Can Reinforcement Learning and Chain of Thought improve random numbers generated by LLM.

Case study of coin flips

Witold Czubala[1]

witod.czubala@gmail.com

Introduction

This study investigates the behavior of Large Language Models (LLMs) in generating statistically random sequences, specifically examining the influence of Reinforcement Learning (RL) and Chain of Thought (CoT) prompting. We aim to quantify improvements in the quality of random sequences generated by LLMs when these techniques are applied.

The ability of LLMs to generate high-quality random numbers has significant implications across various fields. An example could be in finance where the user can ask LLM to analyze their investment portfolio for planning a retirement. This requires creating scenarios that are usually based on Monte Carlo simulations of different possible outcomes. This necessitates an evaluation of LLMs' intrinsic capabilities for random number generation.

The coin flip generation task serves as a focused test case, allowing for straightforward verification of statistical properties and biases in the generated sequences. Prior research (e.g., Koevering et al., 2024) has identified existing biases in LLM-generated coin flips, such as a preference for Heads over Tails, recency effects, and distributional anomalies.

Further studies have explored methods like in-context learning to mitigate such biases (Gupta et al., 2025). This work specifically investigates the potential of RL-tuned models and CoT prompting to enhance the randomness of LLM-generated coin flip sequences.

Using an agent-based framework, we compare coin flip sequences generated by DeepSeek V3 (DSV3) with those from Deepseek R1 (DSR1)[2] , an open-source model fine-tuned with RL on a DSV3 base (Guo et al., 2025). Furthermore, by analyzing both the initial and final outputs from DSR1's CoT process, we assess the impact of its internal reasoning on the randomness of the generated sequences.

Experiment

The primary advantage of the coin flip task is its interpretability and the ease of comparison against a well-defined theoretical distribution (the binomial distribution). Our experimental setup involved prompting both DSV3 and DSR1 to generate a sequence of 10 random coin flips, with this process repeated 100 times for each model and configuration. The resulting datasets were then analyzed to compare the performance

[1] Data Scientist at UBS and Lecturer at Columbia University – all views are mine

[2] DSR1 is open source and trained with RL using DSV3 as a base.

between DSV3 and DSR1, and also to evaluate the difference between the initial and CoT-refined final outputs from DSR1.

Methodology Details

The CoT process employed by DSR1 inherently produces verbose outputs, often including intermediate reasoning steps and self-corrections. It can involve a number of results that are being self-corrected. The model is only somewhat responsive to the instructions of brevity. Examples of the prompts used are provided in the Appendix. We extract the first and the last series of coin tosses if several are available from the model answer to assess pre-CoT and post-CoT states, respectively. Sequence extraction was managed using an Autogen-based agentic framework: one agent handled LLM prompting and response validation, while another parsed and extracted coin flip sequences. Further details on this setup are in the Appendix..

For comparison DSV3 answers are concise and usually include one result per question.

We set the models at a temperature of 1.0 and used no seed to promote variability in the generated sequences. We also tested results at temperature 1.5, the maximum recommended by DeepSeek for creative generation.

Deep Seek V3

Analysis of DSV3 outputs reveals significant deviations from the theoretical binomial distribution for the number of heads, as well as a high frequency of the same sequence of tosses repetition. We use Total Variation Distance (TVD) to compare distributions. The results are presented in Figure 1 and Figure 2. DSV3 exhibits a strong bias towards generating an equal number of heads and tails (5H/5T), underrepresenting sequences with more unbalanced outcomes compared to the true binomial distribution (Figure 1).

The second issue is a significant number of repeated sequences presented in Table 2 on page 9 and Table 3 on page 10 in the Appendix. To have an idea how unlikely these results are we can think of it as throwing a dice with 1,024 sides (2^10 as we have 10 flips) 100 times, where each of the sides represents a sequence e.g. 'HTTHHTTHTH' It is hard to find analytical solutions to the problem of predicting the exact distribution of these repetitions given the vast number of possible combinations. For this purpose, we decided to run a Monte Carlo experiment which included 1,000,000 runs generating random 100 numbers out of 1,024. The results are presented in Figure 7 on page 13 and Figure 8 on page 14.

We observe that DSV3 generated instances where a single sequence was repeated up to 6 times, often in conjunction with other repeated sequences. Figure 7 (our Monte Carlo simulation) shows that 6 repetitions of any single sequence did not occur in 1,000,000 trials, while 5 repetitions of a single sequence occurred only 72 times. However, from Table 2 and Table 3 (temperatures 1 and 1.5) we can see multiple distinct sequences were often repeated within the same set of 100 DSV3 generations. In Figure 8 page 14 we show that such simultaneous high-frequency repeat patterns are extremely rare in our Monte Carlo simulations. For instance, the most complex repeat patterns observed with notable frequency in 1,000,000 simulations typically involved combinations like several distinct sequences each appearing twice (pairs), or perhaps one sequence appearing three times (a triplet), but not the high co-occurrence of multiple high-frequency repeats seen with DSV3 (compare DSV3's Table 2 on page 9 and Table 3 on page 10 in the Appendix with Monte Carlo Figure 8 on page 14). The conclusion from these results is that coin tosses generated by DSV3 exhibit patterns statistically inconsistent with true random generation. One possible explanation can be that the sequencies of 10 flips are pulled from the tokens in the model training data resulting in very unrealistic repeats of coin tosses.

Since we experiment with different temperature levels we can also see that repeats are less common at 1.5 versus 1.0 (Table 2 on page 9 and Table 3 on page 10 in the Appendix) the occurrence of such high repeat patterns in both DSV3 configurations still corresponds to infinitesimally small probabilities, making them statistically inexplicable by random chance alone.

