TimeSeriesExamAgent: Creating TimeSeries Reasoning Benchmarks at Scale

Malgorzata Gwiazda[1], **Yifu Cai**[2], **Mononito Goswami**[2], **and Artur Dubrawski**[2]

[1]Technical University of Munich, Munich, Germany
malgorzata.gwiazda@tum.de

[2]Auton Lab, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213
{yifuc, mgoswami, awd}@cs.cmu.edu

Abstract

We introduce `TimeSeriesExamAgent`, a scalable and domain-agnostic framework for automatically generating and validating time series reasoning benchmarks. Existing benchmarks lack scalability, are limited to a few specific domains, while building them remains labor intensive. Automated solutions for benchmark creation have been proposed, but these typically rely on a single-step generation process without verification, leading to lower-quality exams. Our framework addresses these limitations by enabling domain experts to easily create high-quality, domain-specific exams from their own datasets. A domain expert provides a dataset, a natural language description, and a simple data-loading method. The agent then orchestrates the generation pipeline, including creating question templates, robustness verification from multiple perspectives, and iterative refinement. We demonstrate the framework on three datasets from two diverse domains– healthcare and finance; and evaluate multiple state-of-the-art language models on the exams generated by `TimeSeriesExamAgent`. Empirically, we demonstrate that the framework produces domain-agnostic benchmarks whose diversity matches human-generated counterparts, and our evaluation of several Large Language Models shows that accuracy remains limited, underscoring open challenges in time-series reasoning.

1 Introduction

Many recent works have applied Large Language Models (LLMs) to time series analysis tasks such as forecasting, anomaly detection, and classification [1, 2, 3, 4, 5, 6]. More recently, attention has shifted to evaluating the reasoning capabilities of LLMs in time series tasks. These evaluations are typically framed in two ways: 1) contextualized traditional tasks such as forecasting, but with added contextual information (e.g., providing a clinical scenario before a prediction) [7, 8, 9, 10, 11], and 2) reasoning and understanding tasks that directly probe concepts in time series (e.g., "what kind of trend does the following series exhibit?") [12, 13].

However, existing benchmarks have clear limitations. Contextualized tasks remain close to traditional metrics (e.g., mean-squared-error for forecasting) without testing deeper reasoning, while reasoning-style benchmarks often focus only on simple properties like trend or seasonality. In practice, real-world domains such as healthcare require more complex reasoning, where tasks like diagnosis naturally combine anomaly detection, classification, and domain knowledge. Curation is another challenge. Annotation or template-based benchmarks are labor-intensive, while LLM-based augmentation often lacks diversity because it simply expands existing datasets. As a result, building specialized, domain-specific benchmarks remains difficult and time-consuming.

NeurIPS 2025 Workshop on Recent Advances in Time Series Foundation Models (BERT^{2}S).

Inspired by other domains, recent works have begun to use agents to automate benchmark construction and have shown promising results [14, 15]. In this work, we propose `TimeSeriesExam Agent`, a pipeline that 1) generates domain-specific multiple-choice questions on time series data, 2) scales efficiently, and 3) ensures reliable ground truth. We also evaluate 4 state-of-the-art LLMs on a benchmark of 531 questions. Our results show that 1) different models perform well in different domains of questions, and 2) all models struggle with highly complex questions that require combining domain knowledge with reasoning. For brevity, we provide detailed related work in Appendix A

2 TimeSeriesExamAgent

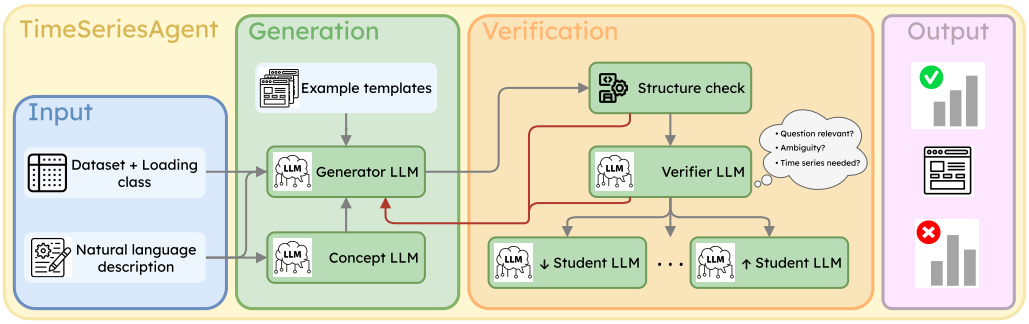


Figure 1: TimeSeriesExamAgent architecture. The user provides exam-making instructions and a custom dataset with minimal loading code. Agent outputs question templates – Python functions generated by a generator LLM and filtered through three progressive stages of verification (syntax and output format check, validation by LLM judge, capability-aligned filtering). Arrows denote data flow, red ones show direction for rejected templates.

