Do Code Models Suffer from the Dunning-Kruger Effect?

Anonymous ACL submission

Abstract

The Dunning-Kruger effect (DKE) is the cognitive bias in which participants with limited competence in a domain tend to over estimate their perceived performance or expertise in that domain. AI models today can outperform humans in many cognitive tasks and humans rely on these models for decision making. It becomes important to understand the inherent biases carried by these models to make sure they are used in a controlled and responsible manner. In this paper, we examine the Dunning-Kruger effect on AI models for programming tasks. By comparing the model on question-answering over increasingly rare programming languages, it is found that the models show a similar competence-capability curve as humans. Upon closer examination of the confidence curves we also find that the strength of the bias is proportionate to the competence of the models, for example, a less competent model has a stronger bias. This aligns with human experiments for the bias. We open source all benchmarks and predictions to encourage research in biases for AI models.

1 Introduction

Large language models have shown great performance on code tasks, outperforming humans in code generation, repair, refactor and question-answering (Huynh and Lin, 2025; Cordeiro et al., 2024; Jelodar et al., 2025). With advancements in model development with reasoning models (Cai et al., 2024) and diffusion models (Chen et al., 2024b), AI models are only going to get better with the models potentially surpassing human performance. With these developments, AI models are being used more frequently to automate a broad range of programming related tasks including code authoring, reviewing, and debugging (Odeh et al., 2024; Anand et al., 2024; Chen et al., 2021).

As AI systems are increasingly being integrated into critical processes both within and outside the programming domain, it becomes essential to recognize and address the biases these models may inherit (Vakali and Tantalaki, 2024) from human-generated training data. Numerous studies and policy discussions have highlighted the risks associated with biased AI outputs, particularly in domains such as healthcare, and employment (Abrams, 2024; Vicente and Matute, 2023; Landers and Behrend, 2023). In this work, we focus on a specific cognitive bias: the Dunning-Kruger Effect (DKE)—a phenomenon in which individuals with lower competence in a domain tend to overestimate their abilities (Mazor and Fleming, 2021; Magnus and Peresetsky, 2022). While DKE has been extensively studied in human psychology, its presence in AI systems remains underexplored. We argue that studying DKE in AI models is valuable for two reasons. First, it offers a lens through which to examine model mis-calibration, particularly in low-competence regimes. Second, it allows us to test whether models exhibit human-like patterns of overconfidence, which could have implications for trust, interpretability, and downstream decision-making.

Our results reveal that the models' perceived performance shows statistically significant inflation compared to actual performance, similar to the effect previously studied in humans. The models' overestimation of their performance becomes more pronounced with lower actual performance of the model and with increasing hardness of the tasks (measured by rarity of the programming domain), aligning strongly with the patterns observed in human cognition. These findings underscore the importance of understanding cognitive biases in AI systems and lay the groundwork for deeper interdisciplinary research at the intersection of cognitive science and machine learning.

In this paper, we make the following contributions:

1. We provide statistically significant evidence of the Dunning-Kruger effect in AI models for programming tasks.

2. We analyze how the strength of this bias varies with (a) the model's base performance and (b) the rarity of the programming domain.

2 Related Work

Cognitive biases in AI models. Studies have shown that LLMs can reflect human-like biases, including overconfidence and self-enhancement, despite lacking self-awareness (Gu et al., 2024; Salecha et al., 2024; Sun et al., 2025; Ye et al., 2024). These biases often stem from training data patterns or architectural choices (Geng et al., 2023; Tjuatja et al., 2024). Among these, overconfidence is particularly concerning, as it can lead to misleading outputs that appear authoritative but are incorrect—an issue that parallels the DKE (Dunning et al., 2003; Kruger and Dunning, 1999; Ehrlinger et al., 2008) observed in human cognition.

Generalization and confidence estimation In the context of code models, prior work has highlighted challenges in generalizing to rare programming languages (Chen et al., 2024a; Cassano et al., 2024; Giagnorio et al., 2025; Mora et al., 2024). While model accuracy drops on out-of-distribution tasks, confidence scores often remain high (Chen et al., 2021), revealing a disconnect between competence and self-assessment. Traditional confidence estimation methods, based on logits or self-reported probabilities, are frequently miscalibrated in unfamiliar domains (Shorinwa et al., 2024; Shen et al., 2024; Yang et al., 2024; Li et al., 2024b). Recent work introduces relative confidence estimation as a more robust alternative (Shrivastava et al., 2025). These methods help uncover behavioral patterns like overconfidence, and our work builds on these techniques to investigate whether code models exhibit the DKE.

3 Methods

For our study, we use *multiple-choice questions* (q, A, a) as tasks where q is the programming-related question, A is the set of answer choices, and a is the expected answer. Each question q also belongs to a domain $q \in D$ that is the broad topic which this question pertains to. In our setting, the tasks are specific questions about programming and domains are the individual programming languages. For example, the question "Variables of which data types are preceded by a dollar sign in Perl?" will have the domain "Perl".

For each such task and model M, we prompt the model M to answer the question q given the choices A—we say the model M is correct on the task if the answer a_M produced by M matches a. We define the *actual performance* $\mathsf{AP}(M, D)$ of the model M on a domain D to the fraction of domain D tasks it is correct on.

