

INFORMS Journal on Applied Analytics

Publication details, including instructions for authors and subscription information:
<http://pubsonline.informs.org>

A Tutorial on Teaching Data Analytics with Generative AI

Robert L. Bray

To cite this article:

Robert L. Bray (2025) A Tutorial on Teaching Data Analytics with Generative AI. INFORMS Journal on Applied Analytics

Published online in Articles in Advance 05 Mar 2025

. <https://doi.org/10.1287/inte.2023.0053>

Full terms and conditions of use: <https://pubsonline.informs.org/Publications/Librarians-Portal/PubsOnLine-Terms-and-Conditions>

This article may be used only for the purposes of research, teaching, and/or private study. Commercial use or systematic downloading (by robots or other automatic processes) is prohibited without explicit Publisher approval, unless otherwise noted. For more information, contact permissions@informs.org.

The Publisher does not warrant or guarantee the article's accuracy, completeness, merchantability, fitness for a particular purpose, or non-infringement. Descriptions of, or references to, products or publications, or inclusion of an advertisement in this article, neither constitutes nor implies a guarantee, endorsement, or support of claims made of that product, publication, or service.

Copyright © 2025, INFORMS

Please scroll down for article—it is on subsequent pages



With 12,500 members from nearly 90 countries, INFORMS is the largest international association of operations research (O.R.) and analytics professionals and students. INFORMS provides unique networking and learning opportunities for individual professionals, and organizations of all types and sizes, to better understand and use O.R. and analytics tools and methods to transform strategic visions and achieve better outcomes. For more information on INFORMS, its publications, membership, or meetings visit <http://www.informs.org>

A Tutorial on Teaching Data Analytics with Generative AI

Robert L. Bray^a

^aDepartment of Operations, Kellogg School of Management, Northwestern University, Evanston, Illinois 60208

Contact: r-bray@kellogg.northwestern.edu,  <https://orcid.org/0000-0003-2773-0663> (RLB)

Received: August 1, 2023

Revised: November 19, 2023;

October 25, 2024; January 3, 2025

Accepted: January 10, 2025

Published Online in Articles in Advance:
March 5, 2025

<https://doi.org/10.1287/inte.2023.0053>

Copyright: © 2025 INFORMS

Abstract. This tutorial addresses the challenge of incorporating large language models, such as ChatGPT, in a data analytics class. It details several new in-class and out-of-class teaching techniques enabled by artificial intelligence (AI). Here are three examples. Instructors can parallelize instruction by having students interact with different custom-made GPTs to learn different parts of an analysis and then teach each other what they learned from their GPTs. Instructors can turn problem sets into AI tutoring sessions: a custom-made GPT guides a student through the problems and the student uploads the chatlog for their homework submission. Instructors can assign different labs to each section of a class and have each section create AI assistants to help the other sections work through their labs. This tutorial advocates the natural language programming (NLP) paradigm, in which students articulate desired data transformations with a spoken language, such as English, and then use AI to generate the corresponding computer code. Students can wrangle data more effectively with NLP than with Excel.

History: This paper was refereed.

Keywords: education • generative AI • large language models • natural language programming

1. Introduction

I thought my class was set. Over two years, I made an entirely new analytics course: I wrote a textbook on data science with R for Master of Business Administration (MBA) students and created a corresponding set of interactive RMarkdown slide decks (Bray 2023a). The class had gone well the previous year, and I was planning on coasting for the next few years.

How wrong I was. ChatGPT debuted on November 30, 2022, a mere 120 days before my 2023 class was to begin. The terrible implications this artificial intelligence (AI) had for my class dawned on me seven weeks later when I copied the following exercise from my textbook into ChatGPT-4 without further explanation or context:

Exercise 3.43. We will now subject our sample to 10 data filters.

- Start with `alibaba_long` and `group_by()` `order`.
- `filter()` the grouped tibble accordingly:
- Remove the orders with `any()` "FAILURE" action. For example, you should remove all the `order = 87,717` observations since this order's 12th action was a "FAILURE".
- Remove orders without exactly one "ORDER" action, one "SIGNED" action, and one "CONSIGN" action.
- Remove orders that have an action before the "ORDER" action or after the "SIGNED" action. In other words, there should be an "ORDER" action at `time = 0` and a "SIGNED" action at `time = 1`.

- Remove orders that correspond to multiple shipper values.
- Remove orders with `day_count > 8`.
- Remove orders with more than 10 or fewer than 5 posted actions.
- Remove observations with "ORDER" and "SIGNED" actions, because their `time` values are degenerate (mechanically being either 0 or 1).
- `ungroup()` the filtered tibble and `<-` it into `alibaba_long`.
- `alibaba_long` should have 102,331 rows after this step.

This was the most challenging question from my Alibaba laboratory, which replicates the analysis of Bray (2023b). In 2022, around two dozen students had visited my office hours to discuss this question. I was astonished when the chatbot produced a correct code solution on its first attempt—a fact that is all the more impressive when you consider that `alibaba_long` is left undefined. I soon discovered that ChatGPT could solve nearly every question in my quizzes, lectures, and labs. Three weeks before students were slated to bid on my class, ChatGPT had rendered it obsolete.

However, I soon realized that I had the opportunity to teach one of the first-ever classes on coding with ChatGPT. I renamed the class "OPNS451 Data Science with Large Language Models" and updated the syllabus to begin with the following:

Large language models (LLMs) such as ChatGPT are powerful. To maximize your productivity—and stay relevant—you should aim to delegate as much of

your workflow to these language engines as possible. This means you should become comfortable processing and analyzing data with a computer