

Reasoning Under Uncertainty: Exploring Probabilistic Reasoning Capabilities of LLMs

Mobina Pournemat^{1,*†}

Keivan Rezaei^{1,†}

Gaurang Sriramanan^{1,†}

Arman Zarei^{1,†}

Hamid Eghbalzadeh²

Yang Wang²

Jiaxiang Fu²

Fuchun Peng²

Zhigang Wang²

Xue Feng²

Nima Noorshams²

Soheil Feizi^{1,†}

¹University of Maryland ²AI at Meta

Abstract

Large language models (LLMs) have demonstrated strong performance across a wide range of reasoning tasks, yet their understanding of joint and conditional probability distributions remains largely unexplored. In this work, we present a comprehensive evaluation of LLMs’ ability to reason about discrete joint and conditional probability distributions involving multiple random variables. Given observations drawn from a distribution, we evaluate models on three carefully designed tasks—mode identification, maximum likelihood estimation, and sample generation—which together capture fundamental skills required for probabilistic reasoning. Our study reveals both capabilities and limitations of current LLMs. In sample generation task, some models produce empirical distributions that closely match the target distribution despite exhibiting limited sample independence. We further observe that model performance is highly sensitive to representation: replacing explicit frequency information with raw samples leads to large accuracy drops, and even minor changes to outcome labels can further degrade performance. Together, these findings provide new insights into how LLMs reason about discrete joint and conditional probability distributions and highlight their strengths and limitations in certain probabilistic reasoning settings.

1 Introduction

Large Language Models (LLMs) have achieved impressive performance across a wide range of Natural Language Processing (NLP) tasks (Devlin et al., 2019; Raffel et al., 2020; Chowdhery et al., 2023). However, understanding how LLMs reason under

uncertainty remains a significant challenge, particularly for applications involving decision-making, forecasting, and statistical analysis (Freedman and Toni, 2025; Gu et al., 2024; Ball et al., 2024). Despite growing interest in probabilistic reasoning, major gaps persist in current evaluations. Prior work has largely focused on simplified settings, such as distributions of single random variables or binary outcomes, often relying on explicitly provided parameters or prior probabilities within the prompt (Nafar et al., 2023; Paruchuri et al., 2024; Nafar et al., 2025). In contrast, many practical scenarios involve the joint space of multiple factors, each with several possible outcomes. Our work addresses this gap by evaluating LLMs on discrete joint and conditional probability distributions over multiple random variables with multi-valued outcomes. Specifically, instead of providing explicit parameters, we supply raw observations or frequency counts, requiring models to implicitly infer the underlying distribution, closer to practical applications.

To probe these capabilities, we introduce a structured evaluation framework comprising three complementary tasks. *Mode Identification* tests whether models can recognize the dominant pattern in an outcome space, *Maximum Likelihood Estimation* tests whether models can recover the entire empirical distribution, and *Sample Generation* evaluates whether the model’s understanding of the distribution is strong enough to support generative behavior. Together, these tasks capture core operations required for probabilistic reasoning. Evaluating models on these tasks in both joint and conditional settings allows us to assess whether LLMs can accurately reason about probability distributions without finetuning on probabilistic datasets. Our experiments reveal several key insights. While models’ reasoning traces show familiarity with conditional and marginalization concepts, they often fail to perform the necessary calculations accurately,

* Correspondence to: mpournem@umd.edu

† All experiments, data collection, and processing activities were conducted by the University of Maryland, College Park (UMD). Meta was involved solely in an advisory role and no experiments, data collection or processing activities were conducted using Meta tools or within its IT environment.

resulting in a significant performance gap between joint and conditional distributions. Furthermore, while few-shot prompting is believed to be an effective strategy, usually boosting performance by 10 to 40%, in our setup, one-shot prompting with a very simple form of the task proved to be surprisingly helpful, boosting performance by over 70%. Analysis of sample generation uncovers a striking heuristic: models violate sample independence by prioritizing high-probability outcomes first, then adjusting subsequent generations to match the observed frequency ratios. Finally, we identify robustness challenges related to problem formulation; models appear to be very sensitive to the outcome labels and generally respond better to simple, minimal labels, suggesting they treat semantically meaningful labels as linguistic context rather than abstract symbols. Additionally, replacing explicit frequency counts with raw samples causes severe performance drops, over 60% as context length grows, highlighting difficulties with implicit counting and long-context reasoning. In summary, this work makes the following contributions:

- We present the first comprehensive study of evaluating LLMs on joint and conditional distributions of discrete random variables.
- We introduce a suite of structured tasks, mode identification, maximum likelihood estimation, and sample generation, to probe complementary aspects of probabilistic reasoning.
- We provide an in-depth analysis of sample generation behavior, showing that accurate distribution matching often arises from deterministic frequency-matching strategies rather than independent sampling.
- We identify key robustness challenges, including sensitivity to outcome notation and difficulties with implicit counting and long-context reasoning.

2 Related Work

Recent work has explored the ability of large language models (LLMs) to perform probabilistic reasoning in a variety of settings (Feng et al., 2024; Qiu et al., 2025; Kadavath et al., 2022). One line of work evaluates models with explicit or semi-structured inputs. (Nafar et al., 2025) study probabilistic queries over Bayesian networks expressed in natural language, where the prompts

explicitly provide prior and conditional probabilities. (Paruchuri et al., 2024) investigate probabilistic reasoning over continuous distributions of single random variables, evaluating tasks such as percentile estimation and probability calculation when distribution parameters are provided. (Nafar et al., 2023) evaluate LLMs on probabilistic logical reasoning tasks defined through linguistic rules and propose fine-tuning strategies to improve performance. (Ozturkler et al., 2022) further suggest that decomposing probabilistic reasoning into separate retrieval and aggregation stages can enhance model performance. (Freedman and Toni, 2025) investigate whether LLMs demonstrate rational and coherent probabilistic beliefs by testing their commitment to complementarity and monotonicity rules, and (Wang et al., 2025) focus on developing models for fine-grained conditional probability estimation. Relatedly, (Zhu and Griffiths, 2024) show that LLMs’ probability judgments are frequently incoherent, and (Mondal et al., 2024) find that models’ explicit answers to probability questions often diverge from the probabilities implicitly encoded in their next-token predictions during text generation. In a complementary direction, (Xia et al., 2024) leverage LLMs to construct ad hoc probabilistic models by eliciting moment constraints over LLM-proposed random variables for open-ended guesstimation questions. Several work examine models’ behavior in applied settings. (Gu et al., 2024) analyze whether LLMs can act as autonomous agents by simulating human behavior through sampling from distributions, and (Liu et al., 2024) introduce QRData, a benchmark for statistical and causal reasoning over data tables. Finally, several studies have shown that LLMs struggle with numerical reasoning, suggesting that these weaknesses can be partially mitigated through automated data generation, targeted training, or by expressing numerical reasoning as executable programs (Geva et al., 2020; Lewkowycz et al., 2022; Chen et al., 2022). Counting tasks are also challenging. (Fu et al., 2024; Ball et al., 2024), and (Zhang et al., 2024) attribute some of these failures to architectural constraints and tokenization issues in transformers (Vaswani et al., 2017).

Our findings connect to this line of work by showing that weaknesses in counting and numerical reasoning can substantially affect probabilistic reasoning. More broadly, our work differs from prior efforts by focusing on discrete joint and conditional probability distributions involving multi-

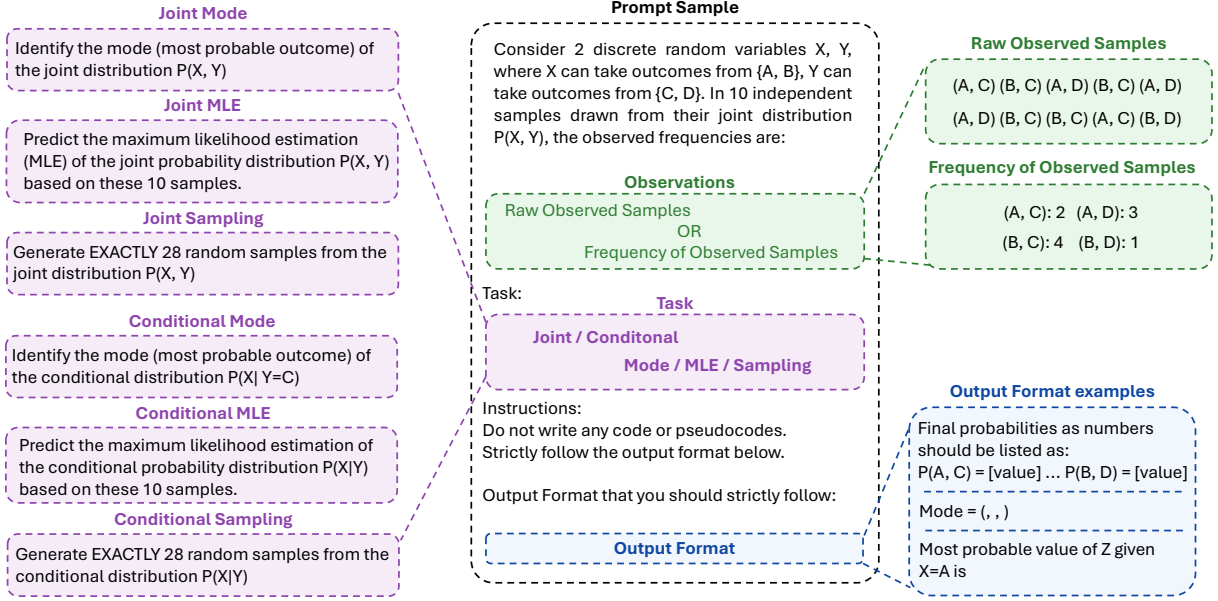


Figure 1: Prompt structure for evaluating LLMs on probabilistic reasoning tasks: each prompt includes definitions of the random variables, observed samples or frequencies, a task specification (e.g., mode, MLE, sample generation), and output formatting instructions.

