Promises and Pitfalls: Using Large Language Models to Generate Visualization Items

Yuan Cui, Lily W. Ge, Yiren Ding, Lane Harrison, Fumeng Yang, and Matthew Kay

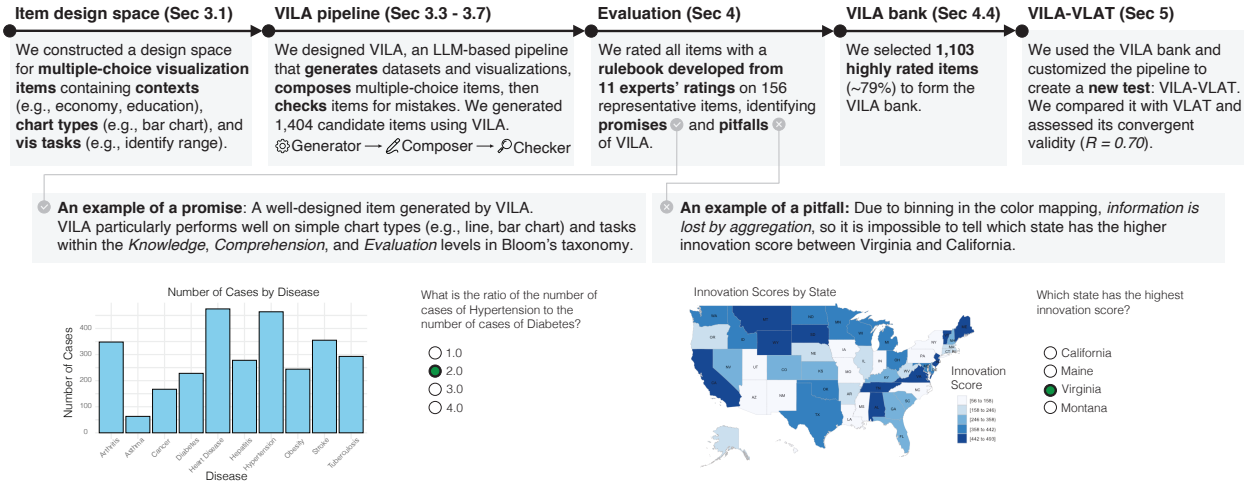


Fig. 1: Overview of this paper: developing the VILA pipeline, evaluating the candidate bank, and demonstrating a potential application—the new VILA-VLAT visualization literacy test.

Abstract—Visualization items—factual questions about visualizations that ask viewers to accomplish visualization tasks—are regularly used in the field of information visualization as educational and evaluative materials. For example, researchers of visualization literacy require large, diverse banks of items to conduct studies where the same skill is measured repeatedly on the same participants. Yet, generating a large number of high-quality, diverse items requires significant time and expertise. To address the critical need for a large number of diverse visualization items in education and research, this paper investigates the potential for large language models (LLMs) to automate the generation of multiple-choice visualization items. Through an iterative design process, we develop the VILA (**V**isualization **I**tems Generated by Large **LA**nguage Models) pipeline, for efficiently generating visualization items that measure people's ability to accomplish visualization tasks. We use the VILA pipeline to generate 1,404 candidate items across 12 chart types and 13 visualization tasks. In collaboration with 11 visualization experts, we develop an evaluation rulebook which we then use to rate the quality of all candidate items. The result is the VILA bank of ~1,100 items. From this evaluation, we also identify and classify current limitations of the VILA pipeline, and discuss the role of human oversight in ensuring quality. In addition, we demonstrate an application of our work by creating a visualization literacy test, VILA-VLAT, which measures people's ability to complete a diverse set of tasks on various types of visualizations; comparing it to the existing VLAT, VILA-VLAT shows moderate to high convergent validity (R = 0.70). Lastly, we discuss the application areas of the VILA pipeline and the VILA bank and provide practical recommendations for their use. All supplemental materials are available at `https://osf.io/ysrhq/`.

Index Terms—Visualization Items, Large Language Models, Visualization Literacy Assessment

1 Introduction

Visualization items—factual questions about visualizations that ask viewers to accomplish visualization tasks—are regularly used in the field of information visualization as educational and evaluative materials [4, 8, 18, 20, 28, 41]. Teachers of visualization courses require an extensive pool of materials for quizzes, practice problems, and exams; researchers require items to study and advance visualization literacy [13,18,20,28]; and developers of visualization question-answering (VQA) systems seek items as training and test data [23, 31, 32].

Many of these applications of visualization items require large and diverse item banks. For example, visualization literacy researchers may want to track skill development over time or to evaluate intervention efficacy through pre- and post-testing; both require large banks to avoid item overuse (the same item being administered to the same person multiple times). Visualization educators need large item banks to reduce the labor of repeatedly creating new materials and to avoid the risk of leaking homework or test materials. Item banks must also be diverse to ensure that measurement is not biased by the properties of any particular item or specific context, and educators need diverse item banks so that they can assess students' skills across a range of cognitive levels. For example, having items that cover all levels of Bloom's taxonomy [26] (a widely-used taxonomy of learning goals) would help students develop—and teachers assess—a variety of skills, all the way from simple value retrieval tasks to high-level tasks like judging the appropriateness of a visualization design.

However, the generation of high-quality, diverse banks of visualization items is a resource-intensive task. Significant expertise is needed to develop well-designed visualizations, craft questions that accurately assess intended competencies, and curate multiple-choice answers with one best option [4, 18, 28]. There is an untapped need for a way to generate high-quality, diverse items more efficiently and at scale.

Recent advancements in LLMs have demonstrated their capacity to perform tasks including text generation, code generation, and question answering, with promising levels of success [2, 5, 9, 14]. Creating a visualization item requires similar abilities such as generating code for a visualization and writing questions based on it. Thus, LLMs may be able to produce diverse, high-quality visualization items at scale.

- *Yuan Cui, Lily W. Ge, Fumeng Yang, and Matthew Kay are with Northwestern University. E-mail: charlescui@u.northwestern.edu, {wanqian.ge, fy, mjskay}@northwestern.edu.*
- *Yiren Ding and Lane Harrison are with Worcester Polytechnic Institute. E-mail: {yding5, ltharrison}@wpi.edu.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

To investigate the promises and pitfalls of LLM-based generation of multiple-choice visualization items, we:

1. **Develop the VILA (Visualization Items Generated by Large LAnguage Models) pipeline**, an LLM-based pipeline for generating large, diverse banks of multiple-choice visualization items. Through an iterative design process, we devise a three-stage pipeline: ⚙Stage I: Generator generates a data visualization and its associated dataset; 🖊Stage II: Composer composes the question, options, and correct answer of the multiple-choice item; and 🔎Stage III: Checker checks the generated item and regenerates a new one if mistakes are detected.
2. **Use the VILA pipeline to construct the VILA bank**, a bank of 1,103 multiple-choice visualization items covering 9 contexts (underlying topics of data), 12 chart types, and 13 visualization tasks covering all levels of Bloom's taxonomy. We use the VILA pipeline to generate 1,404 candidate items and then evaluate them based on our rulebook with 11 visualization experts' input, yielding the VILA bank.
3. **Identify promises and pitfalls of the VILA pipeline** based on the evaluation of the VILA bank. We find the VILA pipeline can reliably generate items with the most common chart types (e.g., *line chart*, *bar chart*, and *area chart*) and items with tasks on the *Knowledge*, *Comprehension*, and *Evaluation* levels of Bloom's taxonomy. We also identify six classes of errors that the VILA pipeline made, such as *Disregards Perceptual Limits* and *Information Lost by Aggregation*, highlighting the current limitations of this approach.
4. **Use VILA to create a new visualization literacy test, VILA-VLAT**, which measures the ability to read and interpret visualizations using the same tasks and chart types as the existing VLAT (Visualization Literacy Assessment Test) [28]. Results of an online study comparing VILA-VLAT and VLAT show moderate-to-high convergent validity ($R = 0.70$), demonstrating the potential of our approach.

Our work highlights the need to precisely understand the promises and pitfalls of LLMs for visualization item generation and to integrate human oversight into their use. We provide recommendations on how to use the VILA pipeline and discuss application areas that can benefit from our work. Our work just scratches the surface of the potential for LLMs, used appropriately, to help generate diverse, high-quality visualization items at scale, addressing a persistent barrier that many research and education applications currently face.

