



An Institute of



Coding without Coding: Generative AI Support for Introductory Statistics

Dr ZHU Tianming

*Mathematics and Mathematics Education Department
National Institute of Education, Nanyang Technological University*

24 Feb 2026



The "Syntax Wall" in Education



-  In teaching graphical descriptive statistics, hand-drawing is no longer practical for complex datasets; Just as a calculator is necessary for calculation, programming is the mandatory engine required for modern data visualization.
-  The Singapore MOE secondary math syllabus emphasizes *Data Handling and Analysis*, requiring students to move beyond "paper and pen" math to manage large datasets.
-  While programming is essential to statistics and data science, it has become a major roadblock for beginners.
-  As students transition from **No-Code (Blocks)** to **High-Code (R/Python)**, syntax errors often halt mathematical discovery and lead to student frustration.
-  Limited class time makes it nearly impossible for educators to balance teaching technical programming with deep statistical concepts.

Can we achieve “coding without coding”?

Technologies & Capabilities



- **DifyLab Platform:** Built on GPT-4.1 and the [Dify](#) platform, optimized specifically for educational support rather than generic assistance.
- **Restricted Output:** The tool is strictly programmed to provide **code only**. It is forbidden from making decisions or interpreting data, ensuring students must perform their own statistical analysis.
- **Natural Language to Code:** Translate a student's intent (e.g., "I want to draw a histogram") into specific R or Python syntax.
- **Integrated Access:** Seamlessly embedded into the [NTULearn](#) course portal for easy deployment and student access.
- **Web-Based Running:** Works perfectly with [Google Colab](#) and [Posit Cloud](#), meaning students can run complex data analysis on any basic laptop without installing software.



Current Existing AI Code Assistants



Tool	Limitations for Teaching Beginners	Cost / Accessibility
GitHub Copilot	Cannot restrict to code-only mode easily; students may lose the chance to think independently.	Paid subscription
ChatGPT / GPT-4	Cannot reliably enforce code-only usage; may provide full analysis or even conclusions.	Free tier limited, paid for full access
Other AI Tutors	Often not customizable; may directly provide complete solutions including interpretations	Varies; many require subscriptions
Colab Gemini / Posit AI	Some customization possible, but not always fine-grained control	Colab free, Posit free tier limited

Case Study: Plotting Categorical Data in R



Example: "On the average day, about how many hours do you personally watch television?" This was measured as a discrete variable, with values 0, 1, 2, and so on. The table below is one way of constructing a frequency table for the 905 subjects who responded. Display the frequencies using a pie chart and a bar graph, respectively.

No. Hours	Frequency
0-1	232
2-3	403
4-5	181
6-7	45
8 and more	44
Total	905



Conceptual Definition

The student identifies the statistical objective (e.g., visualizing a distribution) rather than worrying about the code syntax.



Guided Generation

DifyLab translates the natural language intent into precise R code, following predefined instructional guardrails.



Active Execution

The student runs the code in Posit Cloud, troubleshooting the environment while being shielded from syntax frustration.



Critical Interpretation

The student analyzes the output to draw statistical conclusions—a step the AI is strictly forbidden from performing.

Practical Workflows

(x) Variables

((language))

R

No.	Hours	Frequency
0-1	232	
2-3	403	
4-5	181	
6-7	45	
8 or more	44	
Total	905	

Manually enter the data into a dataframe

```
survey_data <- data.frame(  
  Hours = c("0-1", "2-3", "4-5", "6-7", "8 or more"),  
  Frequency = c(232, 403, 181, 45, 44)  
)
```

Print the dataframe to check

```
print(survey_data)
```

This code creates a dataframe called `survey_data` that contains two columns: `Hours` and `Frequency`. You manually enter the data based on the categories and their corresponding frequencies. Printing the dataframe helps you verify that the data was entered correctly.

i want to draw a pie chart for this frequency table

Manually enter the data into a dataframe

```
survey_data <- data.frame(  
  Hours = c("0-1", "2-3", "4-5", "6-7", "8 or more"),  
  Frequency = c(232, 403, 181, 45, 44)  
)
```