Domain experts are often interested in assessing LLMs on specialized reasoning capabilities rather than on broad, preexisting benchmarks (e.g., evaluating anomaly detection in ECG data versus generic healthcare reasoning). To this end, they typically possess domain-specific datasets and wish to construct benchmarks that reflect the reasoning challenges within these datasets. However, building such benchmarks manually is labor-intensive. To address this challenge, we propose `TimeSeriesExamAgent`, a multi-agent framework that combines planning, generation, and verification to enable automatic benchmark construction. In this section, we describe `TimeSeriesExamAgent` and its workflow in detail. An overview is shown in Fig. 1. The Generation Agent takes as input a description of the natural language task T and a data set D. The description T may include user guidelines for generation, contextual information about the dataset, or other relevant instructions. For convenience, we denote each sample in D as (x_i, z_i), where $x_i \in \mathbb{R}^{n \times d}$ is a time series with n observations and d variables, and z_i is an auxiliary array containing metadata or labels related to the series. The user provides a dataset class D that supports basic operations such as querying the i-th sample.

Generation We generate question templates instead of samples directly, as shown in Fig. 2. Templates offer two advantages: they are scalable, and their abstraction adds an extra layer of robustness. By relying on structured, rule-based generation rather than manual inputs, they reduce the chance of human errors or inconsistencies. Our generator LLM produces a predefined number of templates, each implemented as a Python function. A template contains a formatted string for the question and options, together with parameters that control how many questions to generate. For each question, the template samples a pair (x_i, z_i) from the dataset D and applies a rule-based calculation to determine the correct answer from the time series. For example, in a trend-detection template, the function computes the linear trend coefficient of x_i and selects "Yes, there is a linear trend" if the coefficient exceeds a specified threshold. In addition to such signal-derived logic, templates can also utilize the auxiliary property z_i, effectively transforming classification problems into question–answer form. For instance, if an ECG series in the dataset is labeled as exhibiting atrial fibrillation, the template can present this label as one of the multiple-choice options. Each generated sample consists of the question, its options, the correct answer, and one or more associated time series represented as

numerical values. We provide a breakdown of the Generation Agent and its prompt in Appendix B. An example template is also provided.

Verification We observe that LLM-based generation frequently produces errors or irrelevant outputs, motivating the need for a structured verification process. We propose a multistage verification process to check the accuracy and relevance of each template. If a template fails at any stage, it is returned to the generation agent with feedback. The generation is iterative with a maximum of three attempts, after which the ongoing template is discarded to avoid excessive context length and cost from repeated failures.

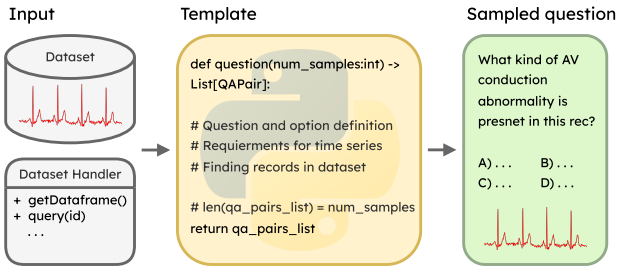


Figure 2: Question generation process: With information about dataset, TimeSeriesAgent generates question template in a form of Python functions. The created function can be called to get arbitrary number of question samples.

Structure verification We check whether the generated template can be executed successfully. We execute the generated template $k = 5$ times; if there are failures, the error message is returned as a feedback.

Content verification Certain aspects of quality control are particularly well-suited for LLM-as-a-judge evaluation. For example, verifying that a question is grammatically correct, free of ambiguity or bias, and genuinely answerable from the accompanying time series can be effectively handled by an LLM. To this end, we use an LLM verifier to assess the validty of each template. A quantitative score is given, and we set a threshold for rejection. If the verifier raises any rejection, its explanation is treated similarly to a structural error and the template is regenerated. We provide the detailed prompt in Appendix C.

Capability-Aligned Filtering To detect templates that generate overly simple or irrelevant exams, we evaluate them using a set of test-taking LLMs with varying capabilities. This approach is inspired by educational theory, particularly the expertise reversal effect [16]. A template is discarded if weaker LLMs achieve higher average accuracy than stronger models, as this typically indicates that the template is flawed or noisy rather than genuinely discriminative. Templates are retained if performance scales with model capability– or if all models perform poorly, since such questions may still capture genuine difficulty. We provide hyper-parameters in Appendix F and other design specifics in Appendix D

3 Experimental Setup, Results and Discussion

First, we generate one exam for each of the three real world datasets: PTB-XL [17], MIT-BIH [18], and yahoo finance stock dataset [19]. In total, we have 209 samples for YFinance, 197 samples for MIT-BIH, and 151 samples for PTB-XL. We sample 4 or 5 instances per template. Thus, the difference in the number of generated samples is a result of the template filtering mechanism above.

We select candidate models to cover a diverse range of performance levels, as indicated by the OpenVLM Leaderboard [24]. The time series were provided as plots, since such method gave better results in recent work [12]. In Table 1, we find that while general-purpose multimodal models such as `gpt-4o` perform well on finance-related questions, their performance is weaker on healthcare benchmarks. This contrast could suggest that the general reasoning ability does not always transfer across domains, particularly when tasks require domain-specific expertise or fine-grained interpretation of physiological signals.