2 Background

2.1 Multiple-Choice Visualization Items

Visualization items are used in the field of information visualization to measure skills and abilities. In particular, multiple-choice visualization items are prevalent in the development of visualization literacy assessments [4, 18, 28]. Boy et al. [4] developed a set of tests to assess people's ability to interpret line charts, bar charts, and scatterplots. Following this work, Lee et al. [28] developed a Visualization Literacy Assessment Test (VLAT) that contains 53 multiple-choice items across 12 chart types. Ge et al. [18] expanded prior definitions of visualization literacy to incorporate the ability to detect visualization misinformation, creating the Critical Thinking Assessment for Literacy in Visualizations (CALVI), which has a bank of 45 multiple-choice items. Outside of assessment development, Huynh et al. [20] used multiple-choice items in their educational role-playing games to support visualization literacy education in young children.

Multiple-choice visualization items are also used in a wide range of experiments outside of visualization literacy. For example, Tandon et al. [41] used multiple-choice items to study the effects of domain and spatial visualization ability on people's performance on information visualization tasks; Chen et al. [8] designed multiple-choice items to compare different techniques of visualizing cyclical data; and Lee-Robbins et al. [30] used multiple-choice items in a large-scale experiment to study how specification formats impact visualization design. Moreover, researchers who develop VQA systems use visualization items to train and test computer vision models that answer questions based on the images of data visualizations [23, 31, 32].

2.2 Automatic Item Generation

Traditional item development processes are highly costly and laborious due to their manual nature. They usually involve subject-matter experts writing and reviewing items. Thus, researchers in educational and psychological measurement have sought ways to reduce the amount of human labor required in this process. This field of study is *automatic item generation* (AIG) [27].

AIG generally consists of two steps: first, item developers create item models or templates that characterize elements in the items that are subject to change. Second, different values are inserted into these templates to generate new but similar items [19]. In the area of achievement tests, researchers have used AIG to develop items for mathematics, science, and medical surgery licensing [16]. This approach has also been used by VQA researchers to create datasets for training and testing purposes: Kafle et al. [23], for example, used fixed templates to generate bar chart items. Methani et al. [32] recruited crowdsourced workers to generate questions for given visualizations, then extracted templates from these questions to generate more items, and asked in-house annotators to check and paraphrase the items. Although a large number of items were generated by this approach, many have answers that only come from a small fixed-sized vocabulary (e.g., "Yes" and "No") and do not cover visualization tasks that involve various operations such as comparison and aggregation [31].

Recent advances in large language models have drawn researchers to explore their capabilities and applications in AIG. Nguyen et al. [34] applied Google's T5, a transformer-based encoder-decoder model, to generate items based on reading materials from a graduate-level data science course; 56.7% of their T5-generated items were deemed pedagogically sound by human experts. For the purpose of training VQA models, Masry et al. [31] used T5 to create a set of 23,100 free-response visualization items based on human-written chart summaries of existing charts crawled from online sources. The items are question-answer pairs across three types of visualizations (*line chart*, *bar chart*, *pie chart*) and 4 types of questions (*data retrieval, visual, compositional, both visual & compositional*); the authors evaluated ∼5% of this set manually and found 86.4% to be answerable and correct. Lee et al. [29] explored generating questions in English education through ChatGPT; they invited experts and teachers to validate the LLM generated questions and found that the majority of questions have high validity.

Inspired by this work, we apply LLMs to generating multiple-choice visualization items covering a wider variety of tasks, chart types, and all cognitive levels of Bloom's taxonomy. We also conduct a thorough evaluation of the types of mistakes LLMs currently make when generating these types of items.

2.3 LLMs for Visualization Education

Data visualization researchers have begun to explore the potential of LLMs in visualization education. Chen et al. [10] evaluated GPT's ability to complete homework and quizzes in a college-level data visualization course. They showed that GPT is capable of completing a range of visualization tasks, such as data interpretation and visualization design, at a promising success rate (GPT-4 scored 80% on the course assignments). Chen et al. also identified limitations of GPT, such as its inability to read visualizations accurately and lack of creativity.

In another empirical study, Joshi et al. [22] evaluated LLMs' ability in educating novices about Parallel Coordinate Plots (PCPs) using Bloom's taxonomy. They showed that while LLMs are capable of generating useful and relevant recommendations to teach people about PCPs, some recommendations are completely inappropriate for PCPs and many do not fall under the proper levels of Bloom's taxonomy. Therefore, they recommend human expert oversight of LLM output.

2.4 Effective Prompt Design for LLMs

A *prompt* refers to the instructions that guide an LLM's response. As the quality of the prompt greatly affects the quality of the model's response [40], researchers and LLM developers have created and evaluated various prompting strategies. Wei et al. [43] introduced *chain-of-thought* prompting, which works by decomposing multi-step problems

into intermediate steps. They showed this approach improved LLM performance on a range of reasoning tasks. Building on the *chain-of-thought* design, Press et al. [39] developed the *self-ask* strategy, which requires the LLM to state a series of follow-up questions and answer them. LLM developers such as OpenAI have also published recommendations regarding the design of prompts [36]. All of these prompt designs share two common core principles—(1) specify the steps needed to fulfill the user's request and (2) separate complex tasks into simpler subtasks—which guided us in the design of our prompts.

3 Item Generation

Our goal is to create a method capable of generating large, diverse banks of multiple-choice items that span a wide range of chart types, visualization tasks, and contexts. We first compiled comprehensive lists of these characteristics, which formed our item design space. Next, we used an iterative design process to create a three-stage LLM-based item generation pipeline, the VILA pipeline. We then used it and the item design space to create a candidate bank of 1,404 items.

3.1 Design Space of Items

Item Definition Figure 2 is an example multiple-choice visualization item. On the left side is a bar chart on energy production by source, which is the **visualization component** of the item. On the right side of the chart is the **textual component**: the **question stem** asks a viewer to identify the source with the highest bar; the question stem is followed by the **options** and the **correct answer**.

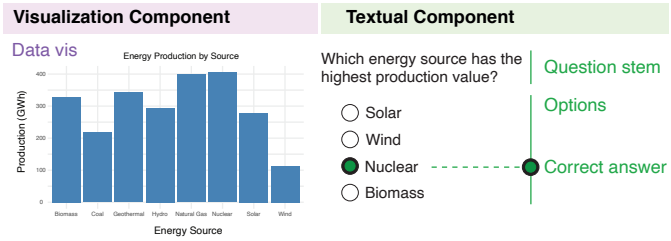


Fig. 2: An example multiple-choice visualization item with its components labeled.

The visualization component can be described by its **chart type** and **context**, where context is the underlying topic of the data. For example, the chart type in Fig. 2 is *bar chart* and the context is *energy and environment*. In the textual component of Fig. 2, the question stem asks the viewer to accomplish the **visualization task**, *locate value*, by choosing the correct answer from the options. The combination of **(context, chart type, visualization task)** characterizes a multiple-choice visualization item, and forms the foundation of our item design space. Below, we describe how we created a diverse and representative list for each of these three dimensions.

Context To compile a list of contexts where data is prevalent in the real world, we adapted the topics from Our World in Data [1], modified the names of the topics for brevity, and created a list of 9 contexts for our design space: *economy*, *education*, *energy & environment*, *food & agriculture*, *health*, *innovation*, *politics*, *population & demography*, and *weather & climate*.

Chart Type We selected the 12 chart types used in VLAT [28]: *area chart*, *bar chart*, *bubble chart*, *choropleth map*, *histogram*, *line chart*, *pie chart*, *scatterplot*, *stacked area chart*, *stacked bar chart*, *tree map*, and *100% stacked bar chart*.

Visualization Task To compile a diverse range of visualization tasks, we adapted a visualization task taxonomy from Burns et al. [6], where they categorized visualization tasks into Bloom's taxonomy.

Considering our need to create multiple-choice items, we revised the tasks in this taxonomy using the criterion of *can a human visualization expert create a multiple-choice item of this task with reasonable quality?*. We renamed some tasks to improve clarity (e.g., *calculate the difference between two points* → *estimate the difference between two values of the same type*), added additional common tasks from another taxonomy [3] that are sufficiently different from existing ones (e.g., *identify range*), and removed tasks that are not suitable for the multiple-choice format (e.g., *translate the data in a chart to a table*). We combined *describe a trend* and *describe the relationship between two variables* into *describe trend or correlation* because they are similar; we also expanded *judge which design is more appropriate* into *judge which task this visualization design best supports* and *judge which visualization design is more appropriate for a task* because they are sufficiently different. The resulting taxonomy below contains 13 visualization tasks across Bloom's taxonomy's 6 cognitive levels:

1. **Knowledge**: *identify labels of scales*, *identify range*, *locate value*, *make comparisons*, *retrieve value*.
2. **Comprehension**: *describe the topic of the visualization*.
3. **Application**: *estimate average*, *estimate difference*, *estimate ratio*.
4. **Analysis**: *describe trend or correlation*.
5. **Synthesis**: *describe the characteristics of an alternative chart type*.
6. **Evaluation**: *judge which task this visualization design best supports*, *judge which visualization design is more appropriate for a task*.