3.1 Measuring Perceived Performance

We use two different techniques to measure perceived performance of AI models, absolute confidence and relative confidence. For absolute confidence, the model is asked to produce a confidence score in the range $[0, 1]$ along with its answer. Model M's absolute confidence $\mathsf{PP}_{\mathsf{Abs}}(M, D)$ on a domain D is the mean of its absolute confidence scores on individual tasks that belong to D.

Previously, relative confidence estimation methods have been shown to produce more reliable confidence scores than absolute confidence estimation (Shrivastava et al., 2025). For every pair of questions q_i and q_j, we prompt the model to indicate which it is more confident in answering. These pairwise preferences are aggregated into scalar confidence scores using two different rank aggregation algorithms, ELO (Elo and Sloan, 1978) and TrueSkill (Herbrich et al., 2006). These algorithms treat each question as a "player" with q_i "winning" against q_j if the model is more confident in answering q_i over q_j. They produce a scalar strength value for each q_i with higher strengths indicating the model's higher confidence (see Appendix A). We normalize the ELO and TrueSkill scores to the range $[0, 1]$ linearly, and set the relative confidences $\mathsf{PP}_{\mathsf{ELO}}(M, D)$ and $\mathsf{PP}_{\mathsf{TrueSkill}}(M, D)$ to be the mean strengths of the questions in D.

3.2 Measuring the Dunning-Kruger Effect

There have been several closely related effects that have all been referred to under the umbrella term of DKE (Kruger and Dunning, 1999). Here, we consider two specific variants from the literature—the intra-participant (Muthukrishna et al., 2018; Moore et al., 2018) and inter-participant versions (Dunning et al., 2003; Hodges et al., 2001; Edwards et al., 2003; Haun et al., 2000). In the intra-participant version, the question is "Does a single participant over-estimate their performance

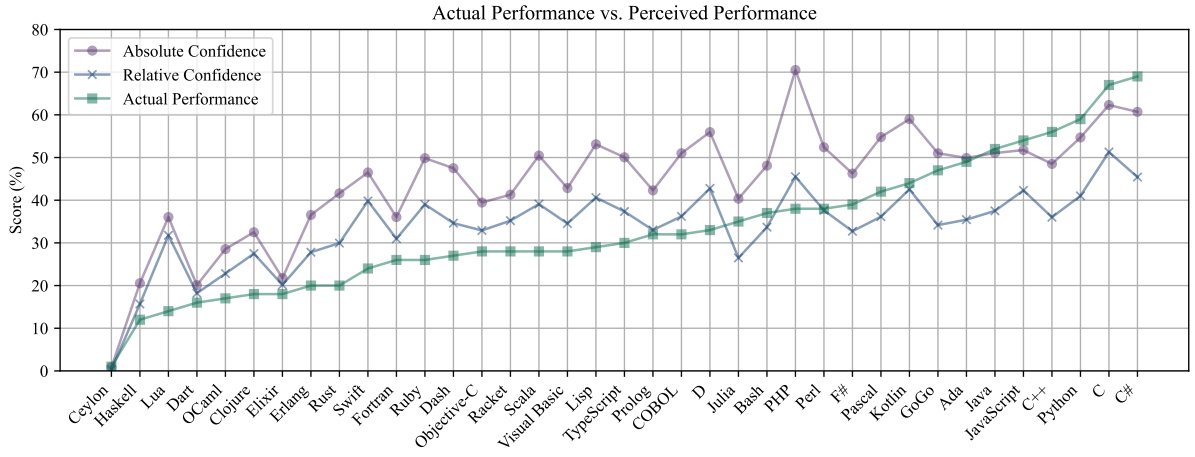


Figure 1: Actual vs. perceived performance for GPT-4o across different languages sorted by actual performance

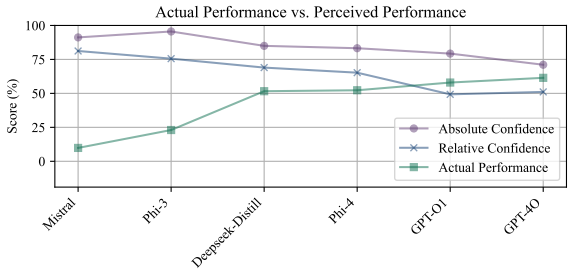


Figure 2: Inter-model DKE

more in domains where they have low actual performance?" and for the inter-participant version, it is "Do participants who show low actual performance over-estimate their performance more?"

For the intra-participant version, we fix M and measure the *over-confidence* per domain D:

$$\Delta_{\text{overconf}}(M, D) = \mathsf{PP}(M, D) - \mathsf{AP}(M, D)$$

where PP is one of $\mathsf{PP}_{\mathsf{Abs}}$, $\mathsf{PP}_{\mathsf{ELO}}$, or $\mathsf{PP}_{\mathsf{TrueSkill}}$. For the inter-participant version, we have:

$$\Delta_{\text{overconf}}(M) = \mathbb{E}_D[\mathsf{PP}(M, D)] - \mathbb{E}_D[\mathsf{AP}(M, D)].$$

Higher Δ_{overconf} in regimes with low actual performance is indicative of the corresponding DKE.