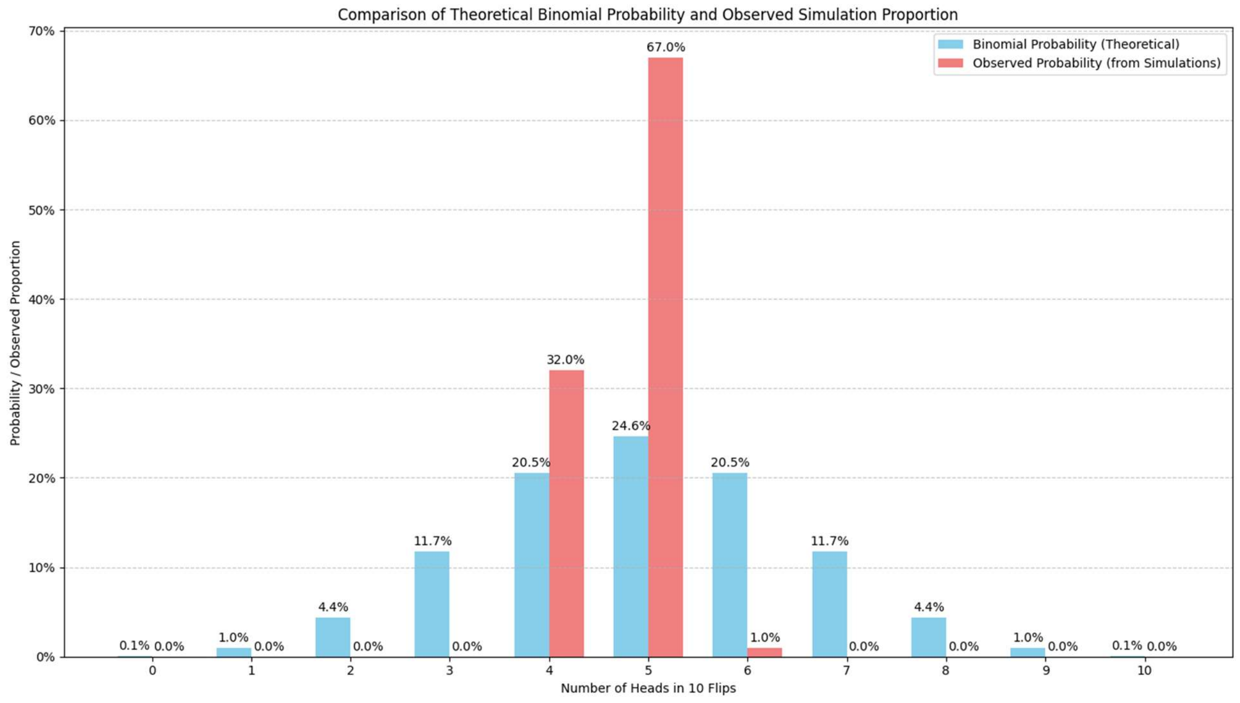

Figure 1 Heads distribution for DSV3 with temperature 1.0

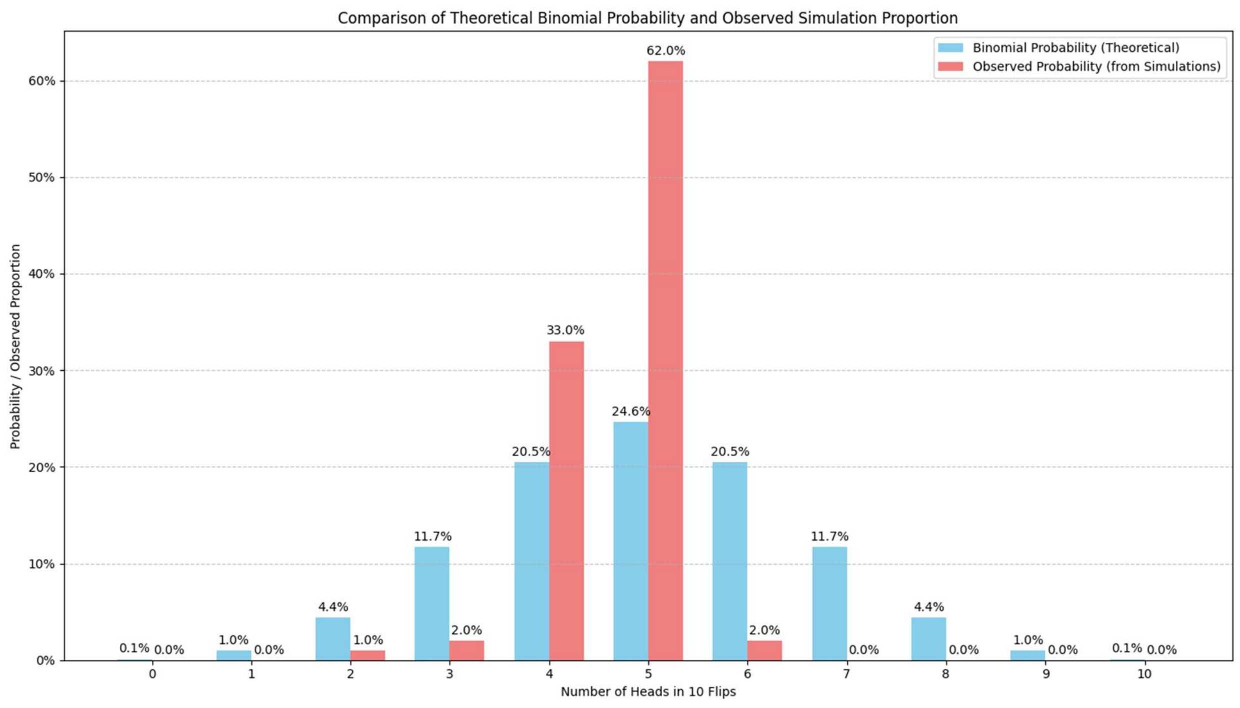

Figure 2 Heads distribution for DSV3 with temperature 1.5

RL Impact (DeepSeek R1)