ple random variables, where models must infer the underlying distribution from samples or their frequency rather than relying on explicitly provided parameters, prior probabilities, or logical rules.

3 Probabilistic Reasoning Framework

In this section, we present our framework for evaluating LLMs’ capability to understand probability distributions. We begin with the mathematical preliminaries and then describe the problem setup and prompt format. Then, we formalize the three structured tasks that form the core of our evaluation, and finally present our comprehensive evaluation pipeline. In this work, we decided to focus on discrete probability distributions, as methods and evaluation strategies for continuous distributions differ enough that addressing both types in one paper would risk superficial coverage.

3.1 Task Definition and Problem Setup

We formalize the probabilistic setup used in our study. Let $X = (X_1, X_2, \dots, X_N)$ denote a tuple of N categorical random variables, where each X_i takes values in a finite label set $\mathcal{L}_i$. We refer to the elements of $\mathcal{L}_i$ as the possible *labels* of X_i . An outcome of the joint random variable X is denoted by $\mathbf{x} = (x_1, x_2, \dots, x_N)$, where $x_i \in \mathcal{L}_i$ for all $i \in [N]$. Let $\mathcal{L} = \mathcal{L}_1 \times \mathcal{L}_2 \times \dots \times \mathcal{L}_N$ denote the set of all possible joint outcomes. The size of the

joint outcome space is then given by

$$|\mathcal{L}| = \prod_{i=1}^N |\mathcal{L}_i|.$$

We focus on mode identification, maximum likelihood estimation, and sample generation because they capture the core operations required to reason about probability distributions, including identifying frequent outcomes, estimating distribution parameters, and reproducing distributional structure. For each task, we consider both joint and conditional settings. The joint setting tests whether models can reason directly over structured outcome spaces, while the conditional setting probes a more demanding form of reasoning that requires incorporating evidence and updating beliefs.

Mode Identification: Given a set of K observations $\mathcal{O} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(K)}\}$ provided to the model, this task evaluates whether the model can identify the mode of the empirical distribution.

Joint task. We measure if the model can find the mode of the joint distribution i.e.,

$$\hat{\mathbf{x}}_{\text{mode}} = \arg \max_{\mathbf{x} \in \mathcal{L}} \frac{1}{K} \sum_{k=1}^K \mathbb{1}[\mathbf{x}^{(k)} = \mathbf{x}]$$

Conditional task. We condition on a random variable X_c and evaluate the mode of a query variable X_q . Given a conditioning value $X_c = x_c$ for

some $x_c \in \mathcal{L}_c$, we define:

$$\hat{x}_q = \arg \max_{x_q \in \mathcal{L}_q} \hat{p}(X_q = x_q \mid X_c = x_c),$$

where $\hat{p}$ is the empirical conditional distribution estimated from the subset of $\mathcal{O}$ satisfying $X_c = x_c$. We report *accuracy* for this task, indicating whether the model correctly identifies the most likely outcome.

Maximum Likelihood Estimation (MLE): Given the observed samples $\mathcal{O}$, the model is asked to estimate the empirical distribution over possible outcomes, either for the joint distribution or in a conditional manner.

Joint task: The model estimates $\hat{p}(X = \mathbf{x})$ for all $\mathbf{x} \in \mathcal{L}$. We measure the total variation distance (TVD) between the predicted empirical distribution $\hat{p}$ and the true empirical distribution p from $\mathcal{O}$, i.e.,

$$\frac{1}{2} \sum_{\mathbf{x} \in \mathcal{L}} |p(X = \mathbf{x}) - \hat{p}(X = \mathbf{x})|.$$

Conditional task: Given $X_c = x_c$, the model estimates $\hat{p}(X_q = x_q \mid X_c = x_c)$ for all $x_q \in \mathcal{L}_q$. We once again report the total variation distance for this task, i.e.,

$$\frac{1}{2} \sum_{x_q \in \mathcal{L}_q} |p(X_q = x_q \mid X_c = x_c) - \hat{p}(X_q = x_q \mid X_c = x_c)|$$

Sample Generation: In this task, the model is prompted to generate new samples from the joint or conditional distribution based on the provided observations $\mathcal{O}$. Unlike the previous tasks focused on estimation or prediction, this task evaluates the model’s generative capability, i.e., whether it can reproduce the empirical distribution through sampling.

Joint task: The model is prompted to generate K' samples based on the joint distribution provided in the context. Let $\hat{\mathcal{O}}$ be the samples generated by the LLM, and let q denote their empirical frequency distribution. We report the total variation distance, which is defined in this setting as:

$$\frac{1}{2} \sum_{\mathbf{x} \in \mathcal{L}} |p(X = \mathbf{x}) - q(X = \mathbf{x})|.$$

Conditional task: The model is prompted to generate K' samples of the random variable X_q conditioned upon $X_c = x_c$. Let $q(X_q \mid X_c = x_c)$ be the empirical frequency distribution over the samples

generated by the LLM. We report the total variation distance, which is defined in this setting as:

$$\frac{1}{2} \sum_{x_q \in \mathcal{L}_q} |p(X_q = x_q \mid X_c = x_c) - q(X_q = x_q \mid X_c = x_c)|$$

By evaluating models on joint and conditional distributions, we examine their ability to handle probabilistic relationships in various settings. Each task focuses on a specific aspect, providing a comprehensive assessment of how well LLMs understand and process discrete probability distributions.

To grant the LLM access to a distribution, we provide a set of K independent observations drawn from the joint distribution. Let $\mathcal{O} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(K)}\}$ denote the observed samples. This information is incorporated into the LLM’s context using one of two following formats: (1) by providing the empirical frequency of each possible outcome $\mathbf{x}$ in $\mathcal{O}$, or (2) by directly including the set of observations $\mathcal{O}$ in the prompt.

As shown in Figure 1, each prompt begins with a fixed segment that defines the random variables $X_1, X_2, \dots, X_N$, their corresponding label sets $\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_N$, followed by observations in one of the mentioned formats. The prompt concludes with a task-specific query and clear instructions that specify the expected output format and prohibit code generation, ensuring the evaluation focuses on the model’s core probabilistic reasoning capability. A complete list of prompts used for each task and setting is provided in Appendix A.

3.2 Evaluation Setup

As described in Section 3.1, prompts explicitly specify the required output format to encourage consistent model responses. When outputs follow the expected format, final answers are extracted using regular expressions and evaluated with the corresponding metrics. In cases where models, particularly smaller ones, produce responses that deviate from the expected format, we use a capable reference model (LLama3.3-70B) as a judge to determine whether the response reflects the correct answer. Additional details on the use of the LLM judge are provided in Appendix C.

4 Experiments

This section presents the experimental setup for evaluating LLMs on probabilistic reasoning tasks. We introduce the models evaluated, describe key implementation details, and report results for the

Model	<i>joint-mode</i>				<i>cond-mode</i>			
	$ \mathcal{L} = 12$	$ \mathcal{L} = 27$	$ \mathcal{L} = 36$	$ \mathcal{L} = 54$	$ \mathcal{L} = 12$	$ \mathcal{L} = 27$	$ \mathcal{L} = 36$	$ \mathcal{L} = 54$
Llama3.1-8B	0.59	0.57	0.35	0.21	0.83	0.79	0.68	0.62
Qwen2.5-7B	0.93	0.83	0.81	0.65	0.69	0.67	0.63	0.60
DeepSeek-R1-Distill-Qwen-7B	0.95	0.93	0.91	0.86	0.93	0.93	0.91	0.83
Llama3.3-70B	1.00	1.00	1.00	0.97	1.00	0.99	0.96	0.95
GPT-4o-mini	0.99	0.98	0.98	0.80	0.99	0.96	0.95	0.88
GPT-4.1-mini	1.00	1.00	0.99	0.96	1.00	1.00	1.00	1.00

Table 1: Accuracy comparison on mode identification tasks across different joint sizes $|\mathcal{L}|$. *cond-mode* requires reasoning over conditional distributions, making it generally more challenging.

three core tasks. We conclude with robustness analyses that test sensitivity to label choices and to providing raw samples instead of frequency counts.