4 Results

We evaluate the presence of the Dunning-Kruger Effect (DKE) in six large language models (LLMs) across 37 programming languages using multiple-choice question answering (MCQA) tasks. The multiple-choice QA data is derived from publicly-available data called CodeNet (Puri et al., 2021). More details on the implementation and experimental setup is in Appendix A.4

4.1 Do code models exhibit the DKE?

Inter-model interpretation of DKE In the inter-model analysis, we observe the DKE pattern: lower-performing models consistently overestimate their capabilities, while higher-performing models exhibit more calibrated or even underconfident behavior. As shown by Fig. 2, models such as Mistral and Phi-3 display a gap between perceived and actual performance. In contrast, models like GPT-4O demonstrate more alignment between perceived and actual performance, especially in relative confidence estimates. Interestingly, the relative confidence curve intersects with the actual performance curve, suggesting that higher-performing models may become under-confident—-an effect not captured by absolute confidence alone.

Inter-domain interpretation of DKE The intra-model analysis further supports the presence of DKE. Figure 1 presents model performance across different domains (programming languages), ordered by actual performance. In domains where models perform poorly—typically rare or low-resource languages such as COBOL, Prolog, and Ceylon—we observe higher overconfidence. Conversely, in high-performing domains like Python and JavaScript, models tend to be better calibrated or even underconfident. This domain-level overestimation is consistent across both absolute and relative confidence measures, reinforcing the hypothesis that models are less aware of their limitations in unfamiliar domains.

4.2 Analysis of Perceived Performance

Absolute Confidence vs. Relative Confidence To quantify these trends, we compute the correlation between overestimation (perceived minus

actual performance) and true performance across both models and domains. Table 1 includes the correlation between (a) actual performance across domains vs. overestimation of performance (AC - RC) and (b) actual performance across models vs. overestimation of perceived performance (AC - RC). The results suggest that the overestimation of perceived performance is higher for models and domains that are more high performing. This indicates that AC becomes an unreliable measure of perceived performance, especially as we encounter increasingly better performing models or domains where LLMs achieve higher performance.

Table 1: Correlation Between Overestimation (AC - RC) and True Performance for Domains and Models

Category	*Method*	*Corr ρ / Tau τ*	*p value*
Domains	Spearman	0.775	1.797×10^{-8}
	Pearson	0.640	2.019×10^{-5}
	Kendall	0.592	3.058×10^{-7}
Models	Spearman	0.775	1.797×10^{-8}
	Pearson	0.640	2.019×10^{-5}
	Kendall	0.592	3.058×10^{-7}

Impact of Rarity of Programming Language We also investigate the relationship between domain rarity and overconfidence. Table 2 shows the correlation of perceived performance with (a) GitHub ranking (most used languages on GitHub) (Ranking, 2025a), (b) IEEE popularity ranking (Ranking, 2024), and (c) TIOBE index (Ranking, 2025b). Across all three sources, we observe a consistent trend: models exhibit higher overconfidence in rarer languages. For instance, GitHub ranking shows a correlation of 0.797 with perceived confidence, highlighting that rarity is a predictor of overconfidence.

Table 2: Percieved performance vs. Rarity Ranking

Ranking	*Method*	*Corr ρ / Tau τ*	*p* (10^{-3})
GitHub	*Spearman*	0.797	1.318
	Kendall	0.690	3.935
IEEE	*Spearman*	0.683	5.863
	Kendall	0.529	8.970
TIOBE	*Spearman*	0.741	0.234
	Kendall	0.662	0.354

5 Discussion and Conclusion

Reviewer agents. Reviewer agents are a common and successful design-pattern in multi-agent systems (Li et al., 2024a; Zhou et al., 2025; Gu et al., 2024; Jin et al., 2024). A model is given a prompt containing specific instructions about the output (e.g., "do not use hateful speech", "do not hardcode passwords", etc). The model's output is passed to a "reviewer agent", often a different instance of the same model, to verify that these rules are not violated. Not all reviewer agents directly correspond to the self-evaluation setting as in our study—the model is often provided auxiliary information during the review (e.g., code execution results, elaborations on output expectations, etc). However, our results suggest that a closer examination of reviewer agents is needed, especially with respect to what kinds of auxiliary information helps AI models make better reviewing decisions.

Cognitive Effect or Statistical Effect? There has been significant debate in the psychology and cognitive science community on whether DKE is a "real" effect with an underlying cognitive cause, or if it is "merely" a statistical effect akin to regression to the mean (Magnus and Peresetsky, 2022). Our results suggest that AI models show DKE-like behaviour and we have an interesting choice between at least 3 possibilities: (a) DKE is a cognitive effect and the underlying cognitive mechanism causing this effect is the same across humans and AI models; (b) DKE is a cognitive effect and the underlying mechanisms are different; or (c) DKE is a purely statistical effect. Our results raise questions questions about options (a) and (b). If the DKE is cognitive in origin, either the cause is so fundamental that it applies to vastly different cognitive systems like humans and AI models, or we have the distasteful option of two very similar effects having different underlying causes. Significant further research of the scope and strength of the DKE in AI models and on the cognitive similarities between humans and AI models is required before an answer to these questions can even be attempted.

Conclusion. Our study is an initial foray into studying whether AI models display cognitive biases that have been previously observed in humans. Our results show that AI models, specifically in the context of answering programming related questions, display DKE-like behaviour. This points to a rich set of future research directions related to the strength and scope of the DKE in models, as well as other self-assessment related cognitive biases including the hard-easy effect or IOED (Juslin, 1993; Levin et al., 2000; Chromik et al., 2021).