Next, we conducted the same experiment (10 tosses repeated 100 times) using DSR1. Similar to DSV3 we also see bias towards 5 heads and tails but with a significant improvement. Comparisons of head count distributions can be made, for example, between DSV3 at temp 1.0 (Figure 2) and DSR1's final output at temp 1.0 (Figure 4) and similarly between DSV3 at temp 1.5 (Figure 1) and DSR1's final output at temp 1.5 (Figure 4). In Table 1 on page 7 we compute Total Variation Distance (TVD) between observed probabilities of the number of heads versus the theoretical binomial distribution. We see an improvement between DSR1 (final toss) and DSV3 for the same temperature levels towards the binomial distribution with lower values reported for DSR1 (total variation range is between 0 and 1 where 0 represents indistinguishable distributions).

Similar to DSV3, DSR1 also exhibits a higher-than-expected repeatability of exact toss sequences, though generally to a lesser extent. We show repeated sequences in Table 5 on page 11 and Table 7 on page 12 (final toss for temperatures 1 and 1.5). Compared to DSV3 the number of exactly repeated sequences is significantly smaller with max value of 5 and 3 respectively for temperature 1.0 and 1.5. When considering the joint occurrence of multiple repeated sequences, DSR1 at temperature 1.5 (Table 7 on page 12 , e.g., showing 1 triplet and up to 16 pairs if summed differently or specific complex patterns) exhibits repeat patterns that, while still rare, are closer to those occasionally seen in true random simulations (e.g., Figure 8 shows a pattern of '1 triplet and 7 pairs' occurred with ~0.8% probability in Monte Carlo trials). The specific configurations observed for DSR1 are still highly improbable but less extreme than DSV3.