Draw a pie chart of the frequency distribution

```
pie(  
  survey_data$Frequency,  
  labels = survey_data$Hours,  
  main = "Pie Chart of Hours Spent"  
)
```

This code creates a dataframe with your frequency data and then draws a pie chart showing the distribution of hours spent. Each slice represents one category of hours, labeled with its range. You don't need a file; you enter the data directly in the code.

how to save the figure

Manually enter the data into a dataframe

```
survey_data <- data.frame(  
  Hours = c("0-1", "2-3", "4-5", "6-7", "8 or more"),  
  Frequency = c(232, 403, 181, 45, 44)  
)
```

Save the pie chart as a PNG file

```
png("pie_chart_hours.png") # Open a PNG device  
pie(  
  survey_data$Frequency,  
  labels = survey_data$Hours,  
  main = "Pie Chart of Hours Spent"  
)  
dev.off() # Close the device and save the file
```

This code saves the pie chart as a PNG image file named "pie_chart_hours.png" in your working directory. The `png()` function starts the process of saving, and `dev.off()` finishes and closes the file. You can change the file name in the quotes if you want to use a different name or format.

Case Study: Plotting Categorical Data in R



The screenshot shows the RStudio interface. The console on the left contains the following R code and its output:

```
> survey_data <- data.frame(
+   Hours = c("0-1", "2-3", "4-5", "6-7", "8 or more"),
+   Frequency = c(232, 403, 181, 45, 44)
+ )
> print(survey_data)
  Hours Frequency
1 0-1         232
2 2-3         403
3 4-5         181
4 6-7          45
5 8 or more     44
> pie(
+   survey_data$Frequency,
+   labels = survey_data$Hours,
+   main = "Pie Chart of Hours Spent"
+ )
> png("pie_chart_hours.png") # Open a PNG device
> pie(
+   survey_data$Frequency,
+   labels = survey_data$Hours,
+   main = "Pie Chart of Hours Spent"
+ )
> dev.off() # Close the device and save the file
RStudioGD
  2
Session restored from your saved work on 2026-Jan-12 08:24:34 UTC (21 days ago)
> |
```

The pie chart, titled "Pie Chart of Hours Spent", displays the distribution of study hours. The segments are labeled with their corresponding hour intervals: "0-1", "2-3", "4-5", "6-7", and "8 or more". The "2-3" interval has the largest frequency, followed by "0-1", "4-5", "6-7", and "8 or more".

Then the students can draw the bar plot accordingly...

The chat interface shows a user request: "I want to draw a Bar Graph according to the data". The response includes the following R code:

```
study_hours <- c("0-1", "2-3", "4-5", "6-7", "8 or more")
hours_frequency <- c(232, 403, 181, 45, 44)

Draw a bar graph of study hours

barplot(hours_frequency, names.arg = study_hours,
main = "Bar Graph of Study Hours",
xlab = "No. of Hours Studied",
ylab = "Frequency",
col = "lightblue")
```

The response also includes a note: "This code creates two vectors: one listing the study hours intervals and one listing the corresponding frequencies. It then displays a bar graph showing how many students are in each study hour group. You can type in the data as shown or use a file—just let me know if you need help reading data from a file."

Example on using Python: Analyzing Excel Data

Example: The **SATSCORES.xls** file contains the average SAT scores for each of the 51 states and the District of Columbia for 1990 and 2005. Use graphs to display the two SAT score distributions. How have the distributions of state scores changed from 1990 to 2005?