6 Limitations

Domain and task choices. A major limitation of this paper is that our study is restricted to one specific domain, i.e., programming. Hence, our results are not immediately generalizable to the wide array of domains that AI language models have been applied to. Similarly, we choose simple multiple-choice question answering as the tasks. Our choice allows us to make the estimation of actual performance simple, without having to consider issues of partial correctness or near match answers, different models being tuned for different response styles, etc. However, for conclusive results on the presence of DKE in AI models in general, both the choice of the domain and choice of tasks needs to be expanded considerably.

Measurements. One crucial measurement in our study is that of perceived performance, i.e., the model's confidence in its answers. Limitations related to models' ability to assign confidence scores have been pointed out previously (Shorinwa et al., 2024; Shen et al., 2024; Yang et al., 2024; Li et al., 2024b), with approaches like relative confidence (Shrivastava et al., 2025) being suggested as alternatives. The conclusion here is that measurement of perceived performance is not as straightforward in AI models as it is in humans. This creates an added layer of complexity in studying self-assessment based cognitive biases in AI models, and adds a threat to the validity of our results.

Explanations for the effect. Many explanations have been hypothesized to be the underlying cause of the DKE in humans (Ehrlinger et al., 2008). Our study intentionally does not attempt to compare or contrast the underlying explanation of the DKE in humans and AI models. Many explanations attributed to the DKE in humans are not directly applicable to AI models. Human DKE explanations rely on causes such as overly positive prior beliefs (Ehrlinger et al., 2008), the distribution of over- and under-performers in the human population, or lack of incentive for accurate self-assessments—none of these apply directly to AI models. One potential explanation that may be common to both humans and AI models is the meta-cognitive explanation, which states that assessing the quality of a performance of a skill is a crucial part of acquiring a skill. This explanation can potentially be tested experimentally in AI models with a controlled study of different training strategies and whether they all lead to simultaneous improvements in performance and in the ability to assess quality of performance. However, this study is significantly beyond the scope of this paper, and we leave it for future work.

References

Zara Abrams. 2024. Addressing equity and ethics in artificial intelligence. *Monitor on Psychology*, 55(3):24–29.

Avinash Anand, Akshit Gupta, Nishchay Yadav, and Shaurya Bajaj. 2024. A comprehensive survey of ai-driven advancements and techniques in automated program repair and code generation. *arXiv preprint arXiv:2411.07586*.

Chengkun Cai, Xu Zhao, Haoliang Liu, Zhongyu Jiang, Tianfang Zhang, Zongkai Wu, Jenq-Neng Hwang, Serge Belongie, and Lei Li. 2024. The role of deductive and inductive reasoning in large language models. *arXiv preprint arXiv:2410.02892*.

Federico Cassano, John Gouwar, Francesca Lucchetti, Claire Schlesinger, Anders Freeman, Carolyn Jane Anderson, Molly Q Feldman, Michael Greenberg, Abhinav Jangda, and Arjun Guha. 2024. Knowledge transfer from high-resource to low-resource programming languages for code llms. *Proceedings of the ACM on Programming Languages*, 8(OOPSLA2):677–708.

Liguo Chen, Qi Guo, Hongrui Jia, Zhengran Zeng, Xin Wang, Yijiang Xu, Jian Wu, Yidong Wang, Qing Gao, Jindong Wang, and 1 others. 2024a. A survey on evaluating large language models in code generation tasks. *arXiv preprint arXiv:2408.16498*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Minshuo Chen, Song Mei, Jianqing Fan, and Mengdi Wang. 2024b. An overview of diffusion models: Applications, guided generation, statistical rates and optimization. *arXiv preprint arXiv:2404.07771*.

Michael Chromik, Malin Eiband, Felicitas Buchner, Adrian Krüger, and Andreas Butz. 2021. I think i get your point, ai! the illusion of explanatory depth in explainable ai. In *Proceedings of the 26th International Conference on Intelligent User Interfaces*, pages 307–317.

Jonathan Cordeiro, Shayan Noei, and Ying Zou. 2024. An empirical study on the code refactoring capability of large language models. *arXiv preprint arXiv:2411.02320*.

David Dunning, Kerri Johnson, Joyce Ehrlinger, and Justin Kruger. 2003. Why people fail to recognize their own incompetence. *Current directions in psychological science*, 12(3):83–87.

Rodney K Edwards, Kenneth R Kellner, Christopher L Sistrom, and Elizabeth J Magyari. 2003. Medical student self-assessment of performance on an obstetrics and gynecology clerkship. *American journal of obstetrics and gynecology*, 188(4):1078–1082.

Joyce Ehrlinger, Kerri Johnson, Matthew Banner, David Dunning, and Justin Kruger. 2008. Why the unskilled are unaware: Further explorations of (absent) self-insight among the incompetent. *Organizational behavior and human decision processes*, 105(1):98–121.

Arpad E Elo and Sam Sloan. 1978. The rating of chessplayers: Past and present. *(No Title)*.

Jiahui Geng, Fengyu Cai, Yuxia Wang, Heinz Koeppl, Preslav Nakov, and Iryna Gurevych. 2023. A survey of confidence estimation and calibration in large language models. *arXiv preprint arXiv:2311.08298*.