Details of our revision can be found in the *Revisions of Burns' Taxonomy* in supplemental materials.

While some visualization tasks may not always be appropriate for certain chart types, we examined all combinations of chart type (12) and visualization task (13) and confirmed that it was possible to design a meaningful, quality item for every combination ($12 \times 13 = 156$). Therefore, our design space contains all 9 contexts $\times 12$ chart types $\times 13$ visualization tasks = 1,404 possible combinations.

3.2 Design Process for the VILA Pipeline

We took an iterative approach to design the VILA pipeline and to engineer the prompts. We made incremental changes to the prompts, used the current pipeline to generate a small number of items, evaluated the results, and iterated until the results could not be significantly improved. None of the test items generated in this phase were included in the final bank. Three important junctures led to significant changes to the pipeline design, and we describe them below.

Separating Visual and Textual Components Generation We began by using a single-stage solution, prompting the LLM to write code that generates a synthetic dataset, a visualization, and the textual component of an item. The code would use random number generators to create a dataset, and while it also used seeds to ensure reproducibility, the LLM cannot execute the code itself to create the actual dataset before making the textual component of the item. As a result, the LLM-selected correct answer in the item was often wrong due to the absence of the actual dataset. Therefore, we separated item generation into two stages: in ⚙ Stage I: Generator, we prompt the LLM to generate code that creates a synthetic dataset and a visualization based on it. We execute the code to create the dataset and the data visualization locally. Then, in 🖊 Stage II: Composer, in a new request to the LLM, we provide the dataset and prompt it to compose the textual component of the item.

Switching from Python to R We began by prompting the LLM to write code for the visualizations using Python and the `matplotlib` and `seaborn` libraries. We observed that, despite our instructions in the prompt, the resulting visualizations often violated basic design principles such as not overcrowding axes with labels. Therefore, we experimented with using R and the `ggplot2` library. We observed a noticeable improvement in aesthetics and legibility and kept this change.

Adding Automatic Self-Inspection Prior work [33] shows that LLMs have the potential to diagnose and fix their own mistakes, so we developed a third stage that reuses the LLM to inspect the items. We found preliminary evidence supporting this claim through our iterative design process, so we created 🔎 Stage III: Checker, where we pass the item generated by ⚙ Stage I and 🖊 Stage II to the LLM, prompting it to check the item and regenerate the item if mistakes are found.

3.3 Overview of the VILA Pipeline

Our iterative design process yielded the three-stage VILA pipeline: ⚙ Stage I: Generator creates a dataset and a visualization based on the given context and chart type; ✎ Stage II: Composer creates the textual component of the item based on the output from the previous stage and the given visualization task; and 🔍 Stage III: Checker inspects the item and re-generates the textual component if mistakes are detected. The prompt summary for each stage is shown in Fig. 3.[1] Each run of a stage involves sending the LLM the prompt with specific inputs and obtaining one respective response (outputs); to obtain multiple responses based on different inputs, a stage needs to be run multiple times and runs of a stage are independent. The three stages are also independent in that a stage does not have access to the prompt of the previous stage. This design compartmentalizes the item generation process and enables users to customize it based on their needs. For instance, if an item developer wishes to use or build their own specific datasets and data visualizations, they could modify or skip ⚙ Stage I and start with ✎ Stage II. We demonstrate this type of customization in Sec. 5.

We chose GPT-4 (gpt-4-1106-preview), a state-of-the-art LLM for generating code and answering exam questions [2]. This GPT-4 version also supports a seed parameter that improves the reproducibility of results [37]. We set the temperature parameter to be 0.2 in all three stages to achieve consistent and coherent results [35]. Details of our model specifications are in supplemental materials.

context | chart type

3.4 Stage I: Dataset and Visual Component Generator ⚙

dataset | vis code

In this stage, we provide a context and a chart type of the visualization as input to the LLM. In each run, we prompt it to generate the R code for creating a synthetic dataset in the given context and the vis code for rendering the given chart type from the dataset. Figure 3.A contains the prompt summary of ⚙ Stage I. The prompt has two parts: (1) a generic part applied every time and (2) a chart-type specific part for each chart type (line A19). The content of our prompt contains requirements on the output (dataset and visualization), which can be summarized into the following categories: consistency (lines A1–A2), compatibility (lines A9 & A10), completeness (lines A14 & A17), legibility (lines A15–A16), and diversity (line A19).

For more complex chart types such as *choropleth map* and *tree map*, prompting the LLM to use specific libraries (choroplethr and treemapify) yielded code with satisfying outputs. Also, certain chart types require prompting the LLM to use specific ggplot2 commands to ensure correct[illegible] *% stacked bar chart* requires prompting the LLM [illegible] axis [illegible] on the x-axis, it was often presented alphabetically de[illegible] instruction requiring temporal values to be displayed chronologically, therefore, we prompted the LLM to avoid using *month* in the dataset.

vis task | dataset | vis code

3.5 Stage II: Textual Component Composer ✎

question stem | options | correct answer

In this stage, we provide three inputs to the LLM: (1) the dataset from ⚙ Stage I, (2) the vis code from ⚙ Stage I, and (3) a vis task from our list of 13 tasks in the revised taxonomy. In each run, we prompt the LLM to output a question stem, its options, and the correct answer for a single item (see Fig. 3.B).

Similar to ⚙ Stage I, the prompt in ✎ Stage II consists of a generic component applied to all items and a task-specific component (line B6). The content of the prompt contains the explanation of the visualization task (line B6) and instructions for how to generate the correct and incorrect options (lines B7–B20).

[1]The full prompts for the three stages are in supplemental materials.

dataset | vis code

vis task | dataset | vis code

The prompting strategy of decomposing a multi-step problem into sub-[illegible] and [illegible]ere creating the textual component can be divided into a series of specific arithmetic operations on how to find the correct answer and how to create the incorrect options.

question stem | options | correct answer | vis task | dataset | vis code

3.6 Stage III: Item Checker and Regenerator 🔍

question stem | options | correct answer

In this stage, we provide (1) the dataset and vis code from ⚙ Stage I, (2) the vis task, and the (3) the textual component (question stem, options, correct answer) from ✎ Stage II. In each run, we prompt the LLM to output a new textual component (question stem, options, correct answer) for a single item if a mistake is detected and otherwise the same textual component from the input.

Figure 3.C contains the prompt summary of 🔍 Stage III: Checker. The prompt again consists of a generic part that is applied to all items and a task-specific part (line C9). The content of the prompt contains two inspection instructions that prompt the LLM to check whether the correct answer is indeed correct (line C13) and whether it is the only correct answer (line C14). It also contains a request to regenerate an item when a mistake is detected (line C20). We present an analysis of the LLM's self-inspection performance in Sec. 4.6.

3.7 Creating the Candidate Bank of Items

We used the VILA pipeline to create a candidate bank of items. We ran ⚙ Stage I 108 times (9 contexts × 12 chart types) to create 108 data visualizations (vis code and datasets). For each of these 108 outputs, we ran ✎ Stage II 13 separate times to create the textual components for 1,404 items. These items were checked by 🔍 Stage III, yielding 1,404 LLM-inspected candidate items that span all combinations of context, chart type, and visualization task. Figure 4 shows 6 example items in the candidate bank across 6 cognitive levels of Bloom's taxonomy.

4 Item Evaluation

We examined the quality of all candidate items. First, we developed a set of evaluation questions to guide the process. We then conducted a two-phase evaluation with these questions. In Phase I, we created a rulebook for answering the evaluation questions based on 11 visualization experts' input. In Phase II, two author-evaluators followed the rulebook and the questions to evaluate the entire candidate bank. Our evaluation resulted in a final bank of 1,103 items and a set of limitations of the LLM for generating multiple-choice visualization items.

4.1 Evaluation Questions

Four authors discussed the most important aspects of the quality of our items. We iterated on the question wording and arrived at four:

1. **Relevance**: *How relevant is this item to assess peoples ability to carry out the visualization task?* This is rated on a scale from 1 (*Not relevant*) to 4 (*Highly relevant*).
2. **Clarity**: *How clear are the question statement and the options?* This is also rated on a scale from 1 (*Not clear*) to 4 (*Highly clear*).
3. **Answerable**: *Is one of the options a correct or approximately correct answer to the question?* This is rated with *Yes* and *No* options.
4. **Correctness**: *Is the answer marked as correct indeed the correct answer?* This is rated with three options: *Yes*, *No*, and *More than one option could be considered correct*.