To further evaluate our benchmark, we compare multiple metrics on questions generated from the dataset with those in `ECG-QA` [10], a template-based benchmark also built on `PTB-XL`. The goal is to demonstrate that our framework achieves comparable diversity without requiring manual template curation. We picked random 50 question samples from each benchmark and calculated the distances for every possible pair within the set. We used the `Qwen/Qwen3-Embedding-8B` sentence

Model	Dataset			Average
	MIT-BIH	PTB-XL	YFinance	
gpt-4o [20]	0.416	0.424	**0.586**	0.473
o3-mini [21]	0.442	0.477	0.555	0.491
Qwen2.5-VL-Instruct [22]	0.411	0.490	0.572	0.491
Gemma-3-27b-it [23]	**0.497**	**0.517**	0.534	**0.516**

Table 1: Comparative performance of four vision–language models across medical (MIT-BIH, PTB-XL) and financial (YFinance) time-series datasets. The results highlight dataset-specific strengths and reveal Gemma-3-27B as the most consistent performer overall. Nonetheless, all models achieve less than 55 mean accuracy, underscoring the difficulty of time-series reasoning for current VLMs. The evaluation protocol is provided in Appendix G

transformer model to extract embeddings, as it achieved the second-best performance among all models on the Hugging Face MTEB leaderboard.

Benchmark Dataset	Mean $\pm$ Std	
	Embedding	Normalized Levenshtein
ECG-QA	0.207 ± 0.079	0.519 ± 0.157
TimeSeriesExamAgent (ours)	$\mathbf{0.301 \pm 0.070}$	$\mathbf{0.542 \pm 0.039}$

Table 2: Question diversity comparison using embedding and normalized Levenshtein distance.

As shown in Table 2, benchmark generated by our framework shows a diversity comparable to one developed by humans. This indicates that the proposed framework is able to capture a wide range of expressions without relying on handcrafted templates, supporting its scalability and adaptability to other domains.

We also employed G-Eval, a probabilistic LLM-as-a-judge framework [25]. An LLM is used to evaluate the relevance of each question, assigning a score between 0 and 1 to indicate how well it meets the specified criteria. Results are presented in Table 3. We provide the detailed G-Eval prompt in Appendix E.

Dataset	Mean Result			
	Specificity	Unambiguity	Domain Relevance	Answerability
ECG-QA	0.604	0.562	0.827	0.898
TimeSeriesExamAgent (ours)	**0.922**	**0.932**	**0.989**	**0.992**

Table 3: Question diversity comparison using G-Eval framework.

4 Limitations and Conclusions

In this work, we present a scalable, domain-specific framework for the automatic generation of time-series benchmarks, enabling the creation of high-quality, large-scale evaluation datasets while minimizing the need for labor-intensive human annotation. A limitation of this study is that the quality of the generated exams depends on the quality and coverage of the time series dataset. Additionally, domain specialists must provide carefully crafted prompts.

For future work, we will explore human-in-the-loop improvements to template generation. In offline sessions with clinicians, we observed that exams produced with such feedback are more likely to be deemed valid. We also plan to validate exam quality by training time series–text alignment models and testing their transfer performance on other established reasoning benchmarks [8]. Finally, there is growing attention on building time series agentic frameworks [26, 27]. Enabling these frameworks to write code in order to answer our benchmark questions would provide valuable insights to the community.

Acknowledgments

This work has been partially supported by the National Science Foundation (awards 2427948, 2406231 and 2530752) and National Institutes of Health (awards R01NR013912 and R01NS124642) and with the gift from Dr. Chirag Nagpal.
We plan to make the code publicly available in the future.

References

[1] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

[2] Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models. *arXiv preprint arXiv:2402.03885*, 2024.

[3] Defu Cao, Furong Jia, Sercan O Arik, Tomas Pfister, Yixiang Zheng, Wen Ye, and Yan Liu. Tempo: Prompt-based generative pre-trained transformer for time series forecasting. *arXiv preprint arXiv:2310.04948*, 2023.

[4] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, et al. Time-llm: Time series forecasting by reprogramming large language models. *arXiv preprint arXiv:2310.01728*, 2023.

[5] Tian Zhou, Peisong Niu, Liang Sun, Rong Jin, et al. One fits all: Power general time series analysis by pretrained lm. *Advances in neural information processing systems*, 36:43322–43355, 2023.

[6] Nina Żukowska, Mononito Goswami, Michał Wiliński, Willa Potosnak, and Artur Dubrawski. Towards long-context time series foundation models. *arXiv preprint arXiv:2409.13530*, 2024.

[7] Jialin Chen, Aosong Feng, Ziyu Zhao, Juan Garza, Gaukhar Nurbek, Cheng Qin, Ali Maatouk, Leandros Tassiulas, Yifeng Gao, and Rex Ying. Mtbench: A multimodal time series benchmark for temporal reasoning and question answering. *arXiv preprint arXiv:2503.16858*, 2025.