4.1 Model Selection

In this study, we evaluate several instruction-finetuned LLMs, selected to represent a wide range of model sizes and capabilities. The models evaluated include Llama3.1-8B, Llama3.3-70B (Grattafiori et al., 2024), Qwen2.5-7B (Yang et al., 2024), DeepSeek-R1-Distill-Qwen-7B (Guo et al., 2025), GPT-4o-mini (Hurst et al., 2024), and GPT-4.1-mini (OpenAI, 2025). Together, these models range from 7B to 70B parameters, enabling a broad comparison of how model scale and training influence probabilistic reasoning performance.

4.2 Experimental Setup

Naming and distribution construction. We follow the standard naming convention for random variables, using names such as X, Y, Z and use English letters for the outcomes of the random variables. We set $K := 5 \times |\mathcal{L}|$, meaning the number of observations is five times larger than the joint space. We then select $|\mathcal{L}|$ values that sum to K , as frequencies, and randomly assign them to different joint outcomes. For larger $|\mathcal{L}|$, we select frequency values to keep distribution entropy comparable across settings to improve consistency of comparisons.

Controlling sensitivity to outcome–frequency assignment. Even when the label set and frequency values are fixed, models can be sensitive to which outcome receives which frequency. To reduce the impact of this surface-level variability, we create ten distinct assignments of frequencies to outcomes and generate ten prompts per assignment, resulting in 100 prompts per task. This design helps ensure that our reported results are not driven

by a specific outcome–frequency mapping.

Frequencies vs. raw samples. Because models often struggle when raw samples are provided directly, we use frequency counts in most experiments, and explicitly analyze the raw-sample setting and the comparison between the two approaches in Section 4.4.2.

Scaling task complexity. To study how performance changes as the probabilistic reasoning problem becomes harder, we vary the joint outcome space size $|\mathcal{L}|$. Increasing either the number of random variables N or the number of labels $|\mathcal{L}_i|$ for each variable causes exponential or multiplicative growth of the joint space respectively, requiring models to reason over more outcomes and probabilities. This increase affects all three tasks: mode identification requires comparing more candidates; MLE requires estimating more probabilities; and sampling requires generating outputs consistent with a more complex distribution.

4.3 Task-Specific Experimental Results

To evaluate conditional queries, one random variable X_c and one of its possible outcomes x_c are randomly selected as evidence. Another variable is then chosen as the query variable X_q and the model is asked to solve one of the tasks according to the conditional distribution $P(X_q | X_c = x_c)$. For each task and each $|\mathcal{L}|$, every model is prompted 100 times, and the outputs are parsed to extract the final answer for each query. Additional details of different distributions are provided in Appendix B.

Mode Identification: In this task, models should identify the mode of either the joint distribution (*joint-mode*) or a conditional distribution (*cond-mode*). The *joint-mode* requires selecting the most frequent outcome directly, while *cond-mode* requires filtering by a condition and reasoning

Model	joint-MLE				cond-MLE			
	$ \mathcal{L} = 12$	$ \mathcal{L} = 27$	$ \mathcal{L} = 36$	$ \mathcal{L} = 54$	$ \mathcal{L} = 12$	$ \mathcal{L} = 27$	$ \mathcal{L} = 36$	$ \mathcal{L} = 54$
Llama3.1-8B	1e-05	0.001	0.003	0.02	0.089	0.115	0.151	0.198
Qwen2.5-7B	0	2e-05	4e-04	0.006	0.077	0.106	0.140	0.177
DeepSeek-R1Distill-Qwen-7B	0	5e-04	0.001	0.02	0.038	0.048	0.069	0.100
Llama3.3-70B	0	0	0	4e-04	0.009	0.003	0.017	0.084
GPT-4o-mini	0	2e-05	7e-04	1e-04	0	0.008	0.017	0.084
GPT-4.1-mini	0	0	1e-05	1e-04	0	5e-06	5e-05	0.016

Table 2: Average TVD across 100 prompts for maximum likelihood estimation tasks at varying joint sizes $|\mathcal{L}|$. *joint-MLE* is consistently easier with near-zero TVD, while *cond-MLE* seems more challenging and highlights a clear gap between models.

over the remaining subset. Table 1 reports model accuracy as $|\mathcal{L}|$ increases. Although *cond-mode* involves fewer possible outputs, its added conditional reasoning makes it more challenging, leading to slightly lower accuracy than *joint-mode*. While Llama3.3-70B and GPT4.1-mini remain near-perfect, smaller models degrade as the joint size grows. To mitigate this drop, we evaluated models using one-shot prompting. While few-shot strategies typically yield moderate gains, we found that in our setting, a single simple demonstration can be surprisingly effective, restoring the accuracy by over 60% (see Appendix D). To better understand where this gap originates, we perform a layer-wise logit analysis, tracking how the rank of the correct answer token evolves across transformer layers. This analysis reveals that *joint-mode* answers are progressively consolidated in the final layers, whereas *cond-mode* answers are not only left unresolved but appear to be actively suppressed in the last layers (Full results in Appendix E).

Maximum Likelihood Estimation: In this task, models are asked to estimate either the full empirical joint distribution (*joint-MLE*) or the conditional probabilities $P(X_q = x_q \mid X_c = x_c)$ for each $x_q \in \mathcal{L}_q$ (*cond-MLE*). We evaluate predictions using TVD, computed for each prompt, and averaged over 100 prompts. As shown in Table 2, models can achieve low TVD values across different joint sizes in *joint-MLE*. In contrast, *cond-MLE* is more challenging, as it requires models to infer conditional probabilities from the joint distribution. This added complexity results in consistently higher TVD values and reveals a gap between smaller and larger models. We further observe a substantial performance improvement when a single demonstration is included in the context before the

main query. As reported in Appendix D, one-shot prompting reduces TVD by an average of 72% across all evaluated models in *cond-MLE* task. This result suggests that instruction finetuned LLMs do not reliably encode the probabilistic knowledge required to solve maximum likelihood estimation—a fundamental and widely used problem in statistics—but can rapidly learn the necessary skill when provided with an explicit solved example.

Sample Generation: This task evaluates the generative abilities of LLMs by requiring them to generate K' samples from either the joint distribution (*joint-sampling*) or a conditional distribution (*cond-sampling*). We set $K' := 7 \times |\mathcal{L}|$ to ensure that the generated samples are sufficient to approximate the target distribution. Since it is inherently difficult to reproduce a target empirical distribution with a finite number of samples, we also compute the TVD between K' samples drawn from Python’s random library and the true distribution and use it as a baseline to compare the quality of samples generated by the models.

As shown in Figure 2, TVD in the *joint-sampling* task increases with $|\mathcal{L}|$. Interestingly, Llama3.3-70B and GPT4.1-mini achieve TVD values even lower than those from Python’s random sampler, which includes sampling noise, suggesting that the underlying distribution of their generated samples is remarkably close to the target. In contrast, all models struggle with the *cond-sampling* task, as it implicitly requires the models to first estimate the conditional probabilities and then generate samples that align with the resulting distribution. Although no model reaches an acceptable performance, DeepSeek-R1-Distill-Qwen, designed for reasoning tasks, outperforms others.

