations, and four task families, neither Full skills nor component ablations improved pass rates over No-Skill in the aggregate. Full skills also increased input tokens by 7.54×.

The practical implication is specific: generated flat-file skills should be validated before deployment. For these verifiable data-science tasks, generic task-family guidance did not replace task-specific checks, canonicalization, executable computation, or repair loops.

6 Limitations

The study estimates the effect of one controlled intervention: standardized, low-curation, flat-file skills inserted at prompt time. The shared generation prompt makes skills comparable across providers and supports mechanical ablations, but it is not optimized for each provider. A practitioner might iterate on the generated skill, use shorter or provider-specific instructions, or package references as selectively loaded files.

The evaluation is also single-shot. We cover many tasks, agent model configurations, and regenerated skill artifacts, but we use one skill-conditioned response per cell. The repeated No-Skill audit checks baseline stability, but it does not replace repeated sampling for every skill-conditioned prompt. The main estimates should therefore be read as single-shot effects, not majority-vote or repeated-sampling estimates.

Our token-matched control uses all tasks and generated skill sets, but only three selected agent model configurations rather than all nine. It tests whether the main pattern is explained by extra prompt length in that subset; it is not a full factorial study of prompt length.

Finally, deterministic pass/fail scoring makes the evaluation auditable, but it abstracts away partial credit and interactive repair. The results apply to execution-facing tasks with deterministic verifiers and do not test skill selection, tool-feedback loops, adaptive skill revision, or long-term memory.

7 Ethical Considerations

This study evaluates generated procedural guidance for data-science tasks. The evaluation uses synthetic or compact external-style fixtures and does not contain personal data collected from human participants. The main risk is indirect: a generated skill can encode brittle or overgeneral rules, such as inappropriate imputation or statistical-test choices, and a practitioner might reuse such guidance without task-specific validation.

We therefore evaluate skills with deterministic verifiers, report harms as well as gains, and scope the paper to controlled task completion rather than deployment advice. API-based evaluations may also change as providers update models; exact model identifiers, display labels, and preflight records are kept with the harness to support reproducibility.

References

Anonymous. 2026a. Anonymous. Non-archival workshop paper.

Anonymous. 2026b. Anonymous. Under review for non-archival workshop paper.

Anthropic. 2025. Agent skills: Overview. Accessed: 2026-04-28.

Dan Becker. 2018. Melbourne housing snapshot. Kaggle dataset.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901, Red Hook, NY, USA. Curran Associates, Inc.

Le Chen, Erhu Feng, Yubin Xia, and Haibo Chen. 2026. SkVM: Revisiting language VM for skills across heterogenous LLMs and harnesses. *Preprint*, arXiv:2604.03088.

Paulo Cortez. 2008. Student performance. UCI Machine Learning Repository.

Paulo Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis. 2009. Wine quality. UCI Machine Learning Repository.

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

Kaggle. 2012. Titanic: Machine learning from disaster. Kaggle competition dataset.

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Wen-Tau Yih, Daniel Fried, Sida Wang, and Tao Yu. 2023. DS-1000: A natural and reliable benchmark for data science code generation. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 18319–18345. PMLR.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Lan Li, Liri Fang, Bertram Ludäscher, and Vetle I. Torvik. 2025. AutoDCWorkflow: LLM-based data cleaning workflow auto-generation and benchmark. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 7766–7780, Suzhou, China. Association for Computational Linguistics.

Xiangyi Li, Wenbo Chen, Yimin Liu, Shenghan Zheng, Xiaokun Chen, Yifeng He, Yubo Li, Bingran You, Haotian Shen, Jiankai Sun, Shuyi Wang, Binxu Li, Qunhong Zeng, Di Wang, Xuandong Zhao, Yuanli Wang, Roey Ben Chaim, Zonglin Di, Yipeng Gao, Junwei He, Yizhuo He, Liqiang Jing, Luyang Kong, Xin Lan, Jiachen Li, Songlin Li, Yijiang Li, Yueqian Lin, Xinyi Liu, Xuanqing Liu, Haoran Lyu, Ze Ma, Bowei Wang, Runhui Wang, Tianyu Wang, Wengao Ye, Yue Zhang, Hanwen Xing, Yiqi Xue, Steven Dillmann, and Han-chung Lee. 2026. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *Preprint*, arXiv:2602.12670.

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. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

Microsoft. 2024. Northwind and pubs sample databases for microsoft SQL server. Sample database scripts.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Nathalia Nascimento, Everton Guimaraes, Sai Sanjna Chintakunta, and Santhosh Anitha Boominathan. 2024. LLM4DS: Evaluating large language models for data science code generation. *Preprint*, arXiv:2411.11908.