[8] Yaxuan Kong, Yiyuan Yang, Yoontae Hwang, Wenjie Du, Stefan Zohren, Zhangyang Wang, Ming Jin, and Qingsong Wen. Time-mqa: Time series multi-task question answering with context enhancement. *arXiv preprint arXiv:2503.01875*, 2025.

[9] Haoxin Liu, Shangqing Xu, Zhiyuan Zhao, Lingkai Kong, Harshavardhan Prabhakar Kamarthi, Aditya Sasanur, Megha Sharma, Jiaming Cui, Qingsong Wen, Chao Zhang, et al. Time-mmd: Multi-domain multimodal dataset for time series analysis. *Advances in Neural Information Processing Systems*, 37:77888–77933, 2024.

[10] Jungwoo Oh, Gyubok Lee, Seongsu Bae, Joon-myoung Kwon, and Edward Choi. Ecg-qa: A comprehensive question answering dataset combined with electrocardiogram. *Advances in Neural Information Processing Systems*, 36:66277–66288, 2023.

[11] Xu Wang, Jiaju Kang, Puyu Han, Yubao Zhao, Qian Liu, Liwenfei He, Lingqiong Zhang, Lingyun Dai, Yongcheng Wang, and Jie Tao. Ecg-expert-qa: A benchmark for evaluating medical large language models in heart disease diagnosis. *arXiv preprint arXiv:2502.17475*, 2025.

[12] Yifu Cai, Arjun Choudhry, Mononito Goswami, and Artur Dubrawski. Timeseriesexam: A time series understanding exam. *arXiv preprint arXiv:2410.14752*, 2024.

[13] Willa Potosnak, Cristian Challu, Mononito Goswami, Kin G. Olivares, Michał Wiliński, Nina Żukowska, and Artur Dubrawski. Investigating compositional reasoning in time series foundation models, 2025.

[14] Natasha Butt, Varun Chandrasekaran, Neel Joshi, Besmira Nushi, and Vidhisha Balachandran. Benchagents: Automated benchmark creation with agent interaction. *arXiv preprint arXiv:2410.22584*, 2024.

[15] Gauthier Guinet, Behrooz Omidvar-Tehrani, Anoop Deoras, and Laurent Callot. Automated evaluation of retrieval-augmented language models with task-specific exam generation. *arXiv preprint arXiv:2405.13622*, 2024.

[16] Slava Kalyuga. Expertise reversal effect and its implications for learner-tailored instruction. *Educational psychology review*, 19(4):509–539, 2007.

[17] Patrick Wagner, Nils Strodthoff, Ralf-Dieter Bousseljot, Dieter Kreiseler, Fatima I Lunze, Wojciech Samek, and Tobias Schaeffter. Ptb-xl, a large publicly available electrocardiography dataset. *Scientific data*, 7(1):1–15, 2020.

[18] George B Moody and Roger G Mark. The impact of the mit-bih arrhythmia database. *IEEE engineering in medicine and biology magazine*, 20(3):45–50, 2001.

[19] Ranan Roussi. yfinance: Yahoo! finance market data downloader. `https://github.com/ranaroussi/yfinance`, 2017. Accessed: 2025-08-22.

[20] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[21] OpenAI. Openai o3-mini system card. `https://openai.com/index/o3-mini-system-card/`, January 2025. Accessed: 2025-08-22.

[22] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[23] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

[24] Haodong Duan, Junming Yang, Yuxuan Qiao, Xinyu Fang, Lin Chen, Yuan Liu, Xiaoyi Dong, Yuhang Zang, Pan Zhang, Jiaqi Wang, et al. Vlmevalkit: An open-source toolkit for evaluating large multi-modality models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 11198–11201, 2024.

[25] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment. *arXiv preprint arXiv:2303.16634*, 2023.

[26] Yifu Cai, Xinyu Li, Mononito Goswami, Michał Wiliński, Gus Welter, and Artur Dubrawski. Timeseriesgym: A scalable benchmark for (time series) machine learning engineering agents. *arXiv preprint arXiv:2505.13291*, 2025.

[27] Wen Ye, Wei Yang, Defu Cao, Yizhou Zhang, Lumingyuan Tang, Jie Cai, and Yan Liu. Domain-oriented time series inference agents for reasoning and automated analysis. *arXiv preprint arXiv:2410.04047*, 2024.

[28] Yilin Wang, Peixuan Lei, Jie Song, Yuzhe Hao, Tao Chen, Yuxuan Zhang, Lei Jia, Yuanxiang Li, and Zhongyu Wei. Itformer: Bridging time series and natural language for multi-modal qa with large-scale multitask dataset. *arXiv preprint arXiv:2506.20093*, 2025.

[29] Wanying Wang, Zeyu Ma, Pengfei Liu, and Mingang Chen. Testagent: A framework for domain-adaptive evaluation of llms via dynamic benchmark construction and exploratory interaction. *arXiv preprint arXiv:2410.11507*, 2024.

[30] Mike A Merrill, Mingtian Tan, Vinayak Gupta, Tom Hartvigsen, and Tim Althoff. Language models still struggle to zero-shot reason about time series. *arXiv preprint arXiv:2404.11757*, 2024.