Alessandro Giagnorio, Alberto Martin-Lopez, and Gabriele Bavota. 2025. Enhancing code generation for low-resource languages: No silver bullet. *arXiv preprint arXiv:2501.19085*.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, and 1 others. 2024. A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594*.

Daniel E Haun, Andrea Zeringue, Argie Leach, and Angela Foley. 2000. Assessing the competence of specimen-processing personnel. *Laboratory Medicine*, 31(11):633–637.

Ralf Herbrich, Tom Minka, and Thore Graepel. 2006. Trueskill™: a bayesian skill rating system. *Advances in neural information processing systems*, 19.

Brian Hodges, Glenn Regehr, and Dawn Martin. 2001. Difficulties in recognizing one's own incompetence: novice physicians who are unskilled and unaware of it. *Academic Medicine*, 76(10):S87–S89.

Nam Huynh and Beiyu Lin. 2025. Large language models for code generation: A comprehensive survey of challenges, techniques, evaluation, and applications. *arXiv preprint arXiv:2503.01245*.

Hamed Jelodar, Mohammad Meymani, and Roozbeh Razavi-Far. 2025. Large language models (llms) for source code analysis: applications, models and datasets. *arXiv preprint arXiv:2503.17502*.

Yiqiao Jin, Qinlin Zhao, Yiyang Wang, Hao Chen, Kaijie Zhu, Yijia Xiao, and Jindong Wang. 2024. Agentreview: Exploring peer review dynamics with llm agents. *arXiv preprint arXiv:2406.12708*.

Peter Juslin. 1993. An explanation of the hard-easy effect in studies of realism of confidence in one's general knowledge. *European Journal of Cognitive Psychology*, 5(1):55–71.

Justin Kruger and David Dunning. 1999. Unskilled and unaware of it: how difficulties in recognizing one's own incompetence lead to inflated self-assessments. *Journal of personality and social psychology*, 77(6):1121.

Richard N Landers and Tara S Behrend. 2023. Auditing the ai auditors: A framework for evaluating fairness and bias in high stakes ai predictive models. *American Psychologist*, 78(1):36.

Daniel T Levin, Nausheen Momen, Sarah B Drivdahl IV, and Daniel J Simons. 2000. Change blindness blindness: The metacognitive error of overestimating change-detection ability. *Visual cognition*, 7(1-3):397–412.

Haitao Li, Qian Dong, Junjie Chen, Huixue Su, Yujia Zhou, Qingyao Ai, Ziyi Ye, and Yiqun Liu. 2024a. Llms-as-judges: a comprehensive survey on llm-based evaluation methods. *arXiv preprint arXiv:2412.05579*.

Jia Li, Yuqi Zhu, Yongmin Li, Ge Li, and Zhi Jin. 2024b. Showing llm-generated code selectively based on confidence of llms. *arXiv preprint arXiv:2410.03234*.

Jan R Magnus and Anatoly A Peresetsky. 2022. A statistical explanation of the dunning–kruger effect. *Frontiers in Psychology*, 13:840180.

Matan Mazor and Stephen M Fleming. 2021. The dunning-kruger effect revisited. *Nature Human Behaviour*, 5(6):677–678.

Don A Moore, Amelia S Dev, and Ekaterina Y Goncharova. 2018. Overconfidence across cultures. *Collabra: Psychology*, 4(1).

Federico Mora, Justin Wong, Haley Lepe, Sahil Bhatia, Karim Elmaaroufi, George Varghese, Joseph E Gonzalez, Elizabeth Polgreen, and Sanjit Seshia. 2024. Synthetic programming elicitation for text-to-code in very low-resource programming and formal languages. *Advances in Neural Information Processing Systems*, 37:105151–105170.

Michael Muthukrishna, Joseph Henrich, Wataru Toyokawa, Takeshi Hamamura, Tatsuya Kameda, and Steven J Heine. 2018. Overconfidence is universal? elicitation of genuine overconfidence (ego) procedure reveals systematic differences across domain, task knowledge, and incentives in four populations. *PloS one*, 13(8):e0202288.

Ayman Odeh, Nada Odeh, and Abdul Salam Mohammed. 2024. A comparative review of ai techniques for automated code generation in software development: advancements, challenges, and future directions. *TEM Journal*, 13(1):726.

Ruchir Puri, David S Kung, Geert Janssen, Wei Zhang, Giacomo Domeniconi, Vladimir Zolotov, Julian Dolby, Jie Chen, Mihir Choudhury, Lindsey Decker, and 1 others. 2021. Codenet: A large-scale ai for code dataset for learning a diversity of coding tasks. *arXiv preprint arXiv:2105.12655*.

GitHub Ranking. 2025a. The top programming languages. Accessed: 5/29/2025.

IEEE Ranking. 2024. The top programming languages 2024. Accessed: 5/29/2025.

TIOBE Ranking. 2025b. The tiobe index for may 2025. Accessed: 5/29/2025.

Aadesh Salecha, Molly E Ireland, Shashanka Subrahmanya, João Sedoc, Lyle H Ungar, and Johannes C Eichstaedt. 2024. Large language models display human-like social desirability biases in big five personality surveys. *PNAS nexus*, 3(12):pgae533.

Maohao Shen, Subhro Das, Kristjan Greenewald, Prasanna Sattigeri, Gregory Wornell, and Soumya Ghosh. 2024. Thermometer: Towards universal calibration for large language models. *arXiv preprint arXiv:2403.08819*.