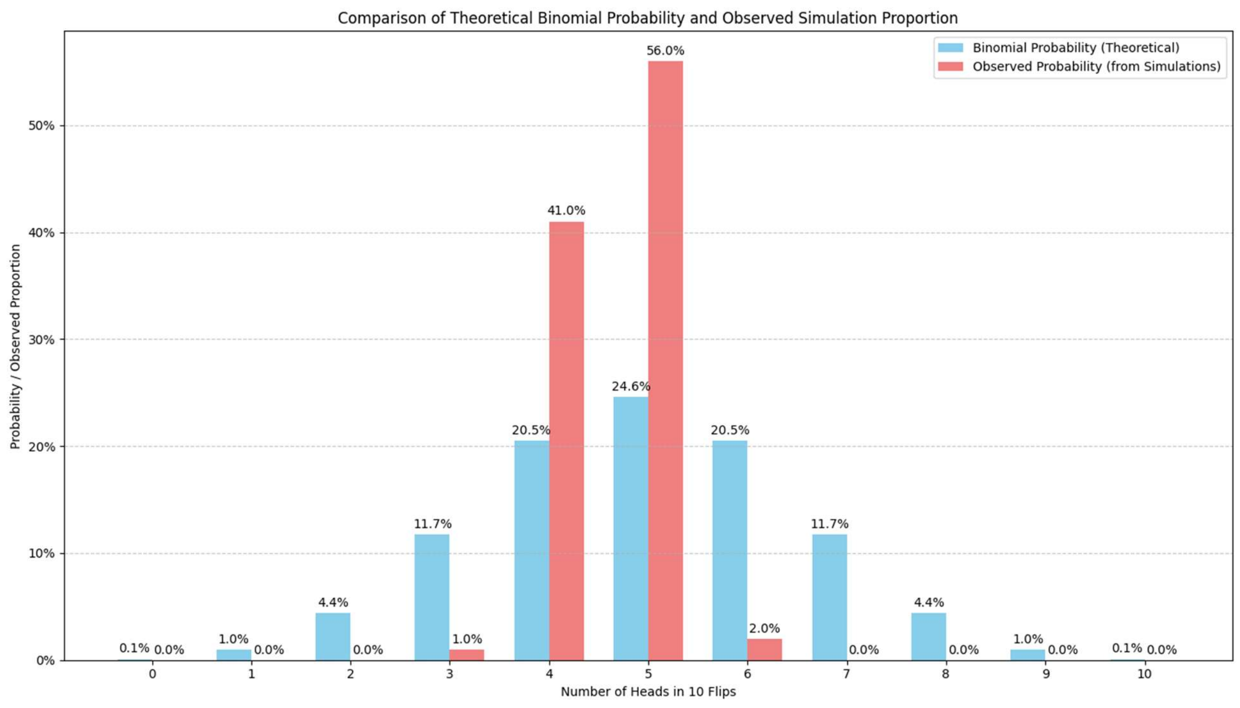

Figure 3 Heads distribution for DSR1 final generated toss with temperature 1.0

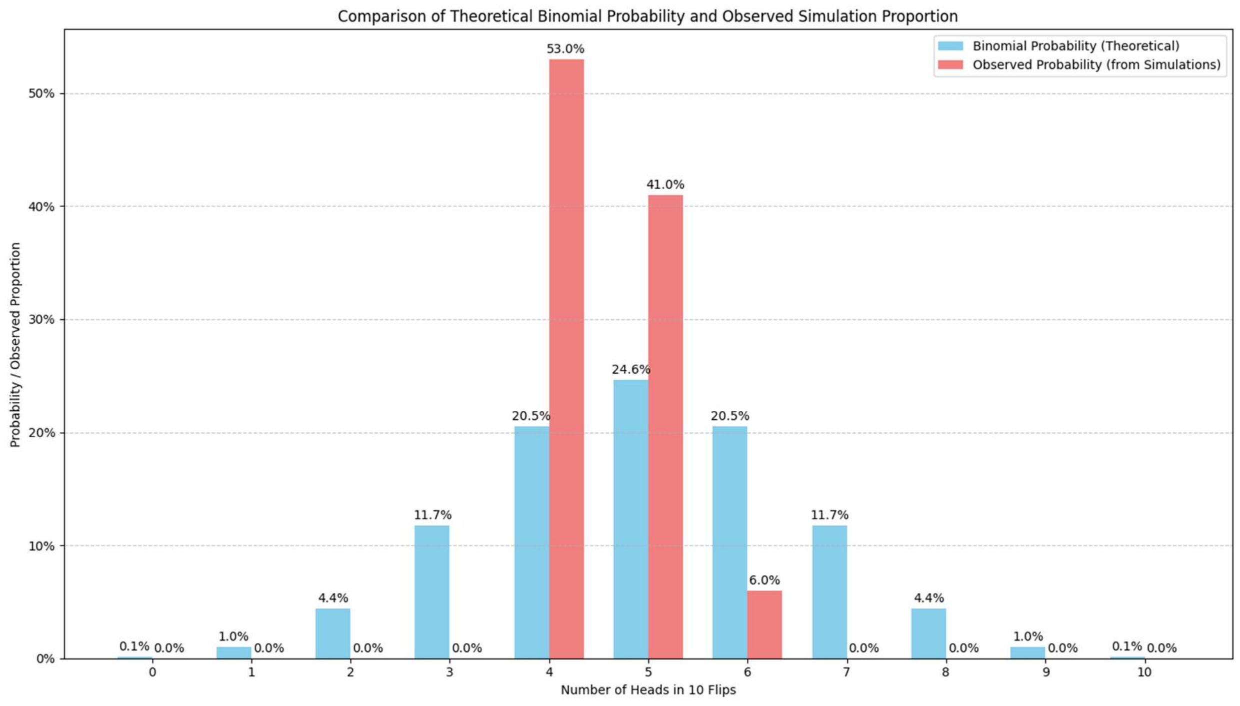

Figure 4 Heads distribution for DSR1 final generated toss with temperature 1.5

Chain of Thought

To evaluate the impact of CoT within DSR1, our agentic framework was designed to extract both the *first* sequence generated (potentially pre-full CoT reasoning) and the *final* sequence (post-CoT reasoning).

By inspecting the model output, we see that it does generate a number of coin toss sequences and updates them frequently citing a need to create truly random and unbiased sample (see a typical example on page 14 in the Appendix). We use Autogen where one agent is prompting the LLM and the other is extracting first and final coin tosses. For details see Appendix on page 9.

First, we compare how empirical distribution of heads in 10 tosses (repeated 100 times) compares to binomial distribution for the first toss versus final toss in DSR1 response for temperature 1.0 and 1.5. For this we compare Figure 5 on page 6 to Figure 3 on page 5 (temperature 1.0) and Figure 6 on page 7 to Figure 4 on page 5 (temperature 1.5). In both temperature settings, there is a visible decrease in the overrepresentation of 5H/5T outcomes in the final (CoT-refined) answer compared to the initial generated sequence. In Table 1 page 7 we can see change in total variation between first and final answer. Notably, the final DSR1 output at temperature 1.5 achieves the lowest TVD against the binomial distribution (48.88%) among all experimental conditions reported in Table 1.

As previously noted, the models generate a surprisingly high frequency of exact sequence repetitions. This phenomenon is particularly striking in the initial sequences generated during DSR1's CoT process. Again for different temperatures we can look at Table 4 on page 11 and Table 6 on page 12. The number of max repeated sequence is 20 for temperature 1.0 and 18 for temperature 1.5. As a reminder Figure 7 on page 13 shows the maximum repeated frequency of any single sequence was 5 (and occurred only 72 times) in 1,000,000 Monte Carlo simulations. In summary, DSR1's initial outputs not only show a strong bias towards 5H/5T outcomes (e.g., ~76% for T1.0 and ~66% for T1.5, compared to the theoretical 24.6%) but these often consist of the *exact same* H/T sequence being repeated.

However, if we look at the final answer after going through CoT the numbers of repeats drops drastically – see Table 5 on page 11 and Table 7 on page 12 for both temperature levels.