Beyond distributional accuracy, we also exam-

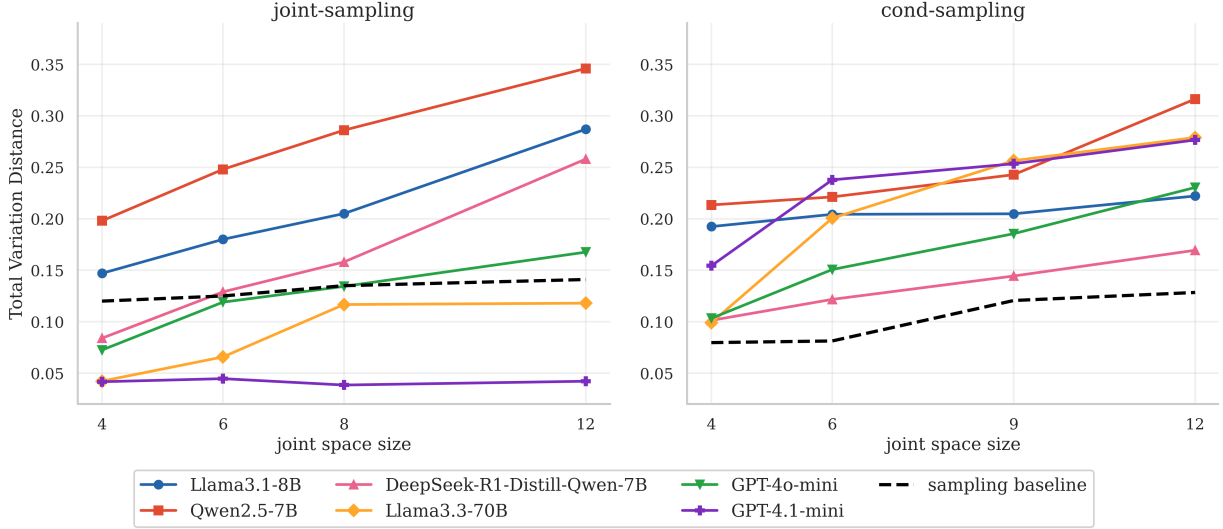


Figure 2: Average TVD across 100 prompts for *joint-sampling* (left) and *cond-sampling* (right) tasks, with models queried to generate $7 \times |\mathcal{L}|$ samples. Here, the sampling baseline is computed as the TVD between K' samples generated by Python’s random function and the true distribution.

ine whether the generated samples are independent. Using the Durbin–Watson statistic and transition matrix entropy (Durbin and Watson, 1951; Shannon, 1948), we observe clear dependencies in the outputs of all models. Generated samples sometimes follow ordering patterns similar to those seen in the input prompt. When inspecting chain-of-thought reasoning, some models explicitly mention the need to shuffle samples, but in practice they prioritize matching target frequencies due to the lack of access to true randomness. (see Appendix F) To further probe this behavior, we prompted GPT-4.1-mini to generate one sample at a time across N separate prompts. Interestingly, the model overwhelmingly favors outcomes with the top three frequencies and almost never produces low-frequency outcomes. Independence tests further suggest that the model does not treat each generation as an independent draw. Instead, it appears to keep track of previously generated samples and adjust future outputs accordingly. This behavior is consistent with the autoregressive nature of LLMs, where each generation is conditioned on prior outputs, making truly independent sampling difficult. Finally, to validate that these findings are not limited to synthetic data, we repeat all six task configurations on a of the Mushroom dataset (mus, 1981). The results closely mirror those observed in the synthetic setting (see Appendix G).

4.4 Robustness Analysis Results

We now evaluate two robustness questions that probe how sensitive model behavior is to (i) superficial changes in label notation and (ii) providing raw samples rather than frequency counts. These analyses shed light on how representation choices can significantly affect performance.

4.4.1 Label Sensitivity Results

We study how sensitive model performance is to the choice of outcome labels by running a controlled set of experiments on three models: Llama3.3-70B, GPT-4.1-mini, and Qwen2.5-7B. All models are evaluated on the *joint-mode*, which they handle reliably under standard conditions, with a fixed joint space size of $|\mathcal{L}| = 54$, while varying only the type of labels used to represent outcomes. We consider ten label categories, including computer science terms, human names, US states, countries, fruits, universities, majors, car brands, neurology terms, and food.

Figure 3 (right) shows the accuracy of each model across label categories. GPT-4.1-mini remains highly consistent, indicating strong robustness to label choice. Llama3.3-70B performs well overall but shows noticeable variation across categories. In contrast, Qwen2.5-7B is much more sensitive, with large drops in accuracy across different labels. Our results suggest that some models generally respond better to simple, minimal labels and tend to interpret meaningful labels as part of the linguistic context rather than interchangeable

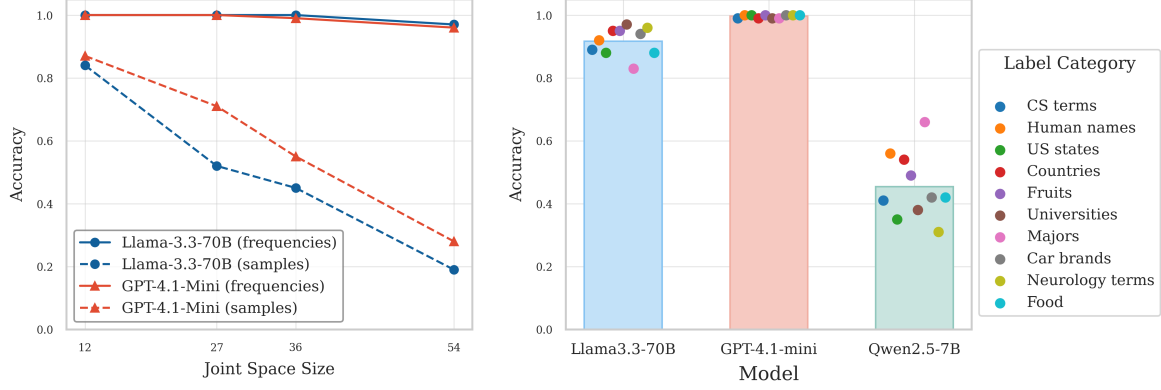


Figure 3: **(Left)** Accuracy comparison on the *joint-mode* task when models are given either raw samples or frequency counts. Accuracy sharply declines with increasing joint space size when raw samples are used, revealing the difficulty of reasoning over longer contexts without explicit frequency information. GPT-4.1-mini shows greater robustness than Llama3.3-70B in this setting. **(Right)** Accuracy of models across ten different label categories in the *joint-mode* task. GPT-4.1-mini remains consistently robust across categories, while Qwen2.5-7B exhibits significant sensitivity.

symbols in a probabilistic space.

4.4.2 Samples-in-Context Results

We compare the standard frequency-based prompts with a more challenging setting in which the raw observed samples are given directly. In this setup, the model must first infer outcome frequencies by implicitly counting occurrences in the prompt before solving the task. As the number of samples K increases, the input becomes longer and tracking the frequencies will be more difficult. In this setting, we focus our analysis on the two strongest models, Llama3.3-70B and GPT-4.1-mini, as smaller models consistently fail under this condition. We evaluate both models on the *joint-mode* task across different joint space sizes, providing raw samples instead of explicit frequency counts.

As shown in Figure 3 (left), accuracy for both models drops almost linearly as $|\mathcal{L}|$ increases and the prompts grow longer. This trend highlights implicit counting and long-context frequency tracking as key bottlenecks, even for large and capable models. Between the two, GPT-4.1-mini consistently outperforms Llama3.3-70B, suggesting greater robustness to increased input length and task complexity. Additional evidence shows that even for fixed $|\mathcal{L}| = 12$, increasing the number of samples K results in degradation of accuracy as the context length grows. This indicates that the degradation is not driven solely by task complexity, but also by limited capacity to reason reliably over long sequences (see Appendix H). Finally,

we explore a mitigation strategy by allowing GPT-4.1-mini to use a code interpreter and explicitly count frequencies via Python before answering, unlike our main experiments. This setting evaluates *system-level* capability, i.e., what becomes possible when the model is augmented with external computation. This hybrid approach leads to a substantial performance improvement and nearly recovers the accuracy observed in the frequency-based setting (see Appendix I).

5 Conclusion

This work presents the first comprehensive evaluation of LLMs on probabilistic tasks involving joint and conditional distributions of discrete random variables. We evaluate LLMs on core statistical tasks—mode identification, maximum likelihood estimation, and sample generation—under settings where observations are provided to help model infer the underlying distribution. Our results show that LLMs do not reliably encode the probabilistic knowledge required to solve these fundamental problems in zero-shot settings, particularly for conditional distributions where accurate marginalization is critical. However, models can rapidly acquire the necessary reasoning skills through minimal in-context supervision. Our analysis of sample generation reveals that some models are able to generate samples whose empirical distributions closely match the target; this accuracy often arises from deterministic strategies that aim

to match expected frequencies rather than from truly stochastic sampling. In practice, models appear to keep track of previously generated samples and adjust subsequent outputs accordingly, reflecting the autoregressive nature of LLMs. We further validate these observations on a real-world dataset, showing that our findings are not limited to synthetic settings. Finally, our robustness analyses demonstrate that probabilistic reasoning in LLMs is highly sensitive to problem formulation; changing outcome labels or replacing explicit frequencies with raw samples can lead to large performance drops, suggesting that current LLMs often rely on representation-specific heuristics rather than abstract probabilistic reasoning. Addressing these challenges is a key step toward developing LLMs capable of robust reasoning under uncertainty.

