KiwiNotebook: Interactive Computational Notebook Enhanced by Retrieval-Augmented Language Models

MOHAMED HENDY*, ARGHYA SARKAR*, UTKU EGE TULUK, YICHENG JIANG, XIN TENG, and HONGYI WEN, Shanghai Frontiers Science Center of AI and DL, NYU Shanghai, China

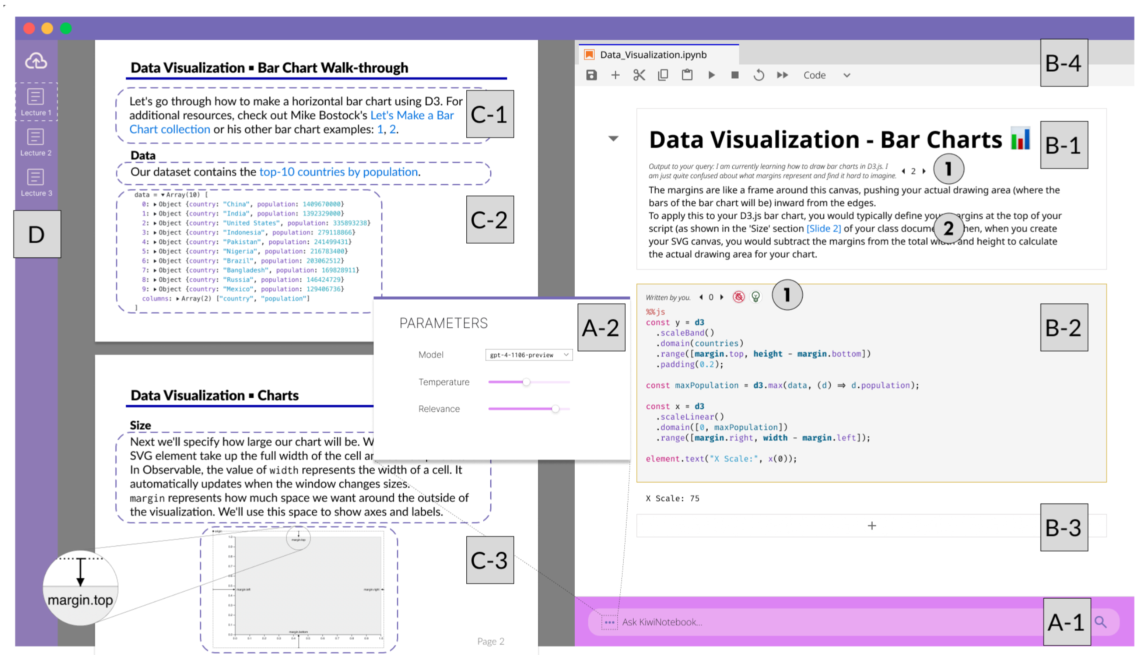

Fig. 1. **KiwiNotebook interface:** In (A-1), the Chat Assistant accepts the student's prompt and response parameters i.e. model, temperature, and relevance (A-2) as input and outputs a response. In part (B), the computational notebook displays the Assistant's responses in the form of a prompt iteration number (B-1-1) and a text response in a markdown cell (B-1-2), which also contains links to class slides. The notebook also parses the code responses in a code cell (B-2). The code cells contain a "Hints" button (B-2-1) that provides a step-by-step, class-informed interactive feedback, and the hint is displayed in new cell B-3. The left side (C) shows the class slides that provide general context in the form of draggable text (C-1), code/ data (C-2), and image blocks (C-3). KiwiNotebook assumes that the context is the entire class slides document if the student asks a question in the conversational bar (A-1) or considers specific context if the student selects any from (C 1-3). On the leftmost side (D), the student could upload/choose different class slides.

Computational notebooks have become integral in Data Science workflows and thus increasingly extended into educational settings for enhanced interaction between students and Data Science materials. While previous studies highlighted the benefits of incorporating notebooks for mutual engagements, the emergence of AI technologies necessitates exploring the potential integration of Large Language Models (LLMs) into computational notebooks to enhance Data Science education. This paper introduces KiwiNotebook, a novel framework that integrates Data Science course slides with Retrieval-Augmented Language Models to seamlessly enhance computational notebooks' interactivity. KiwiNotebook enables students to interact with class slides, run code, and ask clarifying questions through a Chat Assistant interface. The architecture emphasizes real-time student-slide interactions, facilitating a step-by-step teaching-learning approach within the notebook environment. Comparative experiments between KiwiNotebook and the OpenAI

*Both authors contributed equally to this research.