[31] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[32] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[33] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[34] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020.

[35] Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens, 2024.

[36] Qingyue Wang, Yanhe Fu, Yanan Cao, Shuai Wang, Zhiliang Tian, and Liang Ding. Recursively summarizing enables long-term dialogue memory in large language models, 2025.

[37] Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. Openhands: An open platform for ai software developers as generalist agents. *arXiv preprint arXiv:2407.16741*, 2024.

A Related Work

Time series benchmarks The task of creating domain-specific time series reasoning benchmarks is challenging. Existing benchmarks are either domain-agnostic, or limited to a specific domains with high quality datasets. For example, `TimeSeriesExam` [12] introduced over 700 multiple-choice questions to evaluate five general reasoning skills, but its questions primarily assess signal properties (e.g. trend, cyclicity, stationarity) and lack the contextual depth needed for real-world applications. Domain-specific benchmarks address this gap but have limited scope and poor extensibility, since their curation often relies on templates. For instance, ECG-QA [10] and ECG-Expert-QA [11] focus on ECG interpretation, while EngineMT-QA [28] targets industrial settings. Automatic benchmark generation offers a scalable alternative but raises concerns about quality and diversity of generated questions. Without extensive verification, LLM-generated questions often require heavy manual curation [8, 9], which is both difficult and time-consuming—undermining the main advantage of automation.

Title	Multi-Domain	Curation	# Samples	Skill type		
		Fully Automatic		P	R	PS
Time-MQA [8]	✓	✓	200,000	✓	✓	✗
TimeSeriesExam [12]	✗	✗	763	✓	✓	✗
Time-MMD [9]	✓	✗	17,113	✓	✗	✗
MT-Bench [7]	✓	✗	22,000	✓	✓	✗
ECG-QA [10]	✗	✗	414,348	✓	✓	✓
TimeSeriesExamAgent (ours)	✓	✓	600+	✓	✓	✓

Table 4: Overview of time series and multimodal datasets with curation and skill types (P – Prediction, R – Reasoning, P – Practical skills. TimeSeriesExamAgent is universal – tailored to user's needs and with advanced automatic verifications.

Agents for benchmark creation An AI agent is an autonomous system that can observe its environment, reason about possible actions, and act toward achieving a goal. In LLM-based settings, the language model often provides the reasoning or planning layer that guides the agent's decisions. Recent work has shown success in using agents for automatic benchmark creation. Most solutions adopt a multi-agent pipeline with planning, generation, validation, and evaluation modules [14]. For instance, [29] integrates exploratory evaluation using reinforcement learning, while [14] takes a natural language task description as input. However, most of these approaches are not tailored to time series and struggle to generate questions conditioned on numeric data. One recent solution does incorporate time series but is limited to single-step design and lacks extensive verification [30].

B Generation Agent Workflow

We rely on two stages of generation for the templates: planning and generating, inspired by the chain-of-thought (CoT) prompting[31].

Generation planning To provide a relevant and diverse set of templates, we rely on a comprehensive list of domain-specific concepts. There are several ways our pipeline generates a list of concepts:

1. LLM generation: User guidelines and dataset descriptions are provided as input to an LLM, which proposes the concepts.
2. Web Search: We provide the option for generator LLM obtain concepts through web search.
3. Retrieval Augmented Generation: As an option, the user could also provide a relevant file from which the LLM reads and generates concepts[32].

Template generation As input to our generator, the following components are provided:

- User-provided guidelines: a document containing the user's goal or specific requirements,
- Dataset description: a list of columns and example values with ranges from the dataset, with a short usage example,
- List of concepts: generated in previous step. For each template, our pipeline will choose a concept at random to ensure diversity.
- Example templates[Optional]: user-provided few-shot examples presenting required structural elements [33].

B.1 Generation Prompt

```
Here is the goal of the exam questions:
{user_info_text}

Here are sample concepts on which you can base your question generation:
{concept_conversation}

Use the concept numbered {concept_no} from the list to guide the design of
    your question template.

Here is the description of the dataset you will use to generate the
    question:
{dataset_describe}

In your template, use the provided ‘user_dataset‘ object. Use its ‘query(
    index)‘ method to load relevant time series data.

Do not select time series randomly. First, formulate the question, and then
     choose a time series that fits its logic and reasoning needs.

Generate one function-based question template now.
```