Limitations

This study can be extended in several directions, each of which would require substantial time, effort, and computational resources. To keep the scope focused and the experiments feasible, we limited our investigation to a subset of these directions and examined them in depth. One limitation concerns the scale of the joint distributions considered in our experiments. We cap the joint space size at 54, as smaller models struggled with longer contexts, leading to unstable and unreliable behavior. Extending to larger joint spaces would further degrade performance for these models and make fair comparisons across model sizes more difficult. In addition, our focus in this work is on the joint and conditional distributions of discrete random variables. While there is existing work on LLMs' behavior over continuous probability distributions, to the best of our knowledge, no prior work has systematically studied probabilistic reasoning in discrete setting. We decided to focus on discrete distributions, and we believe this gap deserves focused attention as a foundational problem that presents distinct challenges from continuous distributions. Finally, we limited our experiments to a subset of discrete distributions, as covering a broader range of joint and conditional structures would require substantially more computational resources and experimental effort. Exploring the mentioned directions remains an important direction for future work.

Acknowledgements

This project was supported in part by a grant from an NSF CAREER AWARD 1942230, the ONR PECASE grant N00014-25-1-2378, ARO's Early Career Program Award 310902-00001, Army Grant No. W911NF2120076, the NSF award CCF2212458, NSF Award No. 2229885 (NSF Institute for Trustworthy AI in Law and Society, TRAILS), a MURI grant 14262683, DARPA AIQ grant HR00112590066, an award from Google Research and an award from meta 314593-00001.

References

1981. Mushroom. UCI Machine Learning Repository. DOI: <https://doi.org/10.24432/C5959T>.
- Thomas Ball, Shuo Chen, and Cormac Herley. 2024. Can we count on llms? the fixed-effect fallacy and claims of gpt-4 capabilities. *arXiv preprint arXiv:2409.07638*.
- Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.
- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and 1 others. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.
- J. Durbin and G. S. Watson. 1951. Testing for serial correlation in least squares regression ii. *Biometrika*, 38(1/2):159–178.
- Yu Feng, Ben Zhou, Weidong Lin, and Dan Roth. 2024. Bird: A trustworthy bayesian inference framework for large language models. *arXiv preprint arXiv:2404.12494*.
- Gabriel Freedman and Francesca Toni. 2025. Exploring the potential for large language models to demonstrate rational probabilistic beliefs. *arXiv preprint arXiv:2504.13644*.
- Tairan Fu, Raquel Ferrando, Javier Conde, Carlos Arriaga, and Pedro Reviriego. 2024. Why do large language models (llms) struggle to count letters? *arXiv preprint arXiv:2412.18626*.

- Mor Geva, Ankit Gupta, and Jonathan Berant. 2020. Injecting numerical reasoning skills into language models. *arXiv preprint arXiv:2004.04487*.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Jia Gu, Liang Pang, Huawei Shen, and Xueqi Cheng. 2024. Do llms play dice? exploring probability distribution sampling in large language models for behavioral simulation. *arXiv preprint arXiv:2404.09043*.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shitong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.
- Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, and 1 others. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.
- Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.
- Xiao Liu, Zirui Wu, Xueqing Wu, Pan Lu, Kai-Wei Chang, and Yansong Feng. 2024. Are llms capable of data-based statistical and causal reasoning? benchmarking advanced quantitative reasoning with data. *arXiv preprint arXiv:2402.17644*.
- Manuel Mondal, Ljiljana Dolamic, G r me Bovet, Philippe Cudr -Mauroux, and Julien Audiffren. 2024. Implicit probabilistic reasoning does not reflect explicit answers in large language models. *arXiv preprint arXiv:2406.14986*.
- Aliakbar Nafar, Kristen Brent Venable, and Parisa Kordjamshidi. 2023. Teaching probabilistic logical reasoning to transformers. *arXiv preprint arXiv:2305.13179*.
- Aliakbar Nafar, Kristen Brent Venable, and Parisa Kordjamshidi. 2025. Reasoning over uncertain text by generative large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24911–24920.
- OpenAI. 2025. Gpt-4.1-mini model documentation (2025-04-14). <https://platform.openai.com/docs/models/gpt-4.1-mini>. Accessed: 2025-09-23.
- Batu Ozturkler, Nikolay Malkin, Zhen Wang, and Nebojsa Jojic. 2022. Thinksum: Probabilistic reasoning over sets using large language models. *arXiv preprint arXiv:2210.01293*.
- Akshay Paruchuri, Jake Garrison, Shun Liao, John Hernandez, Jacob Sunshine, Tim Althoff, Xin Liu, and Daniel McDuff. 2024. What are the odds? language models are capable of probabilistic reasoning. *arXiv preprint arXiv:2406.12830*.
- Linlu Qiu, Fei Sha, Kelsey Allen, Yoon Kim, Tal Linzen, and Sjoerd van Steenkiste. 2025. Bayesian teaching enables probabilistic reasoning in large language models. *arXiv preprint arXiv:2503.17523*.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.
- Claude Elwood Shannon. 1948. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.
- Liaoyaqi Wang, Zhengping Jiang, Anqi Liu, and Benjamin Van Durme. 2025. Always tell me the odds: Fine-grained conditional probability estimation. *arXiv preprint arXiv:2505.01595*.
- Shepard Xia, Brian Lu, and Jason Eisner. 2024. Let’s think var-by-var: Large language models enable ad hoc probabilistic reasoning. *arXiv preprint arXiv:2412.02081*.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.
- Xiang Zhang, Juntao Cao, and Chenyu You. 2024. Counting ability of large language models and impact of tokenization. *arXiv preprint arXiv:2410.19730*.
- Jian-Qiao Zhu and Thomas L Griffiths. 2024. Incoherent probability judgments in large language models. *arXiv preprint arXiv:2401.16646*.

Appendix

This appendix provides additional details, prompts, and experimental results that complement the main text. Section A presents full prompt examples for each task, and Section B lists the exact frequency distributions used across different joint space sizes. Section C describes the LLM judge prompt template used for output evaluation. Section D reports results from one-shot prompting experiments, while Section F analyzes the independence of samples generated by LLMs. Section G presents evaluations on a real-world dataset to assess generalizability. Section H provides additional results on the effect of the number of samples included in the context, and Section I details experiments in which GPT-4.1-mini was given access to a code interpreter.

A Task Prompts

Each task is evaluated using two prompt variants: one that queries the joint distribution and another that queries a conditional distribution. We design three structured tasks to assess different aspects of probabilistic reasoning. This section presents representative prompt examples used for each task.

A.1 Mode Identification

Below is an example prompt used for the *joint-mode* task with $|\mathcal{L}| = 12$:

Consider 3 discrete random variables X, Y, Z , where X can take outcomes from A, B , Y can take outcomes from C, D , Z can take outcomes from E, F, G . In 60 independent samples drawn from their joint distribution $P(X, Y, Z)$, the observed frequencies are:

- (A, C, E): 5
- (A, C, F): 7
- (A, C, G): 3
- (A, D, E): 10
- (A, D, F): 2
- (A, D, G): 3
- (B, C, E): 6
- (B, C, F): 8
- (B, C, G): 2
- (B, D, E): 3
- (B, D, F): 3
- (B, D, G): 8

Task: Identify the mode (most probable outcome) of the joint distribution $P(X, Y, Z)$.

Instructions:

1. Do not write any code or pseudocode.
2. Strictly follow the output format below.
3. You may explain your reasoning, but the final answer should be explicitly summarized at the end. You get negative penalty for not following the output format.

Output Format that you should strictly follow:

Mode = (, ,)

Below is a variation of the above prompt that requires solving the task on a conditional distribution (*cond-mode*).

Consider 3 discrete random variables X, Y, Z , where X can take outcomes from A, B , Y can take outcomes from C, D , Z can take outcomes from E, F, G . In 60 independent samples drawn from their joint distribution $P(X, Y, Z)$, the observed frequencies are:

- (A, C, E): 5
- (A, C, F): 7
- (A, C, G): 3
- (A, D, E): 10
- (A, D, F): 2
- (A, D, G): 3
- (B, C, E): 6
- (B, C, F): 8
- (B, C, G): 2
- (B, D, E): 3
- (B, D, F): 3
- (B, D, G): 8

Task: Identify the mode (most probable outcome) of the conditional distribution $P(Z | X = A)$.

Instructions:

1. Do not write any code or pseudocode.
2. Strictly follow the output format below.
3. You may explain your reasoning, but the final answer should be explicitly summarized at the end. You get negative penalty for not following the output format.

Output Format that you should strictly follow:

Most probable value of Z given $X=A$ is

A.2 Maximum Likelihood Estimation

The model is asked to estimate the probabilities of the joint distributions using maximum likelihood estimates from observed frequencies.