CONVERSATION ID
5D655733-7190-4E21-B32A-AC9CEC359242

gpt-4.1 CHAT

(x) Variables
([language]) Python

I have an excel dataset named "SATSCORES". It has two columns, one represents the SAT scores in 1990, one represents the SAT scores in 2005, I want to compare the two distributions.

```
import pandas as pd
import matplotlib.pyplot as plt

try:
    sat_data = pd.read_excel("SATSCORES.xlsx")
    plt.figure(figsize=(10, 6))
    plt.hist(sat_data["SAT1990"], alpha=0.5, label="SAT 1990")
    plt.hist(sat_data["SAT2005"], alpha=0.5, label="SAT 2005")
    plt.xlabel("SAT Score")
    plt.ylabel("Frequency")
    plt.title("Comparison of SAT Scores: 1990 vs 2005")
    plt.legend()
    plt.show()
except FileNotFoundError:
    print("The file 'SATSCORES.xlsx' was not found. Please check the filename and location.")
except KeyError:
    print("Make sure your Excel file has columns named 'SAT1990' and 'SAT2005'.")
```

This code loads your Excel file named "SATSCORES.xlsx" and creates overlapping histograms to compare the distributions of SAT scores from 1990 and 2005. Make sure your file has columns called "SAT1990" and "SAT2005". You can input your data by placing the Excel file in the same folder as your code, or by providing the correct path to the file. The code will alert you if the file or columns are missing.



Interpret my histogram

I'm here to help you write and understand code, but I won't interpret results directly. However, here's an explanation of what the histogram code does and how you can use it:

The code creates a plot with two overlapping histograms—one for SAT scores from 1990 and one for SAT scores from 2005. Each histogram shows how frequently certain SAT score ranges appear in your data for each year. By looking at the shapes, centers, and spreads of the bars, you can visually compare the distributions.

To use the code, just make sure your Excel file is named "SATSCORES.xlsx" and has columns called "SAT1990" and "SAT2005". Run the code, and you'll see the plot appear.

If you want to understand your results, try to notice which year's scores are higher, if the distributions are wider or narrower, and if there are any obvious differences. For more details on interpretation, consider asking your instructor or reviewing introductory statistics materials!

The chatbot is limited on interpretation the graphs for students.

Evidence of Student Work

Students applied the "Coding without Coding" framework to solve a 6-question assignment on data visualization.

Q3

(a) Graph

```
> fuel_life <- c(2.0, 3.0, 0.3, 3.3, 1.3, 0.4, 0.2, 6.0, 5.5, 6.5, 0.2, 2.3, 1.5, 4.0,
5.9, 1.8, 4.7, 0.7, 4.5, 0.3, 1.5, 0.5, 2.5, 5.0, 1.0, 6.0, 5.6, 6.0, 1.2, 0.2) # Example data
> stem(fuel_life)
```

The decimal point is at the |

```
0 | 22233457
1 | 023558
2 | 035
3 | 03
4 | 057
5 | 0569
6 | 0005
```

```
> stem_values <- floor(fuel_life)
> freq_table <- table(stem_values)
> relative_freq <- freq_table / length(fuel_life)
> relative_freq_df <- data.frame(
+   Stem = names(relative_freq),
+   Relative_Frequency = as.numeric(relative_freq)
+ )
> print(relative_freq_df)
```

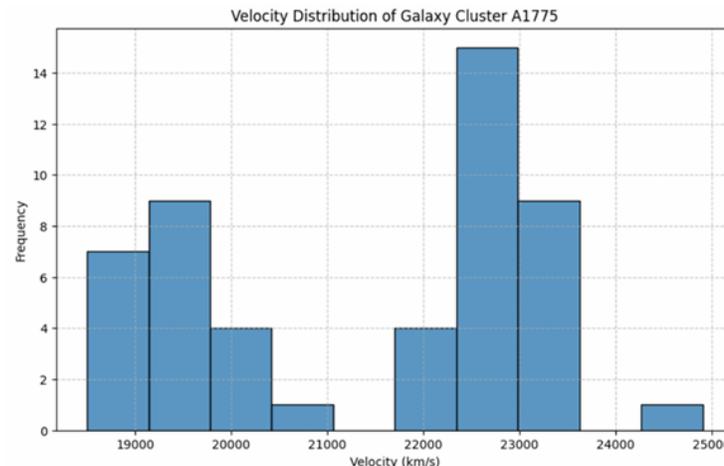
Stem	Relative_Frequency	
1	0	0.26666667
2	1	0.20000000
3	2	0.10000000
4	3	0.06666667
5	4	0.10000000
6	5	0.13333333
7	6	0.13333333