Ola Shorinwa, Zhiting Mei, Justin Lidard, Allen Z Ren, and Anirudha Majumdar. 2024. A survey on uncertainty quantification of large language models: Taxonomy, open research challenges, and future directions. *arXiv preprint arXiv:2412.05563*.

Vaishnavi Shrivastava, Ananya Kumar, and Percy Liang. 2025. Language models prefer what they know: Relative confidence estimation via confidence preferences. *arXiv preprint arXiv:2502.01126*.

Fengfei Sun, Ningke Li, Kailong Wang, and Lorenz Goette. 2025. Large language models are overconfident and amplify human bias. *arXiv preprint arXiv:2505.02151*.

Lindia Tjuatja, Valerie Chen, Tongshuang Wu, Ameet Talwalkwar, and Graham Neubig. 2024. Do llms exhibit human-like response biases? a case study in survey design. *Transactions of the Association for Computational Linguistics*, 12:1011–1026.

Athena Vakali and Nicoleta Tantalaki. 2024. Rolling in the deep of cognitive and ai biases. *arXiv preprint arXiv:2407.21202*.

Lucía Vicente and Helena Matute. 2023. Humans inherit artificial intelligence biases. *Scientific reports*, 13(1):15737.

Haoyan Yang, Yixuan Wang, Xingyin Xu, Hanyuan Zhang, and Yirong Bian. 2024. Can we trust llms? mitigate overconfidence bias in llms through knowledge transfer. *arXiv preprint arXiv:2405.16856*.

Jiayi Ye, Yanbo Wang, Yue Huang, Dongping Chen, Qihui Zhang, Nuno Moniz, Tian Gao, Werner Geyer, Chao Huang, Pin-Yu Chen, and 1 others. 2024. Justice or prejudice? quantifying biases in llm-as-a-judge. *arXiv preprint arXiv:2410.02736*.

Han Zhou, Xingchen Wan, Ruoxi Sun, Hamid Palangi, Shariq Iqbal, Ivan Vulić, Anna Korhonen, and Sercan Ö Arık. 2025. Multi-agent design: Optimizing agents with better prompts and topologies. *arXiv preprint arXiv:2502.02533*.

A Measuring Relative Confidence

Here, we elaborate on the algorithms we use to convert pairwise confidence preferences into scalar relative confidence scores for each question. For every pair of questions q_i and q_j, we prompt the model to indicate which it is more confident in answering to produce a set P of pairwise preferences $q_i < q_j$. Below, we present the details of the ELO and TrueSkill methods to convert these preferences to scalar confidence scores.

A.1 Confidence Estimation Using Elo Rating

We treat each question as a "player" in a tournament, where each pairwise preference is interpreted as a match outcome. All questions are initialized with the same Elo score (Elo and Sloan, 1978) (in our case, 1000). For each preference pair $q_i < q_j \in P$, where q_i is the preferred (winning) question and q_j is the less preferred (losing) question, we compute the expected win probability using the logistic function:

$$P(i \text{ wins}) = \frac{1}{1 + 10^{(S_j - S_i)/K}}$$

where S_i and S_j are the Elo scores of questions q_i and q_j, and K is a sensitivity factor. The scores are then updated as follows:

$$S_i \leftarrow S_i + K \cdot (1 - P(i \text{ wins}))$$

$$S_j \leftarrow S_j - K \cdot P(i \text{ wins})$$

This update process is repeated for all preference pairs over multiple iterations to allow scores to converge. The final Elo scores are normalized using min-max scaling to the range $[0, 100]$ to produce interpretable confidence scores:

$$\text{Confidence}(q_i) = \frac{S_i - \min(S)}{\max(S) - \min(S)}$$

A.2 Confidence Estimation Using TrueSkill

As an alternative to Elo, we also implement confidence estimation using the TrueSkill rating system (Herbrich et al., 2006), which models each question's confidence as a Gaussian distribution over skill: $\mathcal{N}(\mu, \sigma^2)$, where μ represents the estimated confidence and σ the uncertainty.

For each preference pair $q_i < q_j \in P$, where q_i is preferred over q_j, we update the distributions of both questions using Bayesian inference. The update is performed using the TrueSkill factor graph model, which adjusts both μ and σ based on the observed outcome and the prior distributions. After processing all preference pairs, we extract the mean μ_i of each question's distribution as its raw confidence score. These scores are then min-max normalized to the range $[0, 100]$ as in the Elo method.