A prompt sample for *joint-MLE* task for $|\mathcal{L}| = 12$:

Consider 3 discrete random variables X, Y, Z , where X can take outcomes from A, B , Y can take outcomes from C, D , Z can take outcomes from E, F, G . In 60 independent samples drawn from their joint distribution $P(X, Y, Z)$, the observed frequencies are:

- (A, C, E) : 5
- (A, C, F) : 7
- (A, C, G) : 3
- (A, D, E) : 10
- (A, D, F) : 2
- (A, D, G) : 3
- (B, C, E) : 6
- (B, C, F) : 8
- (B, C, G) : 2
- (B, D, E) : 3
- (B, D, F) : 3
- (B, D, G) : 8

Task: Predict the maximum likelihood estimation (MLE) of the joint probability distribution $P(X, Y, Z)$ based on these 60 samples.

Instructions:

1. Think step by step and solve this mathematically using probability theory - do not write any code or pseudocodes as you get negative penalty for that.
2. Clearly state the final estimated probabilities for each (X, Y, Z) outcome. Output probabilities should be expressed as float numbers with up to four decimal points.
3. You may explain your reasoning, but the final answer should be explicitly summarized at the end. only your final answer will be graded.

Output Format:

- Final probabilities as float numbers should be

listed as:

$P(A, C, E) = [\text{value}]$
 $P(A, C, F) = [\text{value}]$
 $P(A, C, G) = [\text{value}]$
 $P(A, D, E) = [\text{value}]$
...
 $P(B, D, F) = [\text{value}]$
 $P(B, D, G) = [\text{value}]$

Below is a variation of the above prompt that requires solving the task on a conditional distribution (*cond-MLE*).

Consider 3 discrete random variables X, Y, Z , where X can take outcomes from A, B , Y can take outcomes from C, D , Z can take outcomes from E, F, G . In 60 independent samples drawn from their joint distribution $P(X, Y, Z)$, the observed frequencies are:

- (A, C, E) : 5
- (A, C, F) : 7
- (A, C, G) : 3
- (A, D, E) : 10
- (A, D, F) : 2
- (A, D, G) : 3
- (B, C, E) : 6
- (B, C, F) : 8
- (B, C, G) : 2
- (B, D, E) : 3
- (B, D, F) : 3
- (B, D, G) : 8

Task: Predict the maximum likelihood estimation (MLE) of the conditional distribution $P(X | Z = E)$.

Instructions:

1. Think step by step and solve this mathematically using probability theory - do not write any code or pseudocodes as you get negative penalty for that.
2. Clearly state the final estimated probabilities as float numbers with up to four decimal points.

3. You may explain your reasoning, but the final answer should be explicitly summarized at the end.

Output Format:

- Final probabilities as float numbers should be listed as:

$P(X=A \mid Z=E) = [\text{value}]$

$P(X=B \mid Z=E) = [\text{value}]$

A.3 Sample Generation

In this task, the model is asked to generate samples either from the joint distribution or from a conditional distribution, and we compare the empirical distribution of generated samples with the true distribution.

A prompt sample for *joint-sampling* task for $|\mathcal{L}| = 12$:

Consider 3 discrete random variables X, Y, Z, where X can take outcomes from A, B, Y can take outcomes from C, D, Z can take outcomes from E, F, G. In 60 independent samples drawn from their joint distribution $P(X, Y, Z)$, the observed frequencies are:

- (A, C, E): 5
- (A, C, F): 7
- (A, C, G): 3
- (A, D, E): 10
- (A, D, F): 2
- (A, D, G): 3
- (B, C, E): 6
- (B, C, F): 8
- (B, C, G): 2
- (B, D, E): 3
- (B, D, F): 3
- (B, D, G): 8

Task: Generate EXACTLY 84 random samples from the joint distribution $P(X, Y, Z)$.

Instructions:

1. OUTPUT MUST BEGIN with ### Output on a new line.
2. List EXACTLY 84 samples numbered 1 to 84.

3. Do not write any code or pseudocode.
4. Strictly follow the output format below.
5. You will be penalized for writing codes or not following the output format.

Output Format that you should strictly follow:

Output

1. (X, Y, Z)

2. (X, Y, Z)

...

84. (X, Y, Z)

Where X can take outcomes from A, B, Y can take outcomes from C, D, Z can take outcomes from E, F, G.

Below is a variation of the above prompt that requires solving the task on a conditional distribution (*cond-sampling*).

Consider 3 discrete random variables X, Y, Z, where X can take outcomes from A, B, Y can take outcomes from C, D, Z can take outcomes from E, F, G. In 60 independent samples drawn from their joint distribution $P(X, Y, Z)$, the observed frequencies are:

- (A, C, E): 5
- (A, C, F): 7
- (A, C, G): 3
- (A, D, E): 10
- (A, D, F): 2
- (A, D, G): 3
- (B, C, E): 6
- (B, C, F): 8
- (B, C, G): 2
- (B, D, E): 3
- (B, D, F): 3
- (B, D, G): 8

Task: Generate EXACTLY 84 random samples from the conditional distribution $P(Z \mid X=A)$.

Instructions:

1. OUTPUT MUST BEGIN with ### Output on a new line.
2. List EXACTLY 84 samples numbered 1 to 84.

84.
 3. Do not write any code or pseudocodes.
 4. Strictly follow the output format below.
 5. You will be penalized for writing codes or not following the output format.

Output Format that you should strictly follow:

Output

1. Z

2. Z

...

84. Z

Where Z can take outcomes from E, G, F.

B Distributions Used for Each Joint Space Size

Distribution used for $|\mathcal{L}| = 4$

("A", "C"): 6, ("A", "D"): 4, ("B", "C"): 8, ("B", "D"): 2

Distribution used for $|\mathcal{L}| = 6$

("A", "C"): 7, ("A", "D"): 2, ("A", "E"): 4, ("B", "C"): 9, ("B", "D"): 2, ("B", "E"): 6

Distribution used for $|\mathcal{L}| = 8$

("A", "C", "E"): 7, ("A", "C", "F"): 4, ("A", "D", "E"): 9, ("A", "D", "F"): 2, ("B", "C", "E"): 4, ("B", "C", "F"): 3, ("B", "D", "E"): 6, ("B", "D", "F"): 5

Distribution used for $|\mathcal{L}| = 9$

("A", "D"): 6, ("A", "E"): 3, ("A", "F"): 4, ("B", "D"): 9, ("B", "E"): 2, ("B", "F"): 5, ("C", "D"): 4, ("C", "E"): 3, ("C", "F"): 4

Distribution used for $|\mathcal{L}| = 12$

("A", "C", "E"): 8, ("A", "C", "F"): 2, ("A", "C", "G"): 7, ("A", "D", "E"): 3, ("A", "D", "F"): 3, ("A", "D", "G"): 3, ("B", "C", "E"): 6, ("B", "C", "F"): 10, ("B", "C", "G"): 2, ("B", "D", "E"): 8, ("B", "D", "F"): 5, ("B", "D", "G"): 3,

Distribution used for $|\mathcal{L}| = 27$

("A", "D", "G"): 8, ("A", "D", "H"): 4, ("A", "D", "I"): 2, ("A", "E", "G"): 3, ("A", "E", "H"): 4,

("A", "E", "I"): 5, ("A", "F", "G"): 6, ("A", "F", "H"): 3, ("A", "F", "I"): 6, ("B", "D", "G"): 10, ("B", "D", "H"): 6, ("B", "D", "I"): 1, ("B", "E", "G"): 7, ("B", "E", "H"): 2, ("B", "E", "I"): 4, ("B", "F", "G"): 6, ("B", "F", "H"): 6, ("B", "F", "I"): 3, ("C", "D", "G"): 4, ("C", "D", "H"): 5, ("C", "D", "I"): 6, ("C", "E", "G"): 7, ("C", "E", "H"): 4, ("C", "E", "I"): 2, ("C", "F", "G"): 3, ("C", "F", "H"): 8, ("C", "F", "I"): 5,

Distribution used for $|\mathcal{L}| = 36$

("A", "E", "H"): 6, ("A", "E", "I"): 8, ("A", "E", "J"): 13, ("A", "F", "H"): 9, ("A", "F", "I"): 1, ("A", "F", "J"): 2, ("A", "G", "H"): 11, ("A", "G", "I"): 2, ("A", "G", "J"): 1, ("B", "E", "H"): 15, ("B", "E", "I"): 4, ("B", "E", "J"): 8, ("B", "F", "H"): 1, ("B", "F", "I"): 5, ("B", "F", "J"): 5, ("B", "G", "H"): 1, ("B", "G", "I"): 7, ("B", "G", "J"): 2, ("C", "E", "H"): 4, ("C", "E", "I"): 5, ("C", "E", "J"): 12, ("C", "F", "H"): 3, ("C", "F", "I"): 2, ("C", "F", "J"): 2, ("C", "G", "H"): 4, ("C", "G", "I"): 8, ("C", "G", "J"): 12, ("D", "E", "H"): 2, ("D", "E", "I"): 4, ("D", "E", "J"): 2, ("D", "F", "H"): 1, ("D", "F", "I"): 2, ("D", "F", "J"): 2, ("D", "G", "H"): 9, ("D", "G", "I"): 2, ("D", "G", "J"): 3