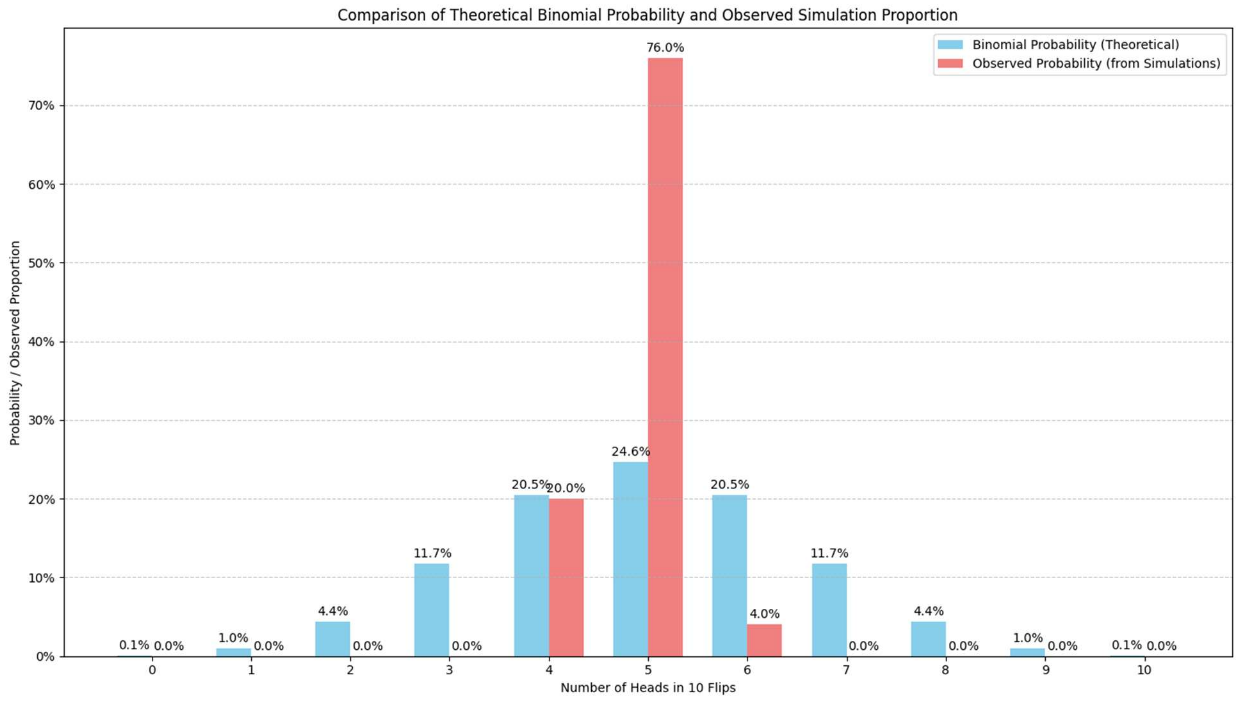

Figure 5 Heads distribution for DSR1 first generated toss with temperature 1.0

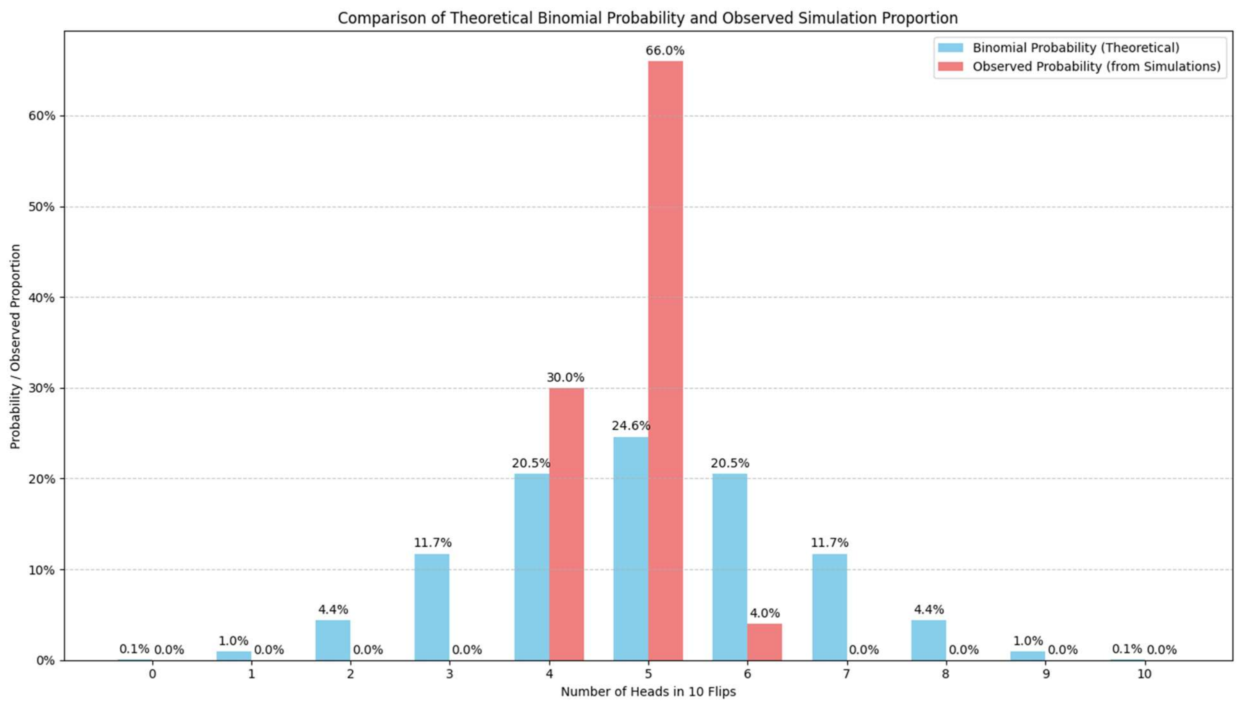

Figure 6 Heads distribution for DSR1 first generated toss with temperature 1.5

Model Configuration	Total Variation
DSV3 Temperature 1.0	53.88%
DSR1 Temperature 1.0 First Toss	51.39%
DSR1 Temperature 1.0 Final Toss	51.88%
DSV3 Temperature 1.5	49.88%
DSR1 Temperature 1.5 First Toss	50.88%
DSR1 Temperature 1.5 Final Toss	48.88%