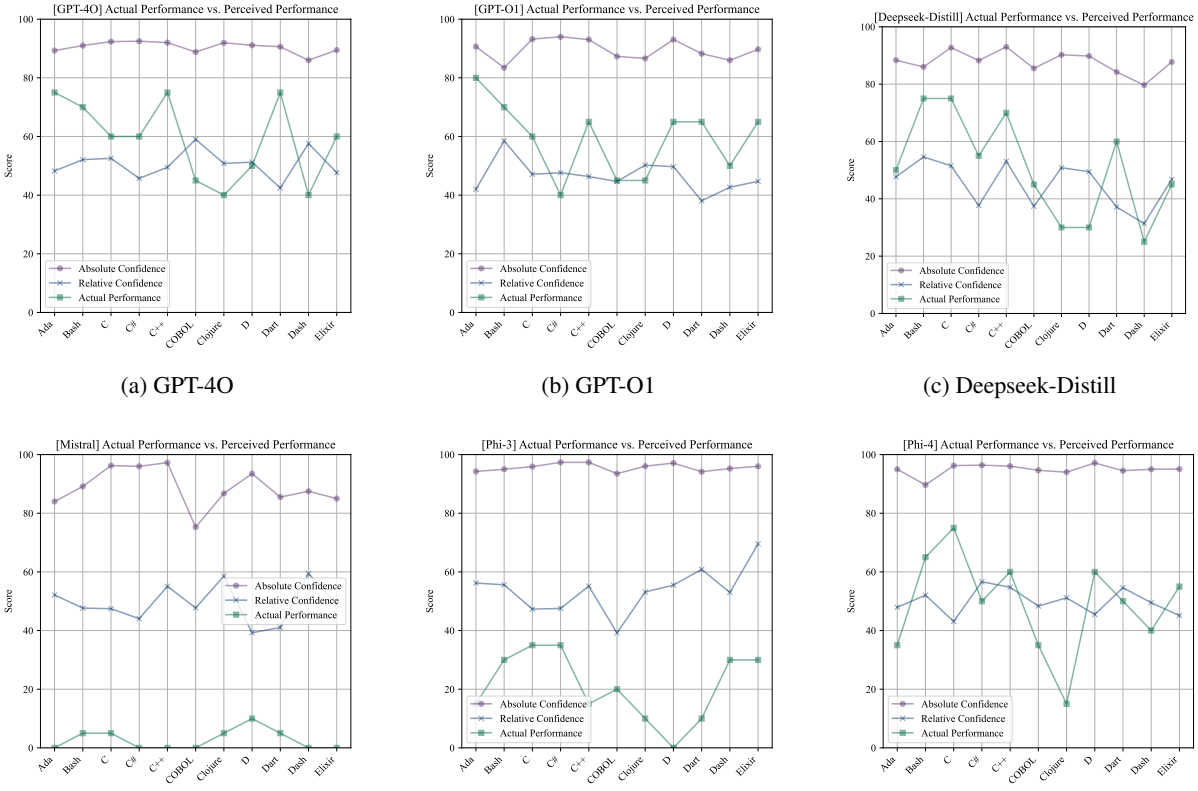
This methodology enables robust and interpretable confidence estimation by leveraging the model's relative preferences, rather than relying on coarse, absolute confidence scores.

A.3 Dataset

We create the MCQA problems for the codenet tasks (Puri et al., 2021) covering (1) code generation; (2) code understanding; (3) code syntax; and (4) code repair. For doing this scalably we Table 3 summarizes the number of tasks used for each domain.

Table 3: Data Statistics

Domain	***Number of Samples***
Ada	118
Bash	1000
C	897
C#	1000
C++	1000
COBOL	1000
Ceylon	90
Clojure	430
D	1000
Dart	195
Dash	155
Elixir	205
Erland	141
F#	1000
Fortran	1000
Go	1000
Haskell	1000
Java	1000
JavaScript	1000
Julia	1000
Lisp	1000
Kotlin	1000
Lua	1000
OCaml	1000
Objective-C	727
PHP	1000
Pascal	1000
Perl	1000
Prolog	231
Python	1000
Racket	145
Ruby	1000
Rust	1000
Scala	1000
Swift	1000
TypeScript	1000
Visual Basic	987



(a) GPT-4O (b) GPT-O1 (c) Deepseek-Distill

(d) Mistral (e) Phi-3 (f) Phi-4

Figure 3: Dunning-Kruger plots for various models.

A.4 Implementation Details

The programming languages include in the study are - Ada, Bash, C, C#, C++, COBOL, Ceylon, Clojure, D, Dart, Dash, Elixir, Erland, F#, Fortran, Go, Haskell, Java, JavaScript, Julia, Lisp, Kotlin, Lua, OCaml, Objective-C, PHP, Pascal, Perl, Prolog, Python, Racket, Ruby, Rust, Scala, Swift, TypeScript and Visual Basic. To generate the pairwise question preference data, we randomly sample 5 questions to generate multiple comparisons per question. The model's preferences are parsed to construct a directed graph of confidence judgements wherein each comparison yields a winner-loser pair, forming the basis for confidence ranking. For Elo rating ranking estimation, ratings are initialized randomly at 1000 for each question and updated iteratively based on outcome of each comparison. The final scores are normalized to a 0–100 scale. The win probability is scaled using a sensitivity factor, K which is set at 400 following hyperparameters selected in previous work (Shrivastava et al., 2025). The win probabilities are also estimated over 10 repetitions to allow scores to converge. In the TrueSkill ranking system, the questions are initialized with default values $\mu = 25.0$ and $\sigma = 8.333$, following standard TrueSkill settings implemented using the Python package. Similar to the Elo rating method, the rankings in the TrueSkill method are normalized to 0-100 scale.

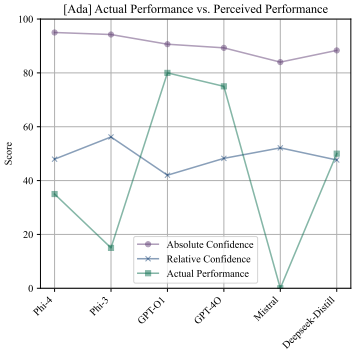
Models and Sizes We use GPT-4O (size unknown), GPT-O1 (size unknown), Deepseek-Distill (70B), Mistral (7B), Phi-3 (8B) and Phi-4 (20B) for this paper.