Distribution used for $|\mathcal{L}| = 54$

("A", "D", "F", "I"): 1, ("A", "D", "F", "J"): 5, ("A", "D", "F", "K"): 9, ("A", "D", "G", "I"): 1, ("A", "D", "G", "J"): 10, ("A", "D", "G", "K"): 2, ("A", "D", "H", "I"): 3, ("A", "D", "H", "J"): 2, ("A", "D", "H", "K"): 5, ("A", "E", "F", "I"): 1, ("A", "E", "F", "J"): 1, ("A", "E", "F", "K"): 5, ("A", "E", "G", "I"): 5, ("A", "E", "G", "J"): 12, ("A", "E", "G", "K"): 2, ("A", "E", "H", "I"): 6, ("A", "E", "H", "J"): 4, ("A", "E", "H", "K"): 2, ("B", "D", "F", "I"): 10, ("B", "D", "F", "J"): 2, ("B", "D", "F", "K"): 1, ("B", "D", "G", "I"): 7, ("B", "D", "G", "J"): 2, ("B", "D", "G", "K"): 3, ("B", "D", "H", "I"): 8, ("B", "D", "H", "J"): 2, ("B", "D", "H", "K"): 7, ("B", "E", "F", "I"): 2, ("B", "E", "F", "J"): 2, ("B", "E", "F", "K"): 3, ("B", "E", "G", "I"): 4, ("B", "E", "G", "J"): 5, ("B", "E", "G", "K"): 13, ("B", "E", "H", "I"): 2, ("B", "E", "H", "J"): 4, ("B", "E", "H", "K"): 12, ("C", "D", "F", "I"): 12, ("C", "D", "F", "J"): 1,

("C", "D", "F", "K"): 3, ("C", "D", "G", "I"): 1, ("C", "D", "G", "J"): 15, ("C", "D", "G", "K"): 9, ("C", "D", "H", "I"): 11, ("C", "D", "H", "J"): 1, ("C", "D", "H", "K"): 4, ("C", "E", "F", "I"): 1, ("C", "E", "F", "J"): 5, ("C", "E", "F", "K"): 3, ("C", "E", "G", "I"): 11, ("C", "E", "G", "J"): 13, ("C", "E", "G", "K"): 3, ("C", "E", "H", "I"): 7, ("C", "E", "H", "J"): 4, ("C", "E", "H", "K"): 1

C LLM Judge Prompt

We used the LLM judge only for the mode identification task, where model responses were often difficult to evaluate using rule-based methods. Many responses implied an answer but did not state it clearly or follow a consistent format, making extraction with regular expressions unreliable. For the other two tasks—MLE and sampling—models typically followed the expected output format, or when they did not, their responses clearly lacked an answer, making them easy to discard.

To ensure the judge’s reliability, we compared its decisions with human judgments on 100 randomly selected responses. The judge matched human decisions in 99 cases, indicating its reliable judgments.

To automate evaluation, the judge model receives the task, expected output, and model response, and decides whether the response is correct.

Below, is the **system prompt** used for our judge LLM, Llama3.3-70B

You are an expert evaluator assessing the correctness of responses. Your task is to judge whether a response is correct based on an expected answer. If the response correctly conveys the intended meaning, mark it Correct. Otherwise, mark it Incorrect.

The template of the **user prompt** that is provided to the judge model is given below:

Given the following question, expected answer, and response, judge whether the response is correct. Clearly state your judgment as "Judgment: Correct." or "Judgment: Incorrect."

Question: [question]

Expected Answer: [expected]

Response to Evaluate: [response]

Judgment (Correct/Incorrect):

D One-shot Prompting

As illustrated in Tables 1 and 2, models tend to struggle on the more complex tasks when relying solely on their pretraining knowledge. To address this, we designed a set of experiments in which each prompt was augmented with a single, simpler example of the task. Specifically, we included an example with $|\mathcal{L}| = 12$ along with its step-by-step solution, followed by the target, more challenging task with $|\mathcal{L}| = 54$, and asked the model to solve it.

As shown in Table 3, this one-shot prompting strategy substantially improved the models’ accuracy, highlighting the impact of providing minimal guidance on solving the harder problems. Interestingly, we observed that the performance of smaller models with just one in-context example closely matched the zero-shot performance of much larger models, underscoring the strength of in-context learning as a way to compensate for smaller model size.

An example of the one-shot prompt used in our experiments is provided below.

Task Definition: Consider the joint probability distribution of a set of discrete random variables. You will be provided with the frequencies of each outcome of the joint distribution in a set of independently drawn samples. Your task is to identify the mode (the most probable outcome) of a specific conditional distribution.

— EXAMPLE with Solution—

Consider discrete random variables X , Y , Z with joint distribution $P(X, Y, Z)$ where X can take outcomes from A, B , Y can take outcomes from C, D , Z can take outcomes from E, F, G . In 60 independent samples drawn from the joint distribution $P(X, Y, Z)$, the observed frequencies are:

- (A, C, E): 5
- (A, C, F): 7
- (A, C, G): 3
- (A, D, E): 10
- (A, D, F): 2

Table 3: Performance comparison of different models in zero-shot vs one-shot settings across four tasks with $|\mathcal{L}| = 54$. Mode identification tasks are evaluated using accuracy, while MLE tasks are evaluated using total variation distance (TVD).

Model	<i>joint-mode</i>		<i>cond-mode</i>		<i>joint-MLE</i>		<i>cond-MLE</i>	
	zero-shot	one-shot	zero-shot	one-shot	zero-shot	one-shot	zero-shot	one-shot
Llama3.1-8B	0.21	0.88	0.62	0.93	0.02	0	0.198	0.082
Qwen2.5-7B	0.65	0.98	0.60	0.87	0.006	6e-05	0.177	0.041
DeepSeek-R1Distill-Qwen-7B	0.86	0.9	0.83	0.95	0.02	0.001	0.100	0.09
Llama3.3-70B	0.97	1.0	0.95	1.0	4e-04	0	0.084	0.004
GPT-4o-mini	0.80	1.0	0.88	1.0	1e-04	0	0.084	0.003
GPT-4.1-mini	0.96	1.0	1.0	1.0	1e-04	0	0.016	3e-04

- (A, D, G): 3
- (B, C, E): 6
- (B, C, F): 8
- (B, C, G): 2
- (B, D, E): 3
- (B, D, F): 3
- (B, D, G): 8

Example Task (solved): Identify the mode (most probable outcome) of the conditional distribution $P(Z | X = A)$.

Step-by-step solution:

1. Extract rows where $X = A$:
(A,C,E):5, (A,C,F):7, (A,C,G):3, (A,D,E):10, (A,D,F):2, (A,D,G):3.

2. Aggregate counts by Z among those rows:

- Count($Z = E | X = A$) = 5 + 10 = 15.
- Count($Z = F | X = A$) = 7 + 2 = 9.
- Count($Z = G | X = A$) = 3 + 3 = 6.

3. Total samples with $X = A = 15 + 9 + 6 = 30$.

4. Compute empirical conditional probabilities.

- $P(Z = E | X = A) = 15 / 30 = 0.5$
- $P(Z = F | X = A) = 9 / 30 = 0.3$
- $P(Z = G | X = A) = 6 / 30 = 0.2$

5. Identify the mode: $Z = E$ (highest probability 0.5).

Final Answer:

Most probable value of Z given $X=A$ is E
— END EXAMPLE —

Main Task:

Consider 4 discrete random variables X, Y, Z, T , where X can take outcomes from A, B, C, Y can take outcomes from D, E, Z can take outcomes from F, G, H, T can take outcomes from I, J, K.

In 270 independent samples drawn from their joint distribution $P(X, Y, Z, T)$, the observed frequencies are:

- (A, D, F, I): 6
- (A, D, F, J): 3
- ...
- (C, E, H, I): 4
- (C, E, H, J): 2
- (C, E, H, K): 1

Main Task (to solve): Identify the mode (most probable outcome) of the conditional distribution $P(X | Y = E)$.

Instructions:

1. Do not write any code or pseudocodes.
2. Strictly follow the output format below.
3. You may explain your reasoning, but the final answer should be explicitly summarized at the end. You get negative penalty for not following the output format.