Table 1 Total variation between binomial and observed distributions

Conclusion

The experiment and the data above help understand impact of Reinforcement Learning and Chain of Thought on the LLM's random number generating process. We show significant improvement in generated results that are closer to the binomial distribution and have fewer repeated sequencies when we use DSR1 versus DSV3. We also show improvements achieved using Chain of Thought. However, even with the results that are statistically closest to random (final results from DSR1 with temperature 1.5) we still observe significant deviation from the binomial distribution and a statistically unrealistic number of repeated sequencies.

This suggests that reinforcement learning combined with chain of thought can improve the quality of random numbers generated by LLMs but achieving true statistical randomness requires additional effort or may face inherent LLM limitations.

As a next step we plan to specifically train an LLM using reinforcement learning for random number generation (based on GPRO algorithm see Shao et al 2024) and conduct the above experiments.

References

Gupta, R., Corona, R., Ge, J., Wang, E., Klein, D., Darrell, T., & Chan, D. M. (2025). Enough Coin Flips Can Make LLMs Act Bayesian. *arXiv preprint arXiv:2503.04722*.

Van Koevering, K., & Kleinberg, J. (2024). How Random is Random? Evaluating the Randomness and Humaness of LLMs' Coin Flips. *arXiv preprint arXiv:2406.00092*.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., ... & He, Y. (2025). Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., ... & Guo, D. (2024). Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

DeepSeek R1 set up https://huggingface.co/deepseek-ai/DeepSeek-R1

APPENDIX

Autogen

We set up several agents with Autogen. The first is connecting to the LLM and prompting it. It checks for results of 10 coin tosses within generated series. In case the result was not delivered (this can happen mostly due to lengthy CoT in DSR1 that exceeds the token limit we set up) it prompts again.

The second agent looks for a line that starts with 10-coin toss results (based on a prompt which instructs the LLM to send the final result in a separate line). This is to avoid taking intermediary results where the model is thinking to deliver an answer. In a second variant of the exercise we take the first result of coin tosses and the last to extract and compare CoT impact.

Repeated Tosses

Flip Sequence	Frequency
HTTHHTTHTH	6
HTTHTTHHTT	6
HTTHTHHTTH	5
HTTHHTTHHT	5
HTTHTTTHHT	3
THHTHTTTHT	3
HTTHTTHHTH	3
HTHTTHHTTH	3
HTHHTTHTHT	3
HTTHHTTTHT	3
HTTHHTHTTH	2
HTTHTHHTHT	2
HTTTHTHHTT	2
HTTHTHTHTH	2
HHTHTTHHTT	2
HTTHHTHTHT	2
THTTHHTTHT	2
HTTHHTTTHH	2
HHTHTTTHHT	2
HTTTHTHHHT	2
HTTHHTTHTT	2
HTHTHTTTHH	2
Unique Sequences	36

Table 2 Repeated Sequencies DSV3 temperature 1.0

Flip Sequence	Frequency
HTHTTHHTTH	6
THHTHTTTHT	5
HHTTHHTTHT	3
HTTHHTTHTH	3
HHTHTTTTHT	2
HTTHTHHHTT	2
HTHTTTHTHH	2
HTTTHHTTHH	2
THHTHTTHHT	2
HHTHTTTHTH	2
HTTHHTTTHH	2
TTTHHTHHTT	2
HHTTTHTHHT	2
HTHTHHTTHT	2
HTTHTTTHTH	2
TTTHHTTTHT	2
HTTHTHHTTT	2
HTTHTTHTHT	2
TTHTHHTTHT	2
HTHTTHTHHT	2
HHTHTTTHHT	2
HTTHTHHTTH	2
Unique Sequences	47

Table 3 Repeated Sequencies DSV3 temperature 1.5

Flip Sequence	Frequency
THTHTHTHHT	20
HTTHTHTHTH	4
TTHHTTHTTH	4
THTHHTTHTH	3
THTTHHTHTH	3
HTHHTTTHTH	2
HTHTTTHHHT	2
HTHTTHHTTH	2
HTTHTHHTTH	2
HHTTHTTHHT	2
TTHTHHTTHT	2
THHTTHTHHT	2
HTTHTHHTHT	2
HHTTTHTTHT	2
THHTHTTHTH	2
TTHHTHTTHH	2
THTTHHTTHH	2
TTHHTTHTHT	2
Unique Sequences	40

Table 4 Repeated Sequencies DSR1 First Toss with temperature 1.0

Flip Sequence	Frequency
TTHHTTHTTH	5
HHTTTHTHHT	4
TTHHTHTTHT	4
HHTTTHTHTH	4
THHTHTTHHT	3
HHTTTHTTHT	3
TTHHTTHTHH	3
HTHTTHTTHT	2
TTHTHHHTHT	2
HTHTTTHHHT	2
THTTHHHTTH	2
THTHTHTHHT	2
HTHTTHTHHT	2
THTTHHTTHT	2
TTHTHHTHTT	2
THHTHTTHTH	2
HTHTTTHHTT	2
HTHTTHHTTH	2
HTTHTHHTTH	2
Unique Sequences	50