Relevance is connected to content validity [38], which evaluates how well an item measures what it aims to measure; *Clarity* ensures that readers can understand the item clearly; *Answerable* and *Correctness* are necessary measures that ensure the correct answer can be identified by a reader and it is the best option among available options.

(A) Stage I: Generator prompt

```
You generate datasets and make data visualizations based on the datasets and the user's specific requirements.
You use R programming language and, unless specified, only use libraries including "ggplot2",
"tidyverse", and "dplyr" to generate the datasets and data visualizations.
# instructions for output formats omitted for space, see supplements
I will give you a context and a chart type of data visualization.
I want you to use R programming language to make a synthetic dataset in the given
context and a data visualization of the specified chart type based on the synthetic dataset.
Here is the context: context
Here is the type of data visualization: chart type
The values in the synthetic datasets must have meaningful names in the given context.
All values in the dataset must be whole numbers.
Make sure to generate a dataset that is well suited for the chart type.
Next, I will give you additional requirements for the visualization and the dataset.
The data visualization must be well-designed with the following requirements:
  (1) There should be a title for the visualization. Display the title at the top.
      Do not overlay the title on other components of the visualization.
  (2) Do not overcrowd color legends or axes with labels.
  (3) Leave enough margins around the visualization so that all labels are displayed fully.
  (4) All data points in the dataset must be displayed on the visualization.
  (5) Do not use month as a type of value in the dataset.
For chart type , chart specific requirements
# instructions for output formats omitted for space, see supplements
```

Consistency — Visualization is stylistically consistent

Inputs

Compatibility — The data suits the chart type

Compatibility

Completeness — Show all info about the dataset

Legibility — Visualization is clear

Completeness

Diversity — Variety of patterns within chart types

(B) Stage II: Composer prompt

```
I will first give you the R code as a string.
Here is the R code: vis code
I will now give you the dataset the R code generated as a string.
Here is the dataset: dataset
Write a multiple-choice item associated with the data visualization and the dataset.
The multiple-choice item must have the following properties:
(1) the task the viewer does is to the task the viewer does is to vis task ; vis task description
(2) the item has only one correct answer.
(3) the viewer should be able to find the correct answer with only the visualization,
    question stem, options, and without access to the dataset.
First, generate a question stem that allows a tester taker to do the
task of " vis task " with the data visualization created by the R code.
Second, generate one correct answer to the question stem.
Use the provided dataset to find the correct answer.
Third, generate three other unambiguously incorrect options for the question stem
and the data visualization.
Do not include justifications for the options in the options themselves.
# instructions for output formats omitted for space, see supplements
Make sure the correct answer is one of the options in part 2 of your response.
In the question stem of part 1:
(1) do not include the phrase "R code" or "R script"; the question stem should not reveal how
the dataset or visualization is generated.
(2) the question stem should be succinct; do not use the phrases "According to the visualization",
"Based on the visualization", or similar phrases.
# instructions for output formats omitted for space, see supplements
```

Inputs

Explanation of Task

Instructions creating question stem and options

(C) Stage III: Checker prompt

```
I will give you a multiple-choice item on a data visualization task which consists of six parts:
    part 1: the question stem
    part 2: options
    part 3: the correct answer
    part 4: the R code that generated the dataset and the data visualization
    part 5: the realized dataset from the R code (no more randomness)
    part 6: the visualization task
Here are the first three parts: question stem options correct answer
Here is the visualization task (part 4): vis task ; vis task description
Here is the R code that generated the dataset and the data visualization: vis code
Here is the dataset: dataset
I want you to check whether there are any mistakes in this item. Specifically,
  (1) check that the correct answer is indeed correct based on the question stem,
      the visualization task, and the dataset. You must find the correct answer yourself first
      and then compare it to the current correct answer.
  (2) check that the correct answer is the only one that is correct among all the options.
      The incorrect options must be clear and unambiguously incorrect.
# instructions for output formats omitted for space, see supplements
If you detect a mistake in an item, use the dataset, the R code for the visualization,
and the visualization task to make a new item so that there are no more mistakes.
# instructions for output formats omitted for space, see supplements
```

Inputs

Check correctness

Check uniqueness

Check and remake

Fig. 3: The prompt summary from the three stages of the VILA pipeline. The variables are replaced by a different value each time the prompt is used. The inputs, requirements, and instructions at each stage are annotated on the right.

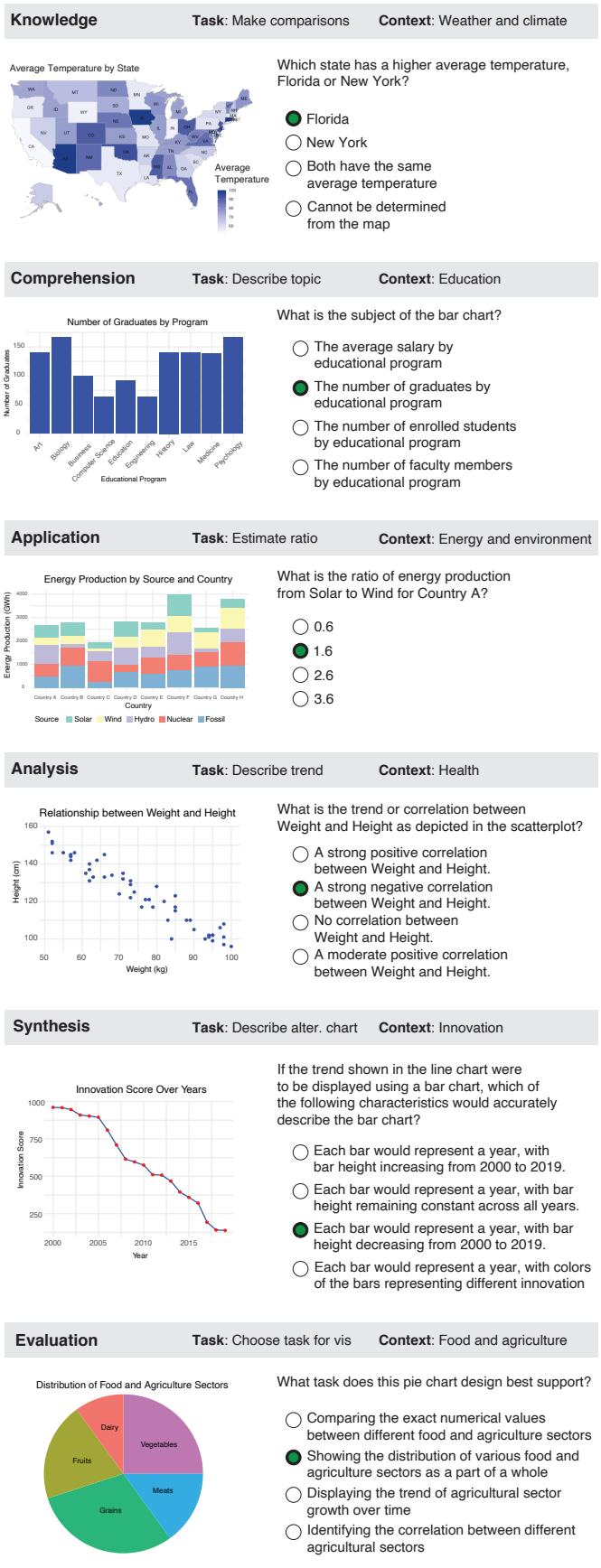


Fig. 4: Example items from the candidate bank across 6 levels of Bloom's taxonomy. The task names are paraphrased for brevity. The highlighted option is the LLM-selected correct answer.

4.2 Phase I: Development of the Rulebook

The goal of this phase is to create a rulebook that will allow two author-evaluators to evaluate the entire candidate bank using criteria grounded in a larger set of experts' opinions. We invited external data visualization experts to evaluate a representative sample of items from the candidate bank, together with the two author-evaluators.

Experts We used a convenience sample of 11 expert participants. They came from six universities and four industry organizations; none were at the same institution(s) as any author. All have obtained a Ph.D. in visualization-related fields.

Representative Sample of the Bank Due to the large size of the bank, we selected a representative sample of 156 items, as a balance between comprehensiveness and practical time constraint, to help create the rulebook. We randomly selected one item from each of the $12 \times 13 = 156$ (chart type, task) combinations, while also ensuring that each of the 9 contexts is represented approximately equally. [2] Each expert received 52 of the 156 items, which included all 13 types of tasks with 4 items for each task. Experts were asked to rate these items based on the 4 evaluation questions and write optional comments. Each item was evaluated by at least 3 experts. The two author-evaluators also evaluated all 156 items independently.