B.2 Example of Question Template

```
def question_6(num_samples, verbose=False):
    hyperparameters = {
        "min_trend_days": 20,
        "max_series_length": 3000,
        "trend_strength_threshold": 0.7,
        "momentum_window": 10,
    }

    question = "Analyzing the price movements of {ticker} over the given
    time period, does the price trend demonstrate strong momentum and
    sustainability, or does it show signs of weakness and potential
    reversal?"

    options = [
        "The trend shows strong momentum with consistent directional
    movement and minimal pullbacks, suggesting the trend is likely to
    continue.",
        "The trend shows signs of weakness with frequent reversals and
    inconsistent momentum, suggesting a potential trend change.",
        "The trend shows mixed signals with alternating periods of strength
     and weakness, making direction unclear.",
        "The price movement shows no clear trend pattern, indicating a
    ranging or sideways market."
    ]

    def calculate_trend_strength(prices):
        if len(prices) < hyperparameters["min_trend_days"]:
            return None, None

        returns = np.diff(prices) / prices[:-1]

        # Calculate momentum consistency
        positive_days = np.sum(returns > 0)
        negative_days = np.sum(returns < 0)
        total_days = len(returns)

        directional_consistency = max(positive_days, negative_days) /
    total_days

        # Calculate average magnitude of moves
        avg_abs_return = np.mean(np.abs(returns))

        # Calculate trend persistence (consecutive moves in same direction)
        consecutive_moves = []
        current_streak = 1
        for i in range(1, len(returns)):
            if np.sign(returns[i]) == np.sign(returns[i-1]):
                current_streak += 1
            else:
                consecutive_moves.append(current_streak)
                current_streak = 1
        consecutive_moves.append(current_streak)

        avg_streak = np.mean(consecutive_moves)
        max_streak = max(consecutive_moves)

        # Determine overall trend direction
```

```
    overall_return = (prices[-1] - prices[0]) / prices[0]
    trend_direction = "up" if overall_return > 0 else "down"

    return {
        "directional_consistency": directional_consistency,
        "avg_abs_return": avg_abs_return,
        "avg_streak": avg_streak,
        "max_streak": max_streak,
        "overall_return": abs(overall_return),
        "trend_direction": trend_direction
    }, returns

qa_pairs = []
df = user_dataset.get_dataframe()

attempted_tickers = set()

while len(qa_pairs) < num_samples:
    if verbose:
        print(f"[Question 6] Generating question {len(qa_pairs)} / {
num_samples}")

    # Select a ticker that hasn't been attempted
    available_tickers = [i for i in df.index if i not in
attempted_tickers]
    if not available_tickers:
        break

    ticker_id = random.choice(available_tickers)
    attempted_tickers.add(ticker_id)

    ticker = df.loc[ticker_id, 'ticker']
    prices = user_dataset.query(ticker_id)

    if len(prices) < hyperparameters["min_trend_days"]:
        continue

    # Limit series length
    if len(prices) > hyperparameters["max_series_length"]:
        start_idx = random.randint(0, len(prices) - hyperparameters["
max_series_length"])
        prices = prices[start_idx:start_idx + hyperparameters["
max_series_length"]]

    # Select a subset for analysis (to make question more focused)
    analysis_length = min(len(prices), random.randint(50, 200))
    start_idx = random.randint(0, len(prices) - analysis_length)
    analysis_prices = prices[start_idx:start_idx + analysis_length]

    trend_metrics, returns = calculate_trend_strength(analysis_prices)
    if trend_metrics is None:
        continue

    # Determine answer based on trend strength metrics
    strength_score = (
        trend_metrics["directional_consistency"] * 0.4 +
        min(trend_metrics["avg_streak"] / 5, 1.0) * 0.3 +
        min(trend_metrics["overall_return"] * 10, 1.0) * 0.3
    )
```

```
        if strength_score >= hyperparameters["trend_strength_threshold"]
    and trend_metrics["max_streak"] >= 5:
            answer = options[0]
        elif strength_score < 0.4 or trend_metrics["directional_consistency
    "] < 0.6:
            answer = options[1]
        elif 0.4 <= strength_score < hyperparameters["
    trend_strength_threshold"]:
            answer = options[2]
        else:
            answer = options[3]

        question_text = question.format(ticker=ticker)

        qa_pairs.append({
            "question": question_text,
            "options": options,
            "answer": answer,
            "ticker": ticker,
            "ts": analysis_prices,
            "relevant_concepts": ["Volume-Price Trend Correlation", "Trend
    Strength Analysis", "Price Momentum"],
            "domain": "finance",
            "detractor_types": ["Incorrect trend interpretation", "
    Misunderstanding momentum signals"],
            "question_type": "multiple_choice",
            "format_hint": "Please answer the question and provide the
    correct option letter, e.g., [A], [B], [C], [D], and option content at
    the end of your answer. All information need to answer the question is
    given. If you are unsure, please provide your best guess.",
        })

    return qa_pairs
```

B.3 Example of Natural Language Description

```
I want to create time series exam testing model understanding of finance
    time series data.

To load the data, use the provided ‘‘‘user_dataset‘‘‘ object.

Given time series come from Yahoo Finance, include closing price of a stock.
     Interval between samples is 1 day.
Make sure that the length of time series (total number of samples of one or
     two time series) does not excide 3000.

Please make sure that exams cannot be answer without timeseries!
```