Table 5 Repeated Sequencies DSR1 Final Toss with temperature 1.0

Flip Sequence	Frequency
THTHTHTHHT	18
HTHTTHHTHT	4
TTHHTTHHTH	3
THTHHTTHTH	2
THTTTHHTHH	2
THHTHTTTHT	2
THTTHHHTHT	2
HHTTHTTHHT	2
THTHTTHHTH	2
HHTTHTTTHT	2
TTHHTHTHHT	2
TTHHTTTHHT	2
Unique Sequences	57

Table 6 Repeated Sequencies DSR1 First Toss with temperature 1.5

Flip Sequence	Frequency
THHTTTHTHH	3
TTTHHHTTHT	2
THTTHHTTHT	2
HTTHTTHTHT	2
TTTHHHTTTH	2
TTHHTTHHTH	2
HTHHTTTHHH	2
HTTHTTTHHT	2
HTTTHTHHTT	2
TTHTTHHTTH	2
HHTTHTTTHT	2
HHTHTTTHHT	2
HTHTTTHTTH	2
HTHTTHTTTH	2
HTTHHTTHHT	2
HTHTTTHTHT	2
HHTHTTTHTH	2
Unique Sequences	65

Table 7 Repeated Sequencies DSR1 Final Toss with temperature 1.5

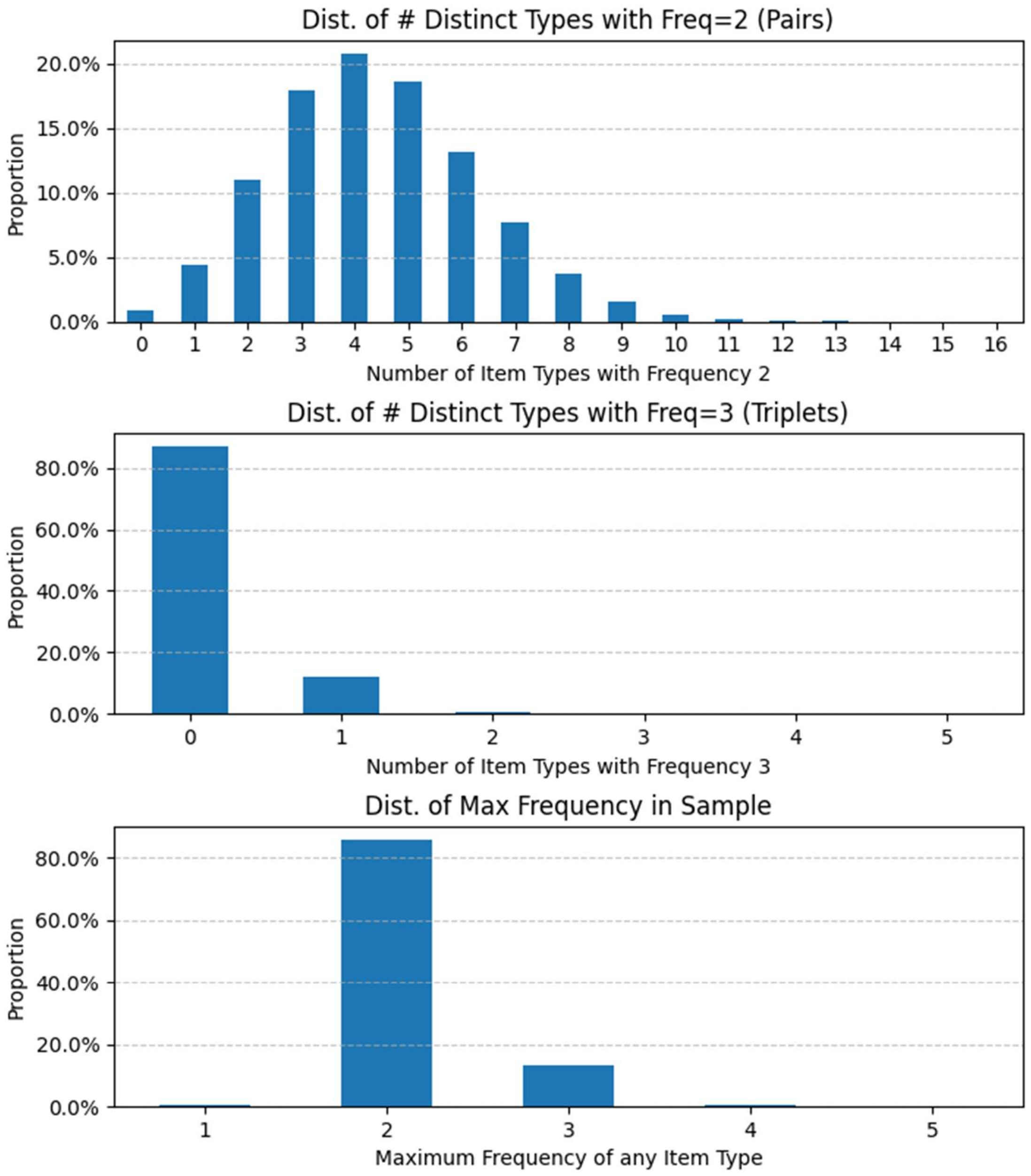

Figure 7 Distribution of the maximum frequency achieved by any single sequence when drawing 100 samples from 1024 possibilities. Based on 1,000,000 Monte Carlo simulations