To analyze the evaluations of the experts and author-evaluators for the purpose of creating a rulebook, we focused on cases with disagreements and low ratings. For instance, we examined all items that at least one expert did not rate *Yes* for *Answerable* or *Correctness*. For such items, we looked at the ratio of *Yes* to non-*Yes* ratings, read the experts' comments (if any), and decided on a final rating. We also documented the justifications for our final ratings and how to handle similar situations as part of the rulebook.

Rulebook Construction [3] The rulebook contains both rating suggestions and guidelines for each (chart type, task) combination.

For the *Relevance* and *Clarity* ratings of a (chart type, task) combination, we computed the average ratings of these two categories, respectively, across the experts and the two author-evaluators. We took the floor and the ceiling of the average ratings and used them as the recommended rating options for items with this (chart type, task) combination. The rulebook contains suggested ratings for these two evaluation questions for all (chart type, task) combinations.

The *Answerable* and *Correctness* categories require more scrutiny depending on the specific item, so we do not provide specific suggested ratings for all items with a (chart type, task) combination. Instead, we identify combinations that are prone to certain types of errors to help author-evaluators make decisions. For example, the combination (*choropleth map*, *estimate difference*) tends to have multiple correct answers, because precise values are lost through aggregation on the color scale; the rulebook reminds the author-evaluators to be mindful of such situations and give ratings on a case-by-case basis.

4.3 Phase II: Evaluation of the Candidate Items

In this phase, the two author-evaluators each evaluated half of the candidate items. When they encountered situations not documented in the rulebook, they discussed to reach an agreement and added it to the rulebook. They also documented cases with ratings against the rulebook's recommendations and their justifications for those decisions.

4.4 Final Bank (The VILA Bank)

We used the following **inclusion criteria** to select the final bank of items: items must have a rating of 3 or 4 for both *Relevance* and *Clarity* and a rating of *Yes* for both *Answerable* and *Correctness*.

Figure 5 summarizes the VILA bank. Each cell indicates the number of items in that (chart type, task) combination in the VILA bank. The VILA bank contains a total of 1,103 items (out of 1404) across 154 (chart type, task) combinations, which is ∼**79%** of the candidate bank. All items have *Yes* ratings to both *Answerable* and *Correctness*. The

[2] The representative sample of items and the code used to select them are provided in supplemental materials.

[3] The complete rulebook is available in supplemental materials.

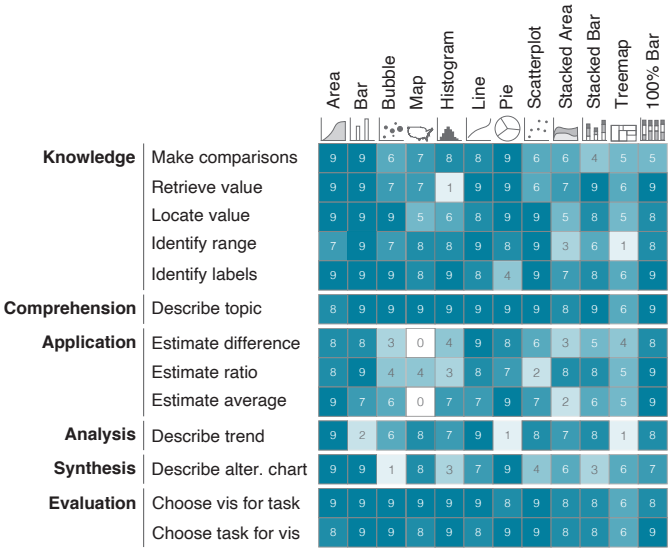


Fig. 5: Summary of the final bank of 1,103 items. Names of chart types and visualization tasks are paraphrased for brevity. Each cell represents a (chart type, task) combination and shows the number of items with that combination in the final bank. The maximum is 9 because there are 9 contexts in our item design space.

average *Relevance* and *Clarity* ratings for the bank are 3.7 and 3.8, respectively. The VILA bank is available in supplemental materials.

4.5 Promises and Pitfalls

Through this evaluation, we also identified the strengths of the VILA pipeline and classified the errors that it is prone to make.

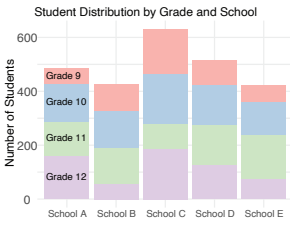
4.5.1 "Promises" of the VILA Pipeline

From Fig. 5, we observe that the VILA pipeline is capable of reliably producing high-quality items with many chart types and tasks. In particular, it performs well on generating items with the most common and simplest chart types, such as *line chart*, *bar chart*, *pie chart*, and *area chart*. It also performs well on creating items with tasks in the *Knowledge* and *Comprehension* cognitive levels. Moreover, we note that the VILA pipeline can reliably generate items with tasks on the highest cognitive level in Bloom's taxonomy, *Evaluation*, which include the tasks *judge which visualization design is more appropriate for a task* (*choose vis for task*) and *judge which task this visualization design best supports* (*choose task for vis*).

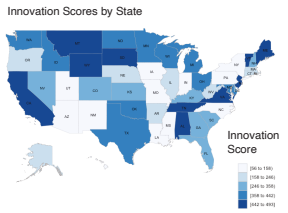
4.5.2 "Pitfalls" of the VILA Pipeline

To better understand the limitations of the VILA pipeline, guide others in its use, and help researchers improve upon this method, we classified the errors the pipeline produced into six major categories.[4]

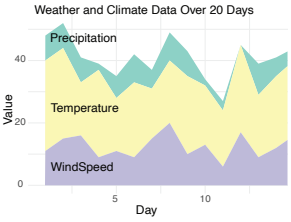
I. Disregards Perceptual Limits The item disregards humans' perceptual limits and is too difficult to answer. For instance, in the stacked bar chart on the right, it is very hard to visually estimate the average number of students in Grade 9 across the 5 schools because the red segments are not aligned. If the incorrect options are too close to the correct one, this item can be impossible to answer.



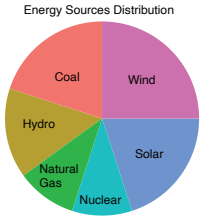
II. Information Lost by Aggregation Data aggregation in the visualization obscures the correct answer. For instance, identifying the state with the highest innovation score (*locate value*) on the choropleth map on the right is impossible because there are multiple states with the same darkest color due to binning in the color mapping.



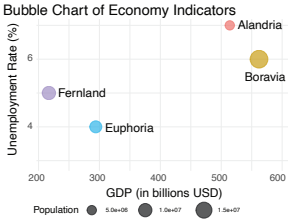
III. Inappropriate Visual-Data Correspondence The visualizations do not correctly represent relationships in the data, violating the principle of visual-data correspondence [25]. For instance, the stacked area chart on the right stacks wind speed, temperature, and precipitation values together, but the sum of these values on the *y*-axis is not meaningful.



IV. Task Not Suitable for Chart The design of the visualization does not support the item's visualization task. For instance, the pie chart on the right does not contain temporal or geographic variables, so the task *describe trend or correlation* is not suitable for this chart.



V. Unclear Phrasing of Question The question stem is not phrased clearly enough for the reader to identify the data needed to answer the question. For instance, when asked "which country has a larger population size, the one with higher GDP or the one with lower unemployment rate?" for the bubble chart on the right, it is unclear which two countries the question is describing, so the item is not answerable.



VI. Incorrect Answer The options do not contain a correct answer, or the actual correct answer is not the LLM-selected correct answer. For instance, the histogram on the right has a symmetric distribution, but the LLM-selected correct answer to the question "what pattern does the histogram show?" is "right-skewed". This may be related to the concept of **hallucination** in language models [21, 24], where the LLM-generated content is nonsensical or unfaithful to the provided source content [21].

While some errors may be fixable by better prompt engineering or different pipeline designs, these classes of errors reveal the limitations of our LLM-based approach. Human experts possess nuanced, often implicit knowledge of how to design good visualizations and good items that the LLM does not currently have. It is therefore crucial to elicit experts' implicit knowledge and incorporate them into the design process. To this end, we discuss the importance of human oversight in Sec. 6.2 and the potential of human-LLM collaboration in Sec. 6.5.

4.6 Performance of Stage III

To investigate the performance of 🔑Stage III: Checker, we also evaluated the versions of the 1,404 items **before** they were inspected by 🔑Stage III: Checker. Combined with the evaluation of the candidate bank from 🔑Stage III, this allows us to examine 🔑Stage III: Checker's capability of

[4]The classification of all errors is available in supplemental materials.

detecting and fixing mistakes. We defined a mistake as an item that would not pass the inclusion criteria in Sec. 4.4.