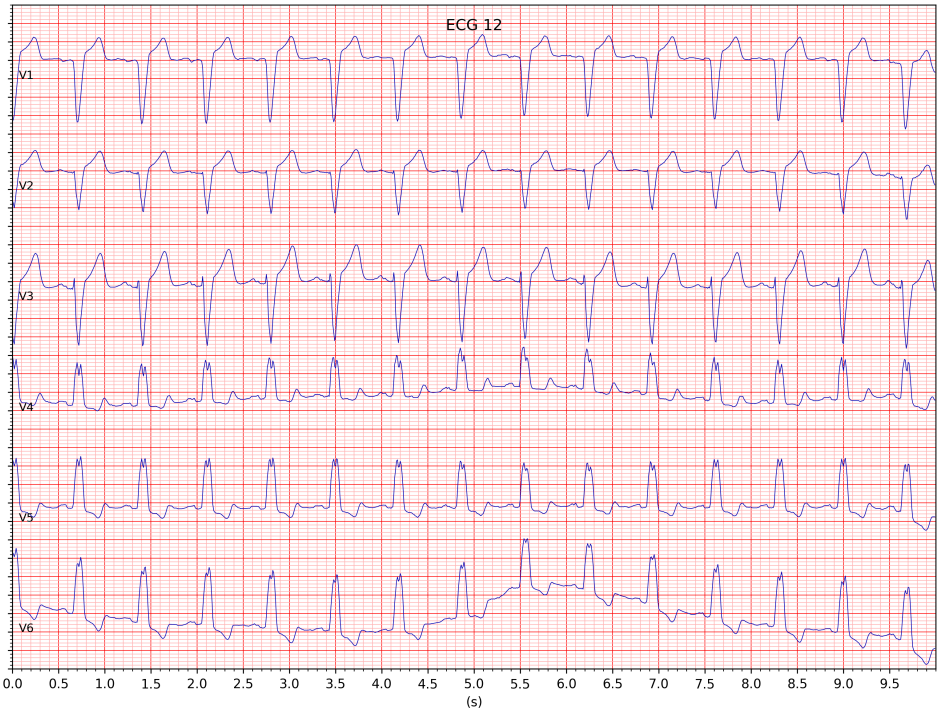
B.4 Examples of Generated Questions

ECG Question Example

Q: Analyze the P-wave morphology and amplitude characteristics in this recording. What atrial abnormality is present?

A. RAO/RAE: Right atrial overload/enlargement with prominent P-waves
B. LAO/LAE: Left atrial overload/enlargement with bifid P-waves
C. Normal P-wave morphology with no atrial abnormalities
D. Absent P-waves indicating atrial fibrillation

answer: LAO/LAE: Left atrial overload/enlargement with bifid P-waves


ECG 12
V1
V2
V3
V4
V5
V6
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5 9.0 9.5
(s)

Finance Question Example

Q: Based on the daily closing price data for MAA over the past 2000 trading days, what does the Relative Strength Index (RSI) analysis reveal about the stock's momentum condition at the end of the period?

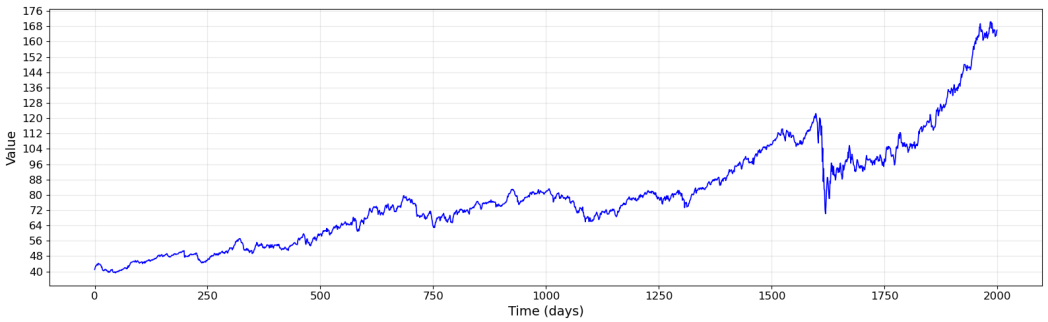
A. The stock is in overbought territory with RSI above 70, suggesting potential selling pressure.

B. The stock is in oversold territory with RSI below 30, suggesting potential buying opportunity.

C. The stock shows neutral momentum with RSI around 50, indicating balanced buying and selling pressure.

D. The stock shows strong upward momentum with RSI consistently increasing but not yet overbought.

answer: The stock shows neutral momentum with RSI around 50, indicating balanced buying and selling pressure.



C LLM Verifier

For each template, we use an LLM to evaluate the generated question. Specifically, we ask:

- Is the question relevant to the given concept?
- Does answering the question require the provided time series?
- Are the question and answer free from ambiguity and bias?

C.1 Validation Prompt

```
You are an expert validator of question templates involving reasoning over
{exam_type} time series data.
You are given an exam question template:

{exam_template}

Your task is to validate the question template using the following criteria:
1. Is the question relevant to {exam_type} time series analysis?
2. Would you need the time series itself to answer the question?
3. Are there no ambiguity in the question or its answer?

If the answer to all is YES or MOSTLY YES, return only the number 1.
If the answer to either is NO, return your objections.
Return 1 (do not include any additional text then) or describe your objections.
```

D Other Design Specifics

Detractors In addition, the mechanism of plausible but incorrect answer choices was implemented. The LLM is prompted to reflect on possible mistakes that the test taker might make while solving the exam. Using this knowledge, misleading, incorrect option choices can be generated.