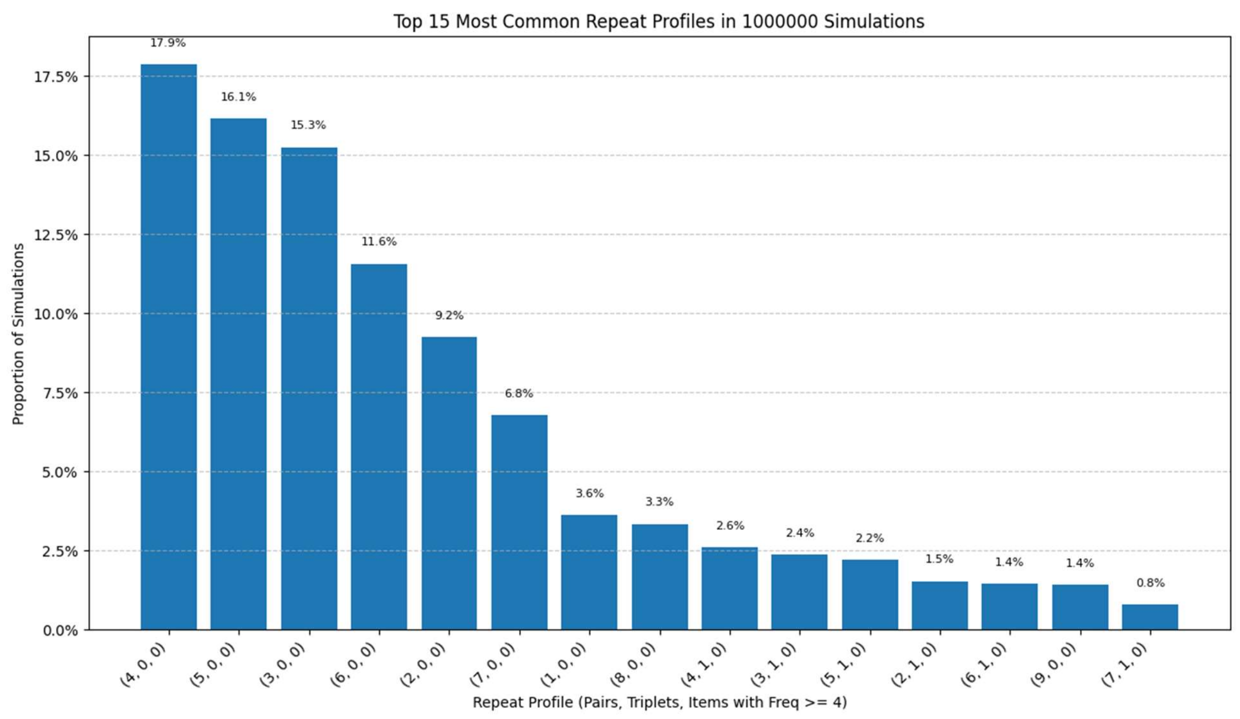

Figure 8 Top 15 most common repeat profiles observed when drawing 100 samples from 1024 possibilities. Profile shows number of distinct sequences appearing twice, thrice, four-or-more times based on 1,000,000 Monte Carlo simulations

Typical DSR1 generated output

"Okay, let me tackle this coin flip simulation. The user wants a 10-character sequence of H and T, no other characters, spaces, or explanations outside the specified format. They mentioned using a 50% chance for each flip, unbiased.

First, I need to generate 10 random flips. The key here is ensuring each flip is independent and each has equal probability. How can I simulate each flip accurately? Well, I can imagine using a random number generator that gives a 50/50 chance for each outcome.

Wait, the example shows the sequence like THTHTHTHHT. So example alternated a bit, but explained each flip is independent. So for each of the 10 positions, generate H or T with 50% chance without any bias or pattern. I need to make sure not to have any streaks or particular patterns but just pure randomness. Of course, in true randomness, there might be streaks, but that's part of the process

So maybe go step by step for each flip. First flip: maybe generate a random number 0 or 1, 0 is T, 1 is H. Repeat ten times. Let's jot down ten random binary choices. For example:

1. H

2. T

3. H

4. T

5. T

6. H

7. H

8. T

9. H

10. T

But arrange those into a string. Let's see... Maybe that gives HTHTTHHTrt— Wait, that last T should be properly listed. So concatenate all. Checking each step is H or T independently, generating a 10-letter string.

After creating the sequence, I need to explain in one sentence that each flip was a 50% chance, done independently. Then add TERMINATE. Strict formatting with no extra text.

Make sure to not include any other characters or mistakes, no typos. No spaces in the sequence, all caps, exactly ten characters. Check to count after writing. Once validated, output the response exactly as shown.

</think>

HHTHTTTHTT

---Explanation---

Used a fair coin model generating each flip with 1/2 probability for H or T independently."

Prompt Example

"You are a direct output coin flip simulator. Your ONLY job is to output a 10-character random and unbiased coin flip sequence ('H'/'T'), "

"a one-sentence explanation, and the word TERMINATE. NO OTHER TEXT, NO THINKING, NO PREAMBLE. "

"Your response MUST be ONLY the formatted output.\n\n"

"STRICT FORMAT (NO DEVIATIONS):\n"

"[10_CHAR_SEQUENCE_HT_ONLY_NO_SPACES_OR_OTHER_CHARS]\n"

"---Explanation---\n"

"[ONE_SENTENCE_EXPLANATION_OF_RANDOMIZATION_METHOD]\n"

"TERMINATE\n\n"

"EXAMPLE OF YOUR EXACT AND ONLY OUTPUT:\n"

"THTHTHTHHT\n"

"---Explanation---\n"

"Each flip was simulated with a 50% chance for H or T, independently.\n"

"TERMINATE\n\n"

"Produce the final answer in the specified format."