Figure 6 summarizes the performance of Stage III. The accuracy of the pipeline with Stage III: Checker is $\frac{958+89+56}{1404} = 0.79$. If we were to use the pipeline without Stage III: Checker, the accuracy would be $\frac{958+100}{1404} = 0.75$; if we were to use Stage III: Checker and just remove all items it flags as mistakes without fixing them, the accuracy would be $\frac{958}{958+220} = 0.81$. This demonstrates that Stage III is capable of improving the VILA pipeline's performance by fixing items that it deemed to have mistakes, but discarding all such items without fixing is also a reasonable approach with high accuracy.

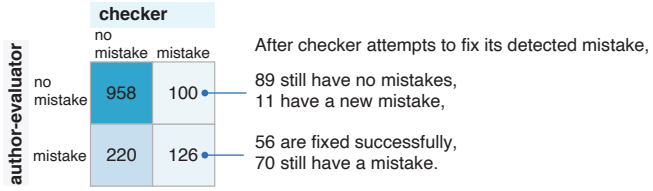


Fig. 6: Confusion matrix of the performance of Stage III: Checker.

5 Application Demonstration: VILA-VLAT

In this section, we provide a demonstration of applying the VILA pipeline and the VILA bank to measure visualization literacy. Specifically, by using the final bank and customizing the VILA pipeline, we created and validated a visualization literacy test, VILA-VLAT, that covers the same (chart type, task) combinations as VLAT [28].

5.1 Construction of VILA-VLAT

To ensure VILA-VLAT is as comprehensive as VLAT, we used VLAT's structure to create VILA-VLAT. VLAT contains 53 items that span 12 chart types with 8 tasks [28]. Each VLAT item is a unique (chart type, task) combination, and each test taker is expected to take all 53 items.

Reclassification of VLAT Tasks The VILA bank has the same 12 chart types as VLAT, but the tasks are not identical. Therefore, we reclassified the task in each VLAT item using our revised task taxonomy. We did so by examining the question stem and the options of each VLAT item to see what task the viewer is asked to accomplish and then looking through our revised task taxonomy to find the most fitting task. This reclassification yielded 46 unique (chart type, task) combinations under our task taxonomy.

Pipeline Customization for Special Items There are 5 VLAT items whose tasks did not fit in our taxonomy: *identify anomalies* for scatterplot and bubble chart, *identify clusters* for scatterplot and bubble chart, and *identify the hierarchical structure* for tree map. All three tasks are specific to certain chart types, and some require special considerations on the data visualization (e.g., the task *identify anomalies* is only suitable for a scatterplot that contains an outlier).

The VILA pipeline is designed to support using customized visualizations and datasets by modifying Stage I. Stage II can also be modified to include additional visualization tasks. To construct these 5 items for VILA-VLAT, we customized the VILA pipeline: for *identify anomalies* and *identify clusters*, we added four chart types (*scatterplot with outlier*, *scatterplot with cluster*, *bubble chart with outlier*, and *bubble chart with cluster*) in Stage I to create 4 chart types $\times$ 9 contexts $=$ 36 visualizations. We checked these visualizations and removed 14 that do not contain a visible outlier or cluster. Next, we added the two tasks and their descriptions in Stage II's prompt, and the 22 remaining visualizations were then used by Stage II and Stage III to create and check the items. Then, we evaluated all 22 items using the 4 evaluation questions and decided to keep all, because they passed the inclusion criteria in Sec. 4.4. For *identify the hierarchical structure*, we used the 6 treemap visualizations and datasets from our final bank. We added the task and its description to the prompt in Stage II and used Stage II and Stage III to create the textual components. All 6 items passed the inclusion criteria.

VILA-VLAT contains 51 items from 51 (chart type, task) unique combinations. For each test taker, one item from each (chart type, task) combination is randomly drawn from our final bank and the 28 customized items. VILA-VLAT uses a total of 371 items: 343 items from the VILA bank and 28 customized items.

5.2 Validity of VILA-VLAT

The inclusion criteria in Sec. 4.4 ensure that all items in VILA-VLAT have a *Relevance* rating of 3 or 4, which is evidence for high **content validity** [12,38]. We also evaluate the **convergent validity** [11] of VILA-VLAT by computing the correlation between test-takers' performance on VILA-VLAT and VLAT. Convergent validity is a measure that establishes the equivalence of two tests through strong correlation.[5]

Participants We recruited 120 participants via **Prolific**.[6] We selected our sample size based on similar experiments on convergent validity from Cui et al. [13], anticipating that about 100 of the 120 participants would complete the entire study. All participants fall under the following criteria: speak fluent English, have normal or corrected-to-normal vision, do not have colorblindness, are between 18 and 65 years old, and are U.S. residents. In addition, **Prolific** participants from similar studies that the authors conducted in the past were excluded.

Procedure Each participant was asked to take both VLAT and VILA-VLAT,[7] with a minimum 5-hour break between sessions to avoid fatigue. We counterbalanced to avoid ordering effects: half of the participants (Group I, 60 people) took VLAT first, and the other half (Group II, 60 people) took VILA-VLAT first. In each session, we presented participants with the consent form and information page describing the structure of the study. We informed them that they needed to select an answer for the current item before proceeding to the next one, and once they moved on, they could not return to previous ones. Participants who failed two or more attention check questions (out of three) were excluded. In the second session, 44 and 45 returned in Groups I and II respectively, for a total of 89 participants. Two participants failed the attention check, leaving 87 participants who completed both tests, and we used their data to compute the correlation for convergent validity.

Results **The correlation coefficient between VLAT and VILA-VLAT was 0.70 with 95% CI of [0.57, 0.79]**. Although there is no gold standard, a coefficient below 0.50 should be avoided and above 0.70 is recommended [7], so VILA-VLAT has **moderate to high convergent validity**. In addition, the average correctness rates for VLAT and VILA-VLAT are 0.76 and 0.77, respectively. Together with the convergent validity evidence, this shows that VILA-VLAT is a valid visualization literacy test that measures similar skills as VLAT and has similar difficulty.[8]

6 Discussion

6.1 Hypothetical Cost Comparison

Is using the VILA pipeline actually worth it? It is hard to provide a universal answer because it depends on the specific needs of different users. Based on the authors' experience on creating visualization items, we present a rough, hypothetical cost comparison for generating the VILA bank to help prospective users reason about the cost of our method and offer people a template to evaluate their specific situations.

Financial Cost The financial cost of creating items consists of two variables: the number of items needed and the unit cost of an item. The cost of creating the VILA bank of 1,103 items using our pipeline and OpenAI's GPT-4 model is $\sim$\$50, which means the unit cost of an item is $\sim$\$0.05. Two author-evaluators, who are Ph.D. students, evaluated all items in the candidate bank, and it took roughly 8 hours per person (a total of 16 hours). A typical hourly rate of a Ph.D. student in the U.S. is \$20; it would cost \$0.23 to evaluate one item. Thus, the total cost per item is $\$0.05 + \$0.23 = \$0.28$.

[5] Our analysis code is preregistered on OSF (see link in abstract) and can be found in supplemental materials.

[6] This study received institutional IRB approval.

[7] Both are implemented using reVISit [15], an open-source software toolkit for creating empirical visualization studies.

[8] Due to an implementation error, two items not in the final bank were administered as part of VILA-VLAT in this study along with the other 371 items. We removed those two items from the responses of participants who encountered them. Details are in the analysis code file of supplemental materials.

An alternative for creating quality items, for instance, is to hire a Ph.D. student in data visualization to make them from scratch. Suppose they can create 4 context-specific visualizations in an hour; it will take them 27 hours to create 108 visualizations (9 contexts $\times$ 12 chart types). Suppose they can write 10 textual components (question, incorrect options, correct option) of a visualization in an hour; it would take them 140 hours to create 1,404 textual components. The unit price of an item in this approach is $\frac{(27+140)\times\$20}{1{,}404} = \$2.38$. While there are other alternatives and the specific costs of humans can vary, the VILA pipeline could be a better financial option than manually creating everything from scratch.

Time Cost Without parallelization, it took $\sim$40 seconds for the VILA pipeline to create an item, so the total time of creating the candidate bank is 16 hours. Together with the 16 hours of evaluation, the total time of creating the VILA bank of 1,103 items is 32 hours, which translates to 34 items per hour. Following our previous assumption, a Ph.D. student can create $\frac{1{,}404}{27+140} = 8.4$ items per hour, so our method could be advantageous in time compared to an entirely manual approach.