R. Quinlan. 1993. Auto MPG. UCI Machine Learning Repository.

R Core Team. 2026. *The R Datasets Package*. R Foundation for Statistical Computing.

Luis Rocha. 2024. Chinook database. Sample database for SQL Server, Oracle, MySQL, PostgreSQL, SQLite, and DB2.

Songyuan Sui, Hongyi Liu, Serena Liu, Li Li, Soo-Hyun Choi, Rui Chen, and Xia Hu. 2025. Chain-of-query: Unleashing the power of LLMs in SQL-aided table understanding via multi-agent collaboration. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 957–986, Mumbai, India. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics. Oral presentation.

Saverio Vito. 2008. Air quality. UCI Machine Learning Repository.

Jingzhe Xu, Rui Wang, Jiannan Wang, and Guoliang Li. 2026. PrepBench: How far are we from natural-language-driven data preparation? *Preprint*, arXiv:2605.08687.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 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*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Dan Zhang, Sining Zhoubian, Min Cai, Fengzu Li, Lekang Yang, Wei Wang, Tianjiao Dong, Ziniu Hu, Jie Tang, and Yisong Yue. 2025. DataSciBench: An LLM agent benchmark for data science. *Preprint*, arXiv:2502.13897.

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. *Preprint*, arXiv:2604.20087.

A Agent Model Configuration Details

Table 3 reports the configured model identifiers used by the evaluation harness. The *model label* column gives the readable label used in the paper, while *harness key* gives the internal configuration name used in result manifests. Preflight checks were run against provider APIs before evaluation; the main preflight report was created on 2026-05-24 UTC, and the token-matched control preflight report was created on 2026-05-26 UTC.

These entries document the configurations used for this study, not a guarantee that the same hosted models will remain publicly available under the same identifiers. This caveat is most relevant for provider-hosted proprietary models and preview identifiers.

B External Task Provenance

The external-style tasks are self-contained fixtures stored with the task set. They are inspired by public datasets or common schemas, but evaluation does not require downloading those resources.

Provider	Model label	Harness key	Configured API identifier
OpenAI	GPT-5.4 Mini	`gpt-5.4-mini-low`	`gpt-5.4-mini-2026-03-17`
OpenAI	GPT-5.5	`gpt-5.5-low`	`gpt-5.5-2026-04-23`
OpenAI	GPT-5.5	`gpt-5.5-xhigh`	`gpt-5.5-2026-04-23`
Google	Gemini 3.1 Flash Lite	`gemini-3.1-flash-lite-low`	`gemini-3.1-flash-lite`
Google	Gemini 3.1 Pro	`gemini-3.1-pro-low`	`gemini-3.1-pro-preview`
Google	Gemini 3.1 Pro	`gemini-3.1-pro-high`	`gemini-3.1-pro-preview`
Anthropic	Claude Haiku 4.5	`claude-haiku-4-5-no-thinking`	`claude-haiku-4-5-20251001`
Anthropic	Claude Opus 4.7	`claude-opus-4-7-low`	`claude-opus-4-7`
Anthropic	Claude Opus 4.7	`claude-opus-4-7-high`	`claude-opus-4-7`

Table 3: Agent model configuration details recorded by the harness.

Tasks	Fixture Type	Source Inspiration
DC-5, DC-6, DC-7	Data cleaning	Melbourne housing, passenger-manifest/Titanic-style records, and IoT sensor time-series formats (Becker, 2018; Kaggle, 2012).
DC-11–DC-14	Data cleaning	UCI-style Wine Quality, Auto MPG, Student Performance, and Air Quality fields (Cortez et al., 2009; Quinlan, 1993; Cortez, 2008; Vito, 2008).
SQL-5, SQL-11, SQL-12	SQL schemas	Spider-style concert/stadium schemas, Chinook music-store schemas, and Northwind customer/order schemas (Yu et al., 2018; Rocha, 2024; Microsoft, 2024).
SQL-6, SQL-7, SQL-13, SQL-14	SQL schemas	Synthetic department, product-order, employee-salary, and category/order-item schemas designed to test common relational operations.
EA-5, EA-7, EA-11–EA-13	Statistical analysis	Classic Iris, mtcars, PlantGrowth, ToothGrowth, and sleep examples (Fisher, 1936; R Core Team, 2026).
EA-6, EA-14; SR-5–SR-7, SR-11–SR-14	Statistical and reporting tasks	Synthetic external-style preference, repository, weather, clinical-trial, product, country, and inventory reports.

Table 4: External-style task provenance.