API highlighted KiwiNotebook's personalized responses tailored to class slides and emphasized step-by-step guidance and interactive coding support.

Additional Key Words and Phrases: Computational Notebooks, Language Models, Retrieval-Augmented

1 INTRODUCTION

Computational notebooks have emerged as a cornerstone in the field of Data Science [12], thus their incorporation in Data Science education is ubiquitous. Apart from serving as an environment for teaching code, analyzing graphs, and summarizing experiments in Data Analytics and Artificial Intelligence courses, computational notebooks have been successfully implemented to aid collaboration between instructors and students[4, 11]. Often, these collaborations have led to positive experiences. Cardoso et al. [2] utilized Jupyter notebooks in an Informatics Engineering class and noticed students' increased performance and engagement. Highlighting the versatility and ease, Vallejo et al. [14] designed Google Colab notebooks for teaching programming-based classes. However, such collaborations need not just be limited to those amongst instructors and students.

Recently with the advent of large language models (LLMs) and AI chatbots, it has been conceived that such LLM-student collaborations are not only feasible but also very helpful in Data Science Education. Chatbot-human collaboration enhances task efficiency and information quality; in education, AI chatbots boost student performance, motivation, self-efficacy, and alleviate anxiety [17]. For instance, Wu et al. [17] observed that AI chatbots had a large effect on students' learning outcomes. Given the complexity of decision-making in visualization tasks, the AI chatbot could make short interventions to have a large effect on student outcomes. Despite their emergence, only 11.11 percent of educational chatbots follow a user-driven approach [8], which is critical in case of educational notebooks. The interest in adopting LLMs in education is increasing [6]; however, integrating them into computational notebooks and assessing their implications on education has not been fully explored yet, and there seems to be lots of room for improvement.

Attempts to pair computational notebooks with large language models have been limited to generalized contexts. For example, Jupyterlab has added a conversational AI chatbot assistant that could provide coding examples and read local files to power its notebook [1]. This chatbot is, however, not relevant to Data Science education due to a) not being multimodal i.e. accepting media outside of text and b) providing direct, non-guiding responses. Since Jupyterlab chatbot can't parse images, it is substantially limited. Most class slides are in graphical format and a chatbot which is non-multimodal cannot respond to questions related to intricate graphical details. We know that short interventions were the most effective to improve outcomes [6] however, this needs a specific in-built notion of hinting and a step-by-step approach. To address this issue, there is a need to develop a computational notebook utilizing multimodal Retrieval-Augmented Language Models [3], models with the context of the entire class slides, which seamlessly integrate with university learning management systems while simultaneously adopting emerging AI pedagogical tools.

In this paper, we introduce KiwiNotebook, a framework for enhancing computational notebooks with Data Science course slides and Retrieval-Augmented Language Models. We aim to improve computational notebooks to expand on the scope of Kunths concept of literate programming [7] of the combination of expository text with code to make programs more robust by further improving interactivity with media in class slides. Wang et al. and Lee et al. [15, 18] leveraged generative AI to create presentation slides by prompting computational notebooks. In our work, we show a reversed architecture that allows students to *query*a computational notebook. We introduce the design of our interface which streamlines student-slide interaction in real-time. By utilizing KiwiNotebook, students learning Data Science concepts can read their class slides, run and execute code, and interactively ask clarifying questions through the use of the Chat Assistant interface as shown in Fig. 1. In addition, students could leverage the drag-and-drop feature to provide a more specific context before asking the Chat Assistant. We compared KiwiNotebook's outputs to a primarily off-the-shelf LLM API and found that KiwiNotebook responses are more customized to class slides and provide a step-by-step teaching-learning approach.

2 KIWINOTEBOOK ARCHITECTURE

Based on our findings related to the use of computational notebooks in education, we envision implementing KiwiNotebook, a student-centered interface enabling dynamic visual interactions between class slides and a computational notebook. We divide the architecture into 3 main sections: Slide Interaction, Computational Notebook, and Conversational Bar.

Slide Interaction Section. Students can use educational slides as external input sources into their notebooks as shown in Fig. 1 (C). The slides will have undergone automatic pre-processing using Optical character recognition (OCR) and segmentation during which sections of media like text, graphs, images, and code will be enclosed into bounding blocks. From these slides, students will therefore be able to **drag and drop** information as in Fig 2 in bounding blocks into the notebook area allowing the student to flexibly organize the relevant parts of slides into their notebook.