6.2 Recommendations for Using the VILA Pipeline

We recommend prospective users of the VILA pipeline first enumerate what chart types, visualization tasks, and contexts their desired items should cover. Users can consult Fig. 5 and the analysis in Sec. 4.5 to assess whether our pipeline can reliably generate the specific (chart type, task) combinations they require. We advise against using our pipeline to generate combinations that are prone to error. If our item design space does not contain certain chart types, tasks, or contexts that users need, we recommend incorporating these new elements by customizing ⚙Stage I and 🖊Stage II as we did for constructing VILA-VLAT in Sec. 5.1. For (chart type, task) combinations that the VILA pipeline can reliably generate (the dark cells numbered 8 or 9 in Fig. 5), human oversight may not be necessary depending on the users' requirements and constraints. For (chart type, task) combinations that are less reliable (cells numbered 5, 6, or 7 in Fig. 5), we recommend users manually evaluate them using the four evaluation questions in Sec. 4.1 and look out for the errors in Sec. 4.5.2. It is also important to consider whether use of VILA with any necessary human supervision will have a cost or time advantage for a particular application (as in Sec. 6.1).

6.3 Potential Application Areas

Visualization Literacy Visualization literacy researchers need large banks of items to reduce item overuse in repeated studies. They also need diverse items to assess people's ability to complete a wide range of tasks across different cognitive levels of Bloom's taxonomy. Adaptive testing of visualization literacy [13], where the next item for a test taker is adaptively chosen based on a person's performance on previous items, can measure people's abilities more accurately and efficiently, but also typically requires large item banks. The VILA pipeline could be used to generate large, diverse banks of visualization literacy items needed for these applications.

Visualization Courses Teachers of data visualization and statistics classes need evaluative and practice materials to assess students' performance and help them improve. The VILA pipeline offers them an efficient way to generate a wide range of items at scale to reduce manual labor. Having a large item bank also reduces the risk of leaking test materials due to recycling content. The diversity of items the VILA pipeline can generate can also help teachers customize interventions for students with different needs.

Visualization Question-Answering (VQA) VQA researchers require large amounts of question-answer pairs and visualization images to develop their models. The VILA pipeline offers an alternative method of dataset creation compared to using fixed templates or recruiting crowdsourcing participants. In addition, current VQA datasets mainly consist of items with tasks on the lower end of Bloom's taxonomy. As research in VQA advances, the ability of the VILA pipeline to generate items with tasks across a wide range of cognitive levels will help researchers assess the ability of their systems and improve their models.

6.4 Design Guidelines for Visualization Items

Throughout the refinement of the VILA pipeline and the evaluation of items, we regularly found the need to explicitly write down otherwise-implicit rules about good item construction so that the LLM was able to apply them. When human developers design visualization items, we often rely on our implicit expertise or have a "hunch" of what makes an item good. For example, if a human expert makes an item that asks the reader to estimate the average value of a line chart, they are aware that the incorrect options cannot be too close to the correct one (see *Disregards Perceptual Limits* in Sec. 4.5.2). However, there is no existing set of explicit guidelines on how to effectively create multiple-choice visualization items. This creates a barrier for people who need to make items but do not have sufficient expertise, and it also makes the item-creation process less systematic. We collected the criteria and instructions from our prompts, as well as the classes of errors from our evaluation, and organized them into design guidelines for creating multiple-choice visualization items. These guidelines are available in our supplemental materials.

6.5 Design Lessons and Human-LLM Collaboration

It is difficult and time-consuming for human developers to design diverse, high-quality visualization items at scale. We demonstrate it is possible to partially automate the item-generation process and create large sets of diverse items. While our prompt designs and resulting items are still imperfect, our work reveals that item generation efficiency can be improved while maintaining quality through human evaluation of items (which is less costly than pure human item generation; see Sec. 6.1), and that our pipeline can be customized to handle special cases where the LLM otherwise cannot generate items (see Sec. 5.1).

However, we have only scratched the surface of the potential for item generation using LLMs. We only used one LLM, and it is worth investigating the pipeline's performance on multiple LLMs or on LLMs fine-tuned for this specific application. We could also incorporate large multimodal models (LMMs) with vision capabilities into the pipeline to generate items based on real-world visualizations. These models may also help address errors we identified in our pipeline, such as *Information Lost by Aggregation* and *Disregards Perceptual Limits*. Also, ⚙Stage I and 🖊Stage II currently are separated, making the visualization generation stage unaware of what types of visualization tasks it needs to support, which can lead to errors such as *Task Not Suitable for Chart*.

In addition to improving the model, we could adopt a more collaborative approach to human–LLM teaming. Currently, human evaluation occurs only after the items have been generated. A more collaborative, interactive system might seek human feedback continuously throughout the entire process, showing possible visualizations or datasets that experts could select from or refine, feeding back into further visualization or item generation. Or, experts might design item templates interactively, triggering real-time generation of items that they can quickly see, evaluate, and refine. We believe methods and insights from creativity support tools in HCI [17] and human-AI collaboration [42] could lead to more efficient and reliable item-generation systems.

7 Conclusion

To create large, diverse multiple-choice visualization items at scale, we developed the LLM-based VILA pipeline that consists of ⚙Stage I: Generator, 🖊Stage II: Composer, and 🔎Stage III: Checker. We used the VILA pipeline to create 1,404 candidate items across 9 contexts, 12 chart types, and 13 visualization tasks, covering all levels of Bloom's taxonomy. We developed a rulebook with visualization experts and used it to evaluate the candidate items. The result is the VILA bank of 1,103 items, $\sim$79% of the candidate items. We categorized the errors of the pipeline, identifying important avenues for future work. We also used the VILA bank and pipeline to create a visualization literacy test, VILA-VLAT, and validated it via an online study. As visualization items are widely needed in research and education, VILA can be used to support the efficient creation of evaluative materials under proper human oversight. Our work establishes a foundational step for the continued exploration of LLMs in visualization item generation.

Supplemental Materials

All supplemental materials are available on OSF at `https://osf.io/ysrhq/`. Icons are designed by the authors or under a Flaticon license.

Acknowledgments

We are grateful for members of the MU Collective at Northwestern University for their support and feedback. We extend our gratitude to Eleanor O'Rourke for her early-stage feedback. We thank our expert panel for giving their time to our research. We also thank the online participants for their time and the reviewers for their insightful comments. This work was supported in part by grants from the National Science Foundation (#DGE-2234667, #2213757, #2127309 to the Computing Research Association for the CIFellows Project) and by the Northwestern Institute on Complex Systems.

References

[1] Our World in Data, 2024. `https://ourworldindata.org/`. 3

[2] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*, 2023. 1, 4

[3] R. Amar, J. Eagan, and J. Stasko. Low-level Components of Analytic Activity in Information Visualization. In *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, pp. 111–117, 2005. doi: 10.1109/INFVIS.2005.1532136 3

[4] J. Boy, R. A. Rensink, E. Bertini, and J.-D. Fekete. A Principled Way of Assessing Visualization Literacy. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1963–1972, 2014. doi: 10.1109/TVCG.2014.2346984 1, 2

[5] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language Models Are Few-Shot Learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, article no. 159, 25 pages. Curran Associates Inc., Red Hook, NY, USA, 2020. doi: 10.48550/arXiv.2005.14165 1

[6] A. Burns, C. Xiong, S. Franconeri, A. Cairo, and N. Mahyar. How to evaluate data visualizations across different levels of understanding. In *2020 IEEE Workshop on Evaluation and Beyond - Methodological Approaches to Visualization (BELIV)*, pp. 19–28. IEEE Computer Society, Los Alamitos, CA, USA, Oct 2020. doi: 10.1109/BELIV51497.2020.00010 3

[7] K. D. Carlson and A. O. Herdman. Understanding the Impact of Convergent Validity on Research Results. *Organizational Research Methods*, 15(1):17–32, 2012. doi: 10.1177/1094428110392383 8

[8] K.-T. Chen, T. Dwyer, B. Bach, and K. Marriott. Rotate or Wrap? Interactive Visualisations of Cyclical Data on Cylindrical or Toroidal Topologies. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):727–736, 2022. doi: 10.1109/TVCG.2021.3114693 1, 2

[9] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. Evaluating Large Language Models Trained on Code. *arXiv preprint arXiv:2107.03374*, 2021. 1

[10] Z. Chen, C. Zhang, Q. Wang, J. Troidl, S. Warchol, J. Beyer, N. Gehlenborg, and H. Pfister. Beyond Generating Code: Evaluating GPT on a Data Visualization Course. In *2023 IEEE VIS Workshop on Visualization Education, Literacy, and Activities (EduVis)*, pp. 16–21. IEEE Computer Society, Los Alamitos, CA, USA, Oct 2023. doi: 10.1109/EduVis60792.2023.00009 2