A.5 Inter-domain results for different models

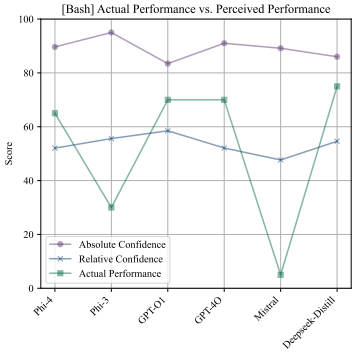
Figure 3 shows the individual plots for inter-model DKE for different domains. We see that the effect can be seen for all models across varying domains. For very small models like Mistral and Phi-3 (less than 8B) we see that the effect is less apparent as the models overall performance is very low and they generally overestimate their performance.

A.6 Inter-model results for different domains

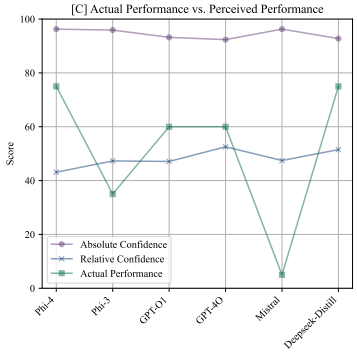
Figure 4 shows the individual plots for inter-model DKE for different domains. We see that the effect can be seen for all domains across varying models.



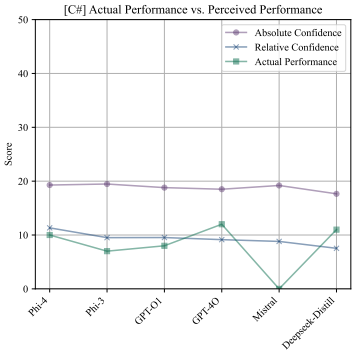
(a) Ada



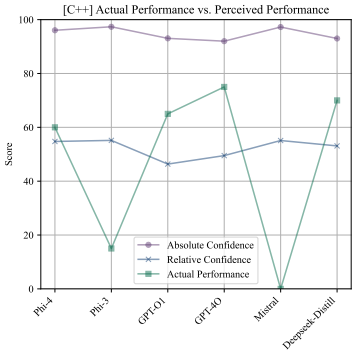
(b) Bash



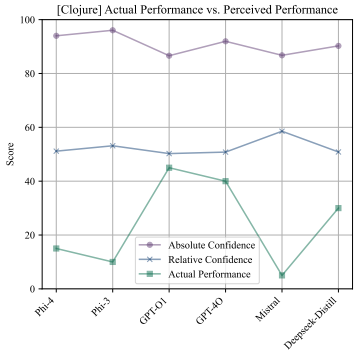
(c) C



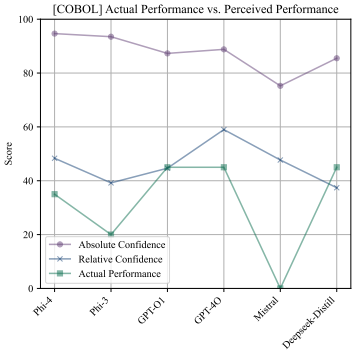
(d) C#



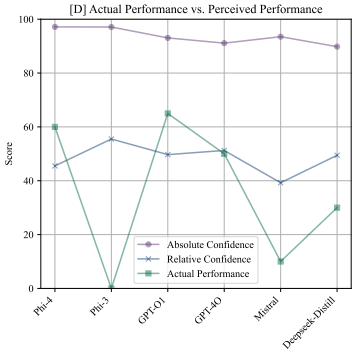
(e) C++



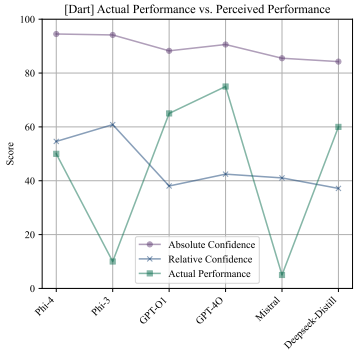
(f) Clojure



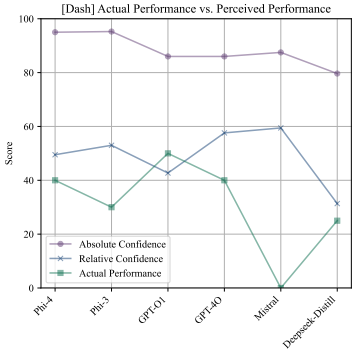
(g) COBOL



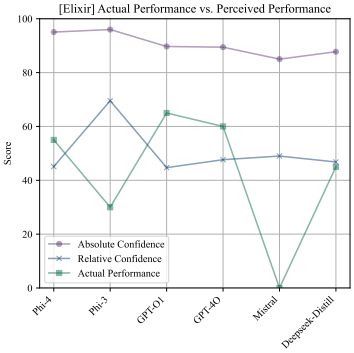
(h) D



(i) Dart



(j) Dash



(k) Elixir

Figure 4: Dunning-Kruger plots for various programming languages.