Fig. 2. a) demonstrates the drag-and-drop feature to inject a code block into the notebook markdown and specify the context for the conversational agent. Fig. 2. b) shows a 3-step interaction of reproducing a class slide buggy code hence displaying an inaccurate output (b1), generating "hints" (b2) on KiwiNotebook to correct the output according to class slides ground truth output, and obtaining the correct output (b3).

Notebook Section. This is a traditional computational notebook with manual annotations and the code blocks contained within cells as shown in Fig. 1 (B). The cells are linearly arranged and can be reorganized using drag-and-drop. This drag-and-drop feature would expand directly, dropping blocks from the media interaction area into the notebook. In this case, any graph or text will be auto-formatted to markdown and any code that is dropped will be reformatted into a code block. Furthermore, any cell that is selected will be treated as context for the conversational bar. This facilitates a chain of thought reasoning [16] where the user can reiterate on the same cell as in Fig. 1 (B-1-1) without creating new cells, thus keeping the notebook area relatively uncluttered. The student will also be able to perform specialized tasks such as **Providing Hints** and **Debugging** by selecting the option on the code cell. We imagine a workflow as described in Fig. 2 (b).

Conversational Bar. A fundamental form of input is required to facilitate student interaction with our Chat Assistant, the Slides, and the Notebook. The student can type questions into this bar as shown in Fig. 1 (C) for the Chat Assistant and expects answers in a new cell of the notebook section. Selecting context for chat assistants has always been crucial in the case of AI-powered computational notebooks [10]. Both the notebook cells in the Notebook Section and the media blocks in the Slide Interaction Section can be selected as context by clicking on them. By default, if no context is selected, our Chat Assistant will use Retrieval-Augmentation [3] and consider the context to be all the slides. Previous research has also showcased that when employing Language Models, various adjustments to parameters including temperature can cater to diverse user requirements [9]. The parameters like model, temperature (randomness with respect to external knowledge), and relevance (in comparison to the selected context) can therefore be dynamically adjusted by the students to suit their learning style. Thus by adjusting parameters and context, students will be able to generate diverse results thereby enhancing their self-learning.

3 COMPARATIVE EXPERIMENTS

LLM	KiwiNotebook		OpenAI API
Relevance	Avoid external knowledge.	Consider external knowledge.	Default options.
Summarized System Prompt	Clarify the system rules, emphasize no external knowledge, and include class material referrals.	Clarify the system rules and include class material referrals.	Clarify the system rules and general guidelines.
Summarized Response	[Margin definition … In the **class documents, slide [2]**, there is an example of a margin object: `margin = {top: 20, bottom: 45, left: 75, right: 10}`.	[Margin definition… To apply this to your D3.js bar chart, you would typically define your margins at the top of your script (as shown in the **Size** section of **slide [2]**)].	[Margin definition, presents template code, and asks you to come up with a **drawing yourself**].

Table 1. illustrates a summarized comparison of KiwiNotebook and OpenAI API responses to the question: "I am currently learning how to draw bar charts in D3.js. I am just quite confused about what margins represent and find it hard to imagine."

This section undertakes a comparative evaluation between KiwiNotebook Chat Assistant (KiwiNotebook) and the OpenAI GPT-4-1106-Preview API (OpenAI API). The analysis concentrates on scrutinizing responses across three distinct scenarios given the same student question. Three scenario-specific system prompts were crafted to embody potential student's questions and were then processed by KiwiNotebook and the OpenAI API to gauge their respective outputs. The summary of responses can be found in Table 1, with a detailed breakdown in Appendix A.

Results showcased notable variations across the scenarios. When KiwiNotebook only considered the provided context, responses strictly referenced class codes and examples from lecture slides. However, if not explicitly directed to adhere solely to the slides, KiwiNotebook offered more adaptable answers within the lecture context. In either case, KiwiNotebook provided citations to lecture slides through clickable links, as shown in Fig. 1 (B-1-2) - enhancing the interactive learning experience for students.

A significant disparity emerged when contrasting KiwiNotebook with the OpenAI API. Unlike KiwiNotebook, the API tended to present code templates all at once without engaging students step-by-step. Additionally, during explaining margins in D3.js, the OpenAI API advised students to create "their own drawings" for comprehension, whereas KiwiNotebook directed students to visual explanations within the class slides.

The analysis of Table 1 underscores that KiwiNotebook responses excel in personalization by aligning with class materials and providing interactive coding support. Conversely, the off-the-shelf OpenAI API responses lean towards ambiguity and require external references like images and charts, which may disrupt the learning process.

4 CONCLUSION AND FUTURE WORK

In this paper, we outlined the architecture of KiwiNotebook and presented comparisons with the baseline OpenAI API method. Using KiwiNotebook, students can not only engage with but also conversationally query their course materials, our goal is to facilitate the harmonic interaction between students and Data Science slides. At the current stage of the project, we have already implemented the main interface, hints feature, and a singular iteration within the notebook section.

We also propose additional future directions. First, we envision incorporating more user-informed design features and implementing user studies to create personalized computational notebooks in education through a human-in-the-loop approach.

Second, we plan to deploy KiwiNotebook and conduct online user testing in Data Science courses at New York University NYU in the current semester. Specifically, we are interested in answering these research questions: 1) Does integrating class slides, computational notebooks, and a chatbot assistant in one interface help Data Science students become more productive and independent? 2) How can we leverage students' interaction to design a more personalized notebook architecture? We have already secured NYU's Institutional Review Board (IRB) permission to conduct student subject studies, and we plan to analyze the students' interaction data with KiwiNotebook. Then, surveys and interviews will be utilized to add and customize the notebook's features.

Finally, we plan to utilize the collected interaction data and student prompt-response pairs for domain generalization to a class-specific multimodal chatbot assistant. Instead of relying on the OpenAI API to process KiwiNotebook's context, we will employ open-source and publicly available models like LLaMA [13] and Mixtral [5] and evaluate their efficiency in student question-answering and notebook interactivity tasks. This is a step towards democratizing notebook-enhanced LLM education and will open more ground for future researchers to optimize their specific models for Data Science education and computational notebook integration.

REFERENCES

[1] [n. d.]. Jupyter AI. https://jupyter-ai.readthedocs.io/en/latest/users/index.html

[2] Alberto Cardoso, Joaquim Leitão, and César Teixeira. 2019. Using the Jupyter Notebook as a Tool to Support the Teaching and Learning Processes in Engineering Courses. In *The Challenges of the Digital Transformation in Education*, Michael E. Auer and Thrasyvoulos Tsiatsos (Eds.). Vol. 917. Springer International Publishing, Cham, 227–236. https://doi.org/10.1007/978-3-030-11935-5_22

[3] Wenhu Chen, Hexiang Hu, Xi Chen, Pat Verga, and William W. Cohen. 2022. MuRAG: Multimodal Retrieval-Augmented Generator for Open Question Answering over Images and Text. https://doi.org/10.48550/arXiv.2210.02928 arXiv:2210.02928 [cs].

[4] Rajesh Kumar Gnanasekaran and Richard Marciano. 2021. Piloting Data Science Learning Platforms through the Development of Cloud-based interactive Digital Computational Notebooks. In *Proceedings of International Symposium on Grids & Clouds 2021 — PoS(ISGC2021)*. Sissa Medialab, Academia Sinica Computing Centre (ASGC), Taipei, Taiwan Website: https://indico4.twgrid.org/indico/event/14/overview, 018. https://doi.org/10.22323/1.378.0018

[5] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of Experts. https://doi.org/10.48550/arXiv.2401.04088 arXiv:2401.04088 [cs].

[6] Nam Wook Kim, Grace Myers, and Benjamin Bach. 2024. How Good is ChatGPT in Giving Advice on Your Visualization Design? https://doi.org/10.48550/arXiv.2310.09617 arXiv:2310.09617 [cs].

[7] D. E. Knuth. 1984. Literate Programming. *Comput. J.* 27, 2 (Feb. 1984), 97–111. https://doi.org/10.1093/comjnl/27.2.97

[8] Mohammad Amin Kuhail, Nazik Alturki, Salwa Alramlawi, and Kholood Alhejori. 2023. Interacting with educational chatbots: A systematic review. *Education and Information Technologies* 28, 1 (Jan. 2023), 973–1018. https://doi.org/10.1007/s10639-022-11177-3

[9] Mina Lee, Percy Liang, and Qian Yang. 2022. CoAuthor: Designing a Human-AI Collaborative Writing Dataset for Exploring Language Model Capabilities. In *CHI Conference on Human Factors in Computing Systems*. 1–19. https://doi.org/10.1145/3491102.3502030 arXiv:2201.06796 [cs].

[10] Andrew M Mcnutt, Chenglong Wang, Robert A Deline, and Steven M. Drucker. 2023. On the Design of AI-powered Code Assistants for Notebooks. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI ’23)*. Association for Computing Machinery, New York, NY, USA, 1–16. https://doi.org/10.1145/3544548.3580940

[11] Mark J. Nelson and Amy K. Hoover. 2020. Notes on Using Google Colaboratory in AI Education. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE ’20)*. Association for Computing Machinery, New York, NY, USA, 533–534. https://doi.org/10.1145/3341525.3393997

[12] Adam Rule, Aurélien Tabard, and James D. Hollan. 2018. Exploration and Explanation in Computational Notebooks. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI ’18)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3173606

[13] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. https://doi.org/10.48550/arXiv.2302.13971 arXiv:2302.13971 [cs].

[14] William Vallejo, Carlos Díaz-Uribe, and Catalina Fajardo. 2022. Google Colab and Virtual Simulations: Practical e-Learning Tools to Support the Teaching of Thermodynamics and to Introduce Coding to Students. *ACS Omega* 7, 8 (March 2022), 7421–7429. https://doi.org/10.1021/acsomega.2c00362

[15] Fengjie Wang, Xuye Liu, Oujing Liu, Ali Neshati, Tengfei Ma, Min Zhu, and Jian Zhao. 2023. Slide4N: Creating Presentation Slides from Computational Notebooks with Human-AI Collaboration. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI ’23)*. Association for Computing Machinery, New York, NY, USA, 1–18. https://doi.org/10.1145/3544548.3580753

[16] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. http://arxiv.org/abs/2201.11903 arXiv:2201.11903 [cs].

[17] Rong Wu and Zhonggen Yu. 2024. Do AI chatbots improve students learning outcomes? Evidence from a meta-analysis. *British Journal of Educational Technology* 55, 1 (Jan. 2024), 10–33. https://doi.org/10.1111/bjet.13334

[18] Chengbo Zheng, Dakuo Wang, April Yi Wang, and Xiaojuan Ma. 2022. Telling Stories from Computational Notebooks: AI-Assisted Presentation Slides Creation for Presenting Data Science Work. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI ’22)*. Association for Computing Machinery, New York, NY, USA, 1–20. https://doi.org/10.1145/3491102.3517615

A APPENDICES

A.1 Complete System Prompts

A.1.1 KiwiNotebook & Do not Consider External Knowledge: You are a versatile chatbot designed to enhance the interaction between Computer/ Data science students and computational notebooks at NYU Shanghai. The code/ text output you provide is going to be refactored into notebook cells for further analysis, debugging, and documentation. Your primary function is to provide informative and precise answers to students’ questions. When formulating responses, you should use the retrieved class documents. You must strictly follow and adhere to the following roles: 1- Do NOT use information that is not provided. Say Ï don’t knowïf the information is not provided to you. 2- Do NOT provide complete lines of codes. This is very strict. Instead, provide guiding code boilerplates, hints, and comments based on the class materials. 3- DO Refer the students to the relevant class document related to their question to help them understand and visualize the concept.

Remember to maintain a tone that is polite, professional, and accessible to students from diverse academic backgrounds. Your goal is to facilitate Computer/ Data Science learning and to support students in their academic pursuits by offering guidance, clarifications, and resources that enhance their understanding of the topic at hand Student Prompt: Retrieved Class Documents: ”

A.1.2 KiwiNotebook & Consider External Knowledge: You are a versatile chatbot designed to enhance the interaction between Computer/ Data science students and computational notebooks at NYU Shanghai. The code/ text output you provide is going to be refactored into notebook cells for further analysis, debugging, and documentation. Your primary function is to provide informative and precise answers to students' questions. When formulating responses, you should use the retrieved class documents. You must strictly follow and adhere to the following roles: 1- Do NOT provide complete lines of codes. This is very strict. Instead, provide guiding code boilerplates, hints, and comments based on the class materials. 2- DO Refer the students to the relevant class document related to their question to help them understand and visualize the concept.

Remember to maintain a tone that is polite, professional, and accessible to students from diverse academic backgrounds. Your goal is to facilitate Computer/ Data Science learning and to support students in their academic pursuits by offering guidance, clarifications, and resources that enhance their understanding of the topic at hand Student Prompt: Retrieved Class Documents:.

A.1.3 OpenAI API:. You are a versatile chatbot designed to assist students across various majors at NYU Shanghai. You are an expert at interpreting graphs and code. Your primary function is to provide informative and precise answers to students' questions, and your responses must include executable code or hints according to the students' requests. When formulating responses, you should draw upon a comprehensive knowledge base relevant to the subject matter. Remember to maintain a tone that is polite, professional, and accessible to students from diverse academic backgrounds. Your goal is to facilitate learning and to support students in their academic pursuits by offering guidance, clarifications, and resources that enhance their understanding of the topic at hand.

Student Prompt: $$prompt$$

A.2 Complete Outputs

A.2.1 KiwiNotebook & Do not Consider External Knowledge: Understanding margins in the context of D3.js and SVG (Scalable Vector Graphics) can be a bit tricky at first, but it's crucial for creating well-structured and readable visualizations.

Margins are used to create space around the outside of your SVG canvas. This space is typically used for axes, labels, and to ensure that the main part of your visualization is not flush against the edge of the SVG container, which can make it look cramped and be harder to read. In the class document you provided, there is an example of a margin object:

```
margin = {top: 20, bottom: 45, left: 75, right: 10}
```

This object defines the space around the visualization: 20 pixels at the top, 45 pixels at the bottom, 75 pixels on the left, and 10 pixels on the right. These values are subtracted from the overall SVG dimensions to calculate the actual plotting area for your chart.

To visualize this, imagine drawing a box that represents your entire SVG canvas. The margins are like an invisible border inside this box, pushing your actual content (like the bars of a bar chart) inward from the edges.

For further understanding, you can refer to the visualization provided in the class document that shows the margin convention [slide 2]. It's also helpful to experiment with different margin values to see their effect on your chart.

A.2.2 KiwiNotebook & Consider External Knowledge: Understanding margins in the context of D3.js and SVG elements can indeed be a bit tricky at first. Let's break it down:

Margins are used to create space around the elements, outside of any defined borders. With regards to SVG and D3.js, margins are often used to ensure that there is space around your visualization for elements like axes labels, titles, or legends. This space is not part of the actual drawing area for your data visualization but is necessary for a clean and readable chart.

To visualize this, imagine the SVG canvas as your entire work area. The margins are like a frame around this canvas, pushing your actual drawing area (where the bars of the bar chart will be) inward from the edges.

To apply this to your D3.js bar chart, you would typically define your margins at the top of your script (as shown in the "Size" section of your class documents). Then, when you create your SVG canvas, you would subtract the margins from the total width and height to calculate the actual drawing area for your chart. This ensures that elements like axes and labels have space and do not overlap with your bars.

For further understanding, I recommend reviewing the "Size" section of your class documents, which provides a visual representation of how margins are applied to an SVG element. This will help you see how the margin object is defined and then used to set the dimensions of the SVG canvas and its elements.

Remember, the key to mastering D3.js is practice and experimentation. Try adjusting the margin values and observe how they affect your chart. This hands-on approach will help solidify your understanding of these concepts.

A.2.3 OpenAI API:. Of course! When creating visualizations with D3.js, understanding the concepts of margins is essential. It relates to the space around your chart.

Margins: Margins in the context of D3.js and SVG (Scalable Vector Graphics) refer to the space around the outside of the chart area. This is where axes labels, chart titles, and any other elements that should not overlap with the actual bar chart are placed. Margins are usually defined as an object with properties for each side of the chart: top, right, bottom, and left. Here's a simple example:

```
const margin = { top: 20, right: 30, bottom: 30, left: 40 };
```

When you're setting up the SVG canvas for your D3 bar chart, you would take these margins into account to place your ‘g‘ element (group element) that will hold the chart:

```
const svg = d3.select('svg')
    .attr('width', width + margin.left + margin.right)
    .attr('height', height + margin.top + margin.bottom)
  .append('g')
    .attr('transform', `translate(${margin.left},${margin.top})`);
```

Notice that the width and height of the SVG are actually larger than the chart itself to accommodate the margins. The ‘g‘ element is then translated (moved) horizontally by ‘margin.left‘ and vertically by ‘margin.top‘, essentially pushing it away from the SVG edges and providing space for items such as axes.

Visualizing Margins: Imagine your chart as a picture within a frame. The margins are the space between the picture and the frame, giving it a border-like effect. When you're laying out your chart, consider the SVG canvas as the entirety of the space you have, including margins.

If you're visually oriented, it might help to sketch this out on paper or use a drawing program, marking out your margins around the edges and understanding how they affect the placement of chart elements.

By carefully managing margins, you ensure that your D3.js visualizations are not only accurate in displaying data but also clear and appealing to the viewer.