Output Format that you should strictly follow:
Most probable value of X given $Y=E$ is

E Logit Analysis of Internal Reasoning

To probe how probabilistic reasoning unfolds inside the model rather than only at its output, we conduct a logit analysis on Qwen2.5-7B, which we selected because we have direct access to its hidden states and LM head, unlike the closed-source models in our evaluation. We take the hidden state at the answer token position at certain layers of the transformer, project it through the final LM head, and record the rank of the correct answer token within the resulting vocabulary distribution at that layer (rank 1 indicates the correct token is the top prediction). We repeat this for both *joint-mode* and *cond-mode* across all four joint space sizes $|\mathcal{L}|$ used in our main experiments, and report the rank trajectory at three representative depths: an early-late layer (L20), a late layer (L24), and the final layer (L28).

Table 4 reports the results. For *joint-mode*, the correct outcome is ranked in the thousands to tens of thousands at layer 20, but consistently collapses to single digits by layer 28 — a three-order-of-magnitude improvement concentrated almost entirely in the last transformer layers. This pattern is consistent with the model progressively consolidating the correct joint outcome into its final hidden representation immediately before generation, mirroring the near-perfect surface-level accuracy we observe for *joint-mode* in Table 1.

The picture for *cond-mode* is very different. The correct answer’s rank remains in the thousands or tens of thousands even at the final layer, indicating that the conditional answer is never confidently resolved internally. More strikingly, the rank often *worsens* between L24 and L28 rather than improving — for instance, from 2,135 to 67,084 at $|\mathcal{L}| = 12$, and from 1,131 to 6,757 at $|\mathcal{L}| = 54$. This suggests that whatever partial signal about the conditional answer exists in intermediate layers is being actively suppressed in the final layers.

These findings offer a mechanistic complement to our behavioral results: the joint-vs-conditional accuracy gap reported in Section 4 is not merely a matter of surface-level output formatting or decoding, but reflects a genuine difference in how far the correct answer is resolved within the model’s internal computation.

F Independence of Generated Samples

In our observations, LLM-generated samples are not fully independent and often follow or-

Task	$ \mathcal{L} $	Rank @ L20	Rank @ L24	Rank @ L28
<i>joint-mode</i>	12	2,627	164	4
	27	53,929	6,087	9
	36	9,133	7,826	13
	54	14,048	3,197	6
<i>cond-mode</i>	12	15,301	2,135	67,084
	27	35,685	7,071	38,581
	36	65,447	3,307	20,458
	54	70,040	1,131	6,757

Table 4: Rank of the correct answer token at three transformer layers of Qwen2.5-7B, obtained via logit projection of each layer’s hidden state through the LM head. Lower rank indicates the correct token is closer to being the top prediction. Results are shown for *joint-mode* and *cond-mode* across four joint space sizes $|\mathcal{L}|$.

dering patterns similar to those in the input prompt. For example, as illustrated below, DeepSeek-R1-Distill-Qwen explicitly acknowledges the need to shuffle the samples. However, due to the lack of access to a code interpreter or true randomness, it prioritizes matching target frequencies over enforcing independence:

"So, the final step is to create a list of 84 triplets, where each triplet is repeated according to its count, and then the list is shuffled to randomize the order. But since I can't perform the shuffling here, I'll have to present the triplet counts and note that the actual output would be a shuffled list"

To investigate this phenomenon more systematically, we conducted additional analyses using two statistical methods: (1) the Durbin–Watson (DW) statistic (Durbin and Watson, 1951) to detect autocorrelation, and (2) transition matrix entropy (Shannon, 1948) to evaluate diversity in transition patterns. Samples are classified as non-independent based on the aggregated results of these two measures. Table 5 reports the corresponding statistics for samples generated by each model.

G Evaluation on Real-world Data

To validate our findings on real-world data, we conducted experiments on the Mushroom dataset (mus, 1981), focusing on three categorical features and a subset of 80 samples. The resulting joint distribution is highly skewed, with a large proportion of zero-frequency outcomes, making the task particularly challenging. Table 6 summarizes the observed outcomes and their frequencies in this subset. The experimental results, detailed in Table 7, show that larger models, as well as distilled variants, consistently outperform smaller models across all task configurations.

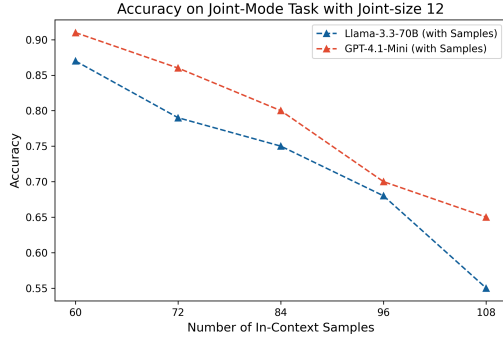


Figure 4: Accuracy on *joint-mode* task with fixed joint space size $|\mathcal{L}| = 12$ as the number of observed in-context samples increases.

H Effect of Sample Count in Context

To investigate the impact of input length independently from task complexity, we fixed the joint space size at $|\mathcal{L}| = 12$ and varied the number of observed samples K provided in the prompt. As shown in Figure 4, increasing the number of in-context samples leads to a clear drop in accuracy for both models. This degradation suggests that the challenge stems not just from the probabilistic reasoning task itself, but also from the models’ limited capacity to process and reason over longer contexts. This highlights a core limitation in current LLMs and points to context scaling as a key direction for future improvement.

I Access to Code Interpreter

In the frequency-based setting, both Llama3.3-70B and GPT-4.1-mini achieved near-perfect accuracy, but their performance degraded sharply when asked to infer frequencies directly from raw samples due to limited counting abilities. To explore mitigation strategies, we conducted an additional set of experiments with GPT-4.1-mini augmented with access to a code interpreter. In this setup, prompts encouraged the model to write Python code to count outcome frequencies before solving the *joint-mode* task. This hybrid approach led to a substantial performance gain. As shown in Table 8, accuracy levels were restored to nearly those observed in the frequency-provided condition, demonstrating that the main limitation lies in the models’ ability to perform reliable counting over long contexts rather than in reasoning about the task itself.

Table 5: Independence analysis of generated samples by each model. Reported values include the Durbin–Watson (DW) statistic and transition matrix entropy. Samples are classified as non-independent if $DW < 1.5$ or > 2.5 , or if the entropy ratio < 0.8 .

Model	(DW) statistic	entropy-ratio	Is independent?
Llama-3.1-8B-Instruct	0.118	0.245	No
Qwen2.5-7B-Instruct-1M	0.002	0.209	No
DeepSeek-R1-Distill-Qwen-7B	0.002	0.195	No
Llama3.3-70B	0.153	0.323	No
GPT-4o-mini	0.636	0.608	No
GPT-4.1-mini	0.149	0.336	No

Table 6: Frequencies of joint distribution outcomes for the selected subset of the Mushroom dataset. Each column corresponds to one outcome of the joint distribution, along with its frequency in 80 samples. The distribution is highly skewed, with several outcomes having zero frequency.

Mushroom Feature					Labels											
gill-attachment	f	f	f	f	f	f	f	f	a	a	a	a	a	a	a	a
gill-size	b	b	b	b	n	n	n	n	b	b	b	b	n	n	n	n
gill-color	k	g	p	o	k	g	p	o	k	g	p	o	k	g	p	o
Frequency	13	17	34	0	2	0	9	0	0	0	0	0	0	0	0	0

Table 7: Performance of different models across all six task configurations on the Mushroom dataset. Mode identification tasks are evaluated using accuracy, while MLE and sampling tasks are evaluated using total variation distance (TVD).

Model	<i>joint-mode</i>	<i>cond-mode</i>	<i>joint-MLE</i>	<i>cond-MLE</i>	<i>joint-samp</i>	<i>cond-samp</i>
Llama-3.1-8B-Instruct	80	78	0.011	0.111	0.465	0.414
Qwen2.5-7B-Instruct-1M	95	75	0.003	0.131	0.544	0.441
DeepSeek-R1-Distill-Qwen-7B	96	96	0.002	0.100	0.237	0.173
Llama3.3-70B	100	100	0.0	0.005	0.111	0.089
GPT-4o-mini	100	91	0.0	0.028	0.287	0.352
GPT-4.1-mini	100	100	0.0	0.0	0.193	0.218

Table 8: Accuracy of GPT-4.1-mini on the *joint-mode* task under three configurations across different joint sizes. Results show that accuracy in the samples-in-context setting with access to code interpreter closely matches the frequencies-provided case.

Setting	$ \mathcal{L} = 12$	$ \mathcal{L} = 27$	$ \mathcal{L} = 36$	$ \mathcal{L} = 54$
Frequencies in context	1.0	1.0	0.99	0.96
Raw samples in context	0.87	0.71	0.55	0.28
Raw samples + code interpreter	1.0	1.0	0.99	0.97