Q5

```
import matplotlib.pyplot as plt
import seaborn as sns

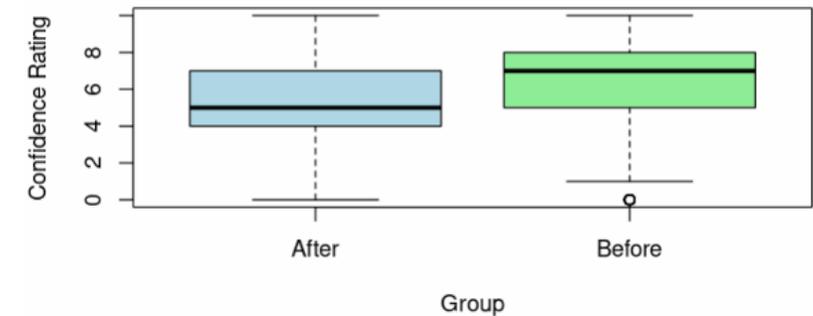
plt.figure(figsize=(10, 6))
sns.histplot(df_galaxy[22922], kde=False, bins=10)
plt.title('Velocity Distribution of Galaxy Cluster A1775')
plt.xlabel('Velocity (km/s)')
plt.ylabel('Frequency')
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()
```

(a)



Q6 (a)

Boxplot of Confidence Ratings Before and After



```
> rating <- 0:10
> respondents_before <- c(16, 16, 34, 34, 53, 395, 269, 395, 359, 125, 107)
> respondents_after <- c(67, 49, 103, 121, 176, 482, 320, 248, 139, 49, 49)
> ratings_before <- rep(rating, times = respondents_before)
> ratings_after <- rep(rating, times = respondents_after)
> group <- c(rep("Before", length(ratings_before)), rep("After", length(ratings_after)))
> all_ratings <- c(ratings_before, ratings_after)
> boxplot(all_ratings ~ group,
+   main = "Boxplot of Confidence Ratings Before and After",
+   xlab = "Group",
+   ylab = "Confidence Rating",
+   col = c("lightblue", "lightgreen"))
```

Why not just use Gemini/ChatGPT?

2. In the 2002 General Social Survey, respondents answered the question, "How many children have you ever had?" The results are

No. children	0	1	2	3	4	5	6	7	8+
Count	799	469	657	481	185	73	40	22	34

(c) Which is the most appropriate graph to display the data? dot plot, stem-and-leaf plot or **histogram?** Why?

(d) Using a sketch or software, construct the graph of this distribution. Based on the graph, characterize the distribution as skewed left, skewed right, or symmetric, and explain your reasoning.

```
[ ] Start coding or generate with AI.
[1] ✓ Os
import pandas as pd

children_data = {
    'Children': ['0', '1', '2', '3', '4', '5', '6', '7', '8+'],
    'Count': [799, 469, 657, 481, 185, 73, 40, 22, 34]
}

df_children = pd.DataFrame(children_data)
display(df_children)
```

Children	Count
0	799
1	469
2	657
3	481
4	185
5	73
6	40
7	22
8+	34

- Gemini or standard ChatGPT will often try to 'be nice' and give the student the full statistical interpretation, effectively doing their homework for them.
- In our testing, Gemini was slower in generating responses and sometimes required additional clarification.

Gemini

T I have a frequency table about the number of children . children 0 1 2 3 4 5 6 7 8+ Count 799 469 657 481 185 73 40 22 34

◆ Okay, I will create a pandas DataFrame from the frequency table for the number of children. Here's the code to do that and display the DataFrame.