Context Condensation A common issue we encountered in the framework was context window overflow during exam regeneration. To mitigate this, we applied context condensation, which reduces the number of tokens while preserving essential information. In our setup, the agent generates templates in a conversational manner. The process begins with a generation prompt, followed by a message containing the generated exam. If errors occur or the exam is rejected during verification, the feedback and regenerated exams are appended to the conversation. Several context condensation techniques exist, such as windowing [34] and context compression [35]. We adopt a summarization-based method [36, 37], which has shown strong results in prior work and fits our use case. Specifically, we summarize non-recent pairs of failing exams and error messages into short descriptions that highlight the issues encountered. These summaries provide the LLM with concise feedback, supporting the generation of higher-quality templates.

E G-Eval

We evaluated a set of generated questions under the G-Eval framework. We used the following criteria:

```
1. SPECIFICITY
Evaluate the specificity of the generated ECG multiple-choice question.

A good question should target a single phenomenon.

Evaluation steps:
1. Read the question and all answer options.
2. Determine if the question targets a single, clearly defined ECG finding
    or clinical interpretation.
3. Assess the ratio of unique medical terms to general words.
4. Penalize if:
   - The question is overly broad or open-ended (e.g., "Is this ECG normal
    ?").
   - The wording leaves diagnostic interpretation unclear.
   - The question covers multiple unrelated phenomena.

Score highest if the question has one precise focus (e.g., "Is there ST
    elevation in lead V3?").

1. UNAMBIGUITY
Evaluate the unambiguity of the generated ECG multiple-choice question.
A question and the answers should not have multiple interpretations.

Evaluation steps:
1. Read the question and all answer options.
2. Determine if the question can be objectively assessed.
3. Check if the answers are clear and unambiguous.
4. Penalize if:
   - The question uses subjective terms (e.g., "Does this look strange?").
   - The answers are open to multiple interpretations.
   - The question cannot be objectively answered.

Score highest if the question is clear and objective (e.g., "Is there
    tachycardia?"),

2. DOMAIN RELEVANCE
```

```
Evaluate the domain relevance of the generated ECG multiple-choice question.

Does the question actually pertain to ECGs and medicine?

Evaluation steps:
1. Read the question and all answer options.
2. Identify medical and ECG-specific terminology.
3. Determine if the question is relevant to ECG interpretation and medical
    diagnosis.
4. Penalize if:
   - The question contains non-medical terms (e.g., "Is the line pretty?").
   - The question is not related to ECG interpretation.
   - The question lacks medical context.

Score highest if the question contains relevant medical terms
(e.g., "QRS," "arrhythmia," "P wave") and pertains to ECG interpretation.

3. ANSWERABILITY
Evaluate the answerability of the generated ECG multiple-choice question.
Even without an answer provided, the question should be answerable based on
     the data (ECG).

Evaluation steps:
1. Read the question and all answer options.
2. Determine if the question can be answered by analyzing ECG waveform data.

3. Assess whether the question requires time series analysis or could be
    answered without it.
4. Penalize if:
   - The question asks about non-ECG factors (e.g., "Was the patient
    nervous?").
   - The question can be answered without analyzing the ECG time series
    data.
   - The question is too general and doesn't require specific ECG analysis.

Score highest if the question requires specific ECG time series analysis
(e.g., "Is there atrial fibrillation?").
Give fewer points if the question can be answered without time series data.
```

F Hyperparameters

In this section, we list all the hyperparameter used for our agentic workflow.

1. Generator LLM: the LLM use to generate concepts and the corresponding template. We used claude-sonnet-4-20250514 (initial generation with `reasoning_effort="medium"`).
2. Concept LLM: the LLM use to generate concepts. We used gpt-4o-2024-08-06.
3. Verifier LLM: the LLM use to verify templates. We used gpt-4o-2024-08-06.
4. Student LLMs: the student LLMs we use to check the exam differentiability. Currently we have two student LLMs: stronger: gpt-4o-2024-08-06 and weaker: gpt-4o-mini.
5. Exam type: We are generating the data connected to specific domain. We used "ECG" and "finance".
6. Few-shot examples: 3 templates prepared beforehand were used to present the desired structure. For each generation, they were randomly sampled from set of 9.

In such a setup, the generation of one template costs 0.09$ on average.

G Evaluation Protocol

All used models were accessed by API with `LiteLLM` Python library. The following API providers were used with default parameters:

- Closed source models – OpenAI API, Anthropic API
- Open source models – Hugging Face Inference Providers API

During the evaluation, the images of the plots were encoded with base64 encoding and provided to the models. Plots were created with DPI = 50. We used setup without context condensation.

H Ethical Concerns

All medical data were processed with respect for patient privacy and ethical standards, following anonymization and approved data protection procedures. Similar precautions should be taken when working with own medical datasets.