[11] C.-L. Chin and G. Yao. Convergent Validity. In *Encyclopedia of Quality of Life and Well-Being Research*, pp. 1275–1276. Springer, Dordrecht, 2014. doi: 10.1007/978-94-007-0753-5_573 8

[12] R. J. Cohen, W. J. Schneider, and R. Tobin. *Psychological Testing and Assessment*. McGraw Hill LLC, New York, NY, 10th ed., 2022. 8

[13] Y. Cui, L. W. Ge, Y. Ding, F. Yang, L. Harrison, and M. Kay. Adaptive Assessment of Visualization Literacy. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):628–637, 2024. doi: 10.1109/TVCG.2023.3327165 1, 8, 9

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In J. Burstein, C. Doran, and T. Solorio, eds., *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota, June 2019. doi: 10.18653/v1/N19-1423 1

[15] Y. Ding, J. Wilburn, H. Shrestha, A. Ndlovu, K. Gadhave, C. Nobre, A. Lex, and L. Harrison. reVISit: Supporting Scalable Evaluation of Interactive Visualizations. In *2023 IEEE Visualization and Visual Analytics (VIS)*, pp. 31–35. IEEE Computer Society, Los Alamitos, CA, USA, oct 2023. doi: 10.1109/VIS54172.2023.00015 8

[16] S. E. Embretson and N. M. Kingston. Automatic item generation: A more efficient process for developing mathematics achievement items? *Journal of Educational Measurement*, 55(1):112–131, 2018. doi: 10.1111/jedm.12166 2

[17] J. Frich, L. MacDonald Vermeulen, C. Remy, M. M. Biskjaer, and P. Dalsgaard. Mapping the Landscape of Creativity Support Tools in HCI. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, 18 pages, p. 118. Association for Computing Machinery, New York, NY, USA, 2019. doi: 10.1145/3290605.3300619 9

[18] L. W. Ge, Y. Cui, and M. Kay. CALVI: Critical Thinking Assessment for Literacy in Visualizations. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI'23, pp. 1–18. Association for Computing Machinery, New York, NY, USA, 2023. doi: 10.1145/3544548.3581406 1, 2

[19] M. J. Gierl and T. M. Haladyna. *Automatic Item Generation: Theory and Practice*. Routledge, 2012. doi: 10.4324/9780203803912 2

[20] E. Huynh, A. Nyhout, P. Ganea, and F. Chevalier. Designing Narrative-Focused Role-Playing Games for Visualization Literacy in Young Children. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):924–934, 2021. doi: 10.1109/TVCG.2020.3030464 1, 2

[21] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung. Survey of Hallucination in Natural Language Generation. 55(12), article no. 248, 38 pages, Mar 2023. doi: 10.1145/3571730 7

[22] A. Joshi, C. Srinivas, E. E. Firat, and R. S. Laramee. Evaluating the recommendations of llms to teach a visualization technique using bloom's taxonomy. *Electronic Imaging*, 36(1):360–1–360–1, 2024. doi: 10.2352/EI.2024.36.1.VDA-360 2

[23] K. Kafle, B. Price, S. Cohen, and C. Kanan. DVQA: Understanding Data Visualizations via Question Answering. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5648–5656, 2018. doi: 10.1109/CVPR.2018.00592 1, 2

[24] A. T. Kalai and S. S. Vempala. Calibrated Language Models Must Hallucinate. *arXiv preprint arXiv:2311.14648*, 2023. 7

[25] G. Kindlmann and C. Scheidegger. An Algebraic Process for Visualization Design. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2181–2190, 2014. doi: 10.1109/TVCG.2014.2346325 7

[26] D. R. Krathwohl. A Revision of Bloom's Taxonomy: An Overview. *Theory into practice*, 41(4):212–218, 2002. doi: 10.1207/s15430421tip4104_2 1

[27] S. Lane, M. R. Raymond, T. M. Haladyna, et al. *Handbook of Test Development*, vol. 2. Routledge New York, NY, 2016. doi: 10.4324/9780203102961 2

[28] S. Lee, S.-H. Kim, and B. C. Kwon. VLAT: Development of a Visualization Literacy Assessment Test. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):551–560, 2017. doi: 10.1109/TVCG.2016.2598920 1, 2, 3, 8

[29] U. Lee, H. Jung, Y. Jeon, Y. Sohn, W. Hwang, J. Moon, and H. Kim. Few-shot is enough: exploring ChatGPT prompt engineering method for automatic question generation in english education. *Education and Information Technologies*, Oct. 2023. doi: 10.1007/s10639-023-12249-8 2

[30] E. Lee-Robbins, S. He, and E. Adar. Learning Objectives, Insights, and Assessments: How Specification Formats Impact Design. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):676–685, 2022. doi: 10.1109/TVCG.2021.3114811 2

[31] A. Masry, X. L. Do, J. Q. Tan, S. Joty, and E. Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In S. Muresan, P. Nakov, and A. Villavicencio, eds., *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 2263–2279. Association for Computational Linguistics, Dublin, Ireland, May 2022. doi: 10.18653/v1/2022.findings-acl.177 1, 2

[32] N. Methani, P. Ganguly, M. M. Khapra, and P. Kumar. PlotQA: Reasoning over Scientific Plots. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1516–1525. IEEE Computer

Society, Los Alamitos, CA, USA, Mar 2020. doi: 10.1109/WACV45572.2020.9093523 1, 2

[33] N. Miao, Y. W. Teh, and T. Rainforth. SelfCheck: Using LLMs to Zero-Shot Check Their Own Step-by-Step Reasoning. *arXiv preprint arXiv:2308.00436*, 2023. 3

[34] H. A. Nguyen, S. Bhat, S. Moore, N. Bier, and J. Stamper. Towards generalized methods for automatic question generation in educational domains. In *Educating for a New Future: Making Sense of Technology-Enhanced Learning Adoption: 17th European Conference on Technology Enhanced Learning, EC-TEL 2022, Toulouse, France, September 1216, 2022, Proceedings*, 13 pages, p. 272284. Springer-Verlag, Berlin, Heidelberg, 2022. doi: 10.1007/978-3-031-16290-9_20 2

[35] OpenAI. Documentation: How Should I Set The Temperature Parameter. `https://platform.openai.com/docs/guides/text-generation/how-should-i-set-the-temperature-parameter`. 4

[36] OpenAI. Documentation: Prompt Engineering. `https://platform.openai.com/docs/guides/prompt-engineering`. 3, 4

[37] OpenAI. Documentation: Reproducible Outputs. `https://platform.openai.com/docs/guides/text-generation/reproducible-outputs`. 4

[38] D. F. Polit, C. T. Beck, and S. V. Owen. Is the CVI an acceptable indicator of content validity? Appraisal and recommendations. *Research in Nursing & Health*, 30(4):459–467, 2007. doi: 10.1002/nur.20199 4, 8

[39] O. Press, M. Zhang, S. Min, L. Schmidt, N. Smith, and M. Lewis. Measuring and Narrowing the Compositionality Gap in Language Models. In H. Bouamor, J. Pino, and K. Bali, eds., *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711. Association for Computational Linguistics, Singapore, Dec. 2023. doi: 10.18653/v1/2023.findings-emnlp.378 3

[40] L. Reynolds and K. McDonell. Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI EA '21, article no. 314, 7 pages. Association for Computing Machinery, New York, NY, USA, 2021. doi: 10.1145/3411763.3451760 2

[41] S. Tandon, A. Abdul-Rahman, and R. Borgo. Measuring Effects of Spatial Visualization and Domain on Visualization Task Performance: A Comparative Study. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):668–678, 2023. doi: 10.1109/TVCG.2022.3209491 1, 2

[42] D. Wang, E. Churchill, P. Maes, X. Fan, B. Shneiderman, Y. Shi, and Q. Wang. From Human-Human Collaboration to Human-AI Collaboration: Designing AI Systems That Can Work Together with People. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI EA '20, 6 pages, p. 16. Association for Computing Machinery, New York, NY, USA, 2020. doi: 10.1145/3334480.3381069 9

[43] J. Wei, X. Wang, D. Schuurmans, M. Bosma, b. ichter, F. Xia, E. Chi, Q. V. Le, and D. Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds., *Advances in Neural Information Processing Systems*, vol. 35, pp. 24824–24837. Curran Associates, Inc., 2022. doi: 10.48550/arXiv.2201.11903 2, 4