T draw a histogram for this distribution

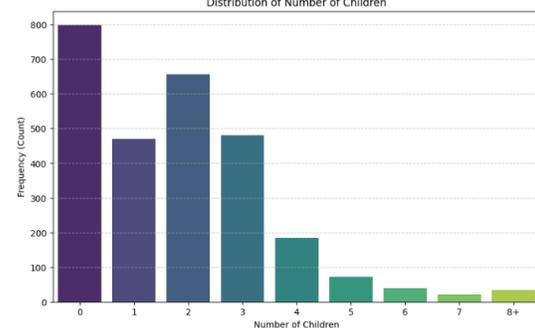
◆ You've provided a frequency table for a categorical variable. A bar chart is the most suitable way to visualize this distribution, showing the count for each category of children. Here's the code to generate it.

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))
sns.barplot(x='Children', y='Count', data=df_children, palette='viridis')
plt.title('Distribution of Number of Children')
plt.xlabel('Number of Children')
plt.ylabel('Frequency (Count)')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

Passing 'palette' without assigning 'hue' is deprecated and will be removed in v0.14.0. Assign the 'x' variable to 'hue' and set 'legend=False' for the same effect.

sns.barplot(x='Children', y='Count', data=df_children, palette='viridis')



Is this a histogram?

Evidence of Impact – Pilot Results (N=8)



Accessibility: The "No-Code" Entry

- **Beginner Friendly:** 8/8 students had little-to-no prior programming background (6 had zero).
- **Technical Reach:** 8/8 successfully navigated R and/or Python code (4 used both; 1 used Python only; 3 used R only).
- **User Satisfaction:** 8/8 found the tool beneficial and easy to navigate for course objectives.

Efficiency: Reclaiming Time

- **Rapid Speed:** 8/8 generated graphs in under 5 minutes; 4 of them finished in under 2 minutes.
- **Significant Relief:** 7/8 strongly agreed the tool removed the frustration of manual coding.
- **Productivity:** 8/8 agreed the assistant made drawing graphs significantly faster.

Pedagogy: Cognitive Logic Over Syntax

- **Intent First:** 8/8 decided on their graph type before asking the AI.
- **Conceptual Bridge:** 7/8 agreed the tool helped them understand the logic of the graphing process.
- **Critical Thinking:** 6/8 actively checked or modified code rather than blindly copy-pasting.

Trust & Long-Term Adoption

- **Reliability:** 8/8 trusted the AI to generate correct and functional code.
- **Future Utility:** 8/8 would consider using the AI Coding Assistant in their future studies or research.

Summary & Future Outlook



An Institute of



Role of Programming in Mathematics Learning

- Programming functions as a tool for mathematical and data reasoning, emphasizing computational thinking over syntax mastery.
- Generative AI supports students by assisting with syntax generation and debugging, allowing learners to focus on statistical concepts and interpretation.
- Supports progression from block-based tools (primary) to AI-supported scripting and statistical computing (secondary and beyond).

Technology and Decision-Making Framework

- AI-assisted coding reduces programming barriers while maintaining authentic data science practices.
- Guided by Singapore's ICT-in-Education Masterplans, National AI Strategy, and computational thinking pedagogy.
- Teachers guide students in interpreting and verifying AI-generated outputs to support reliable and responsible AI use.
- Low-code and AI-supported environments bridge student transition into advanced data science and statistical modeling.

Addressing Equity & Classroom Challenges

- AI provides on-demand personalized support for students with diverse programming and mathematical backgrounds.
- Browser-based and cloud platforms improve accessibility across schools and devices.
- Encouraging students to review and validate AI outputs helps reduce misconceptions and supports inclusive learning.
- Reduces reliance on external tutoring while supporting differentiated learning.

Future-Proofing AI & Educational Technologies

- Model-agnostic learning design ensures adaptability to evolving AI technologies.
- Human-in-the-loop learning emphasizes interpretation, validation, and decision-making rather than automated solution generation.
- Emphasizes transferable skills such as statistical reasoning and critical evaluation.
- Supports scalable and sustainable integration of AI, data science, and computational thinking across K–12 education.

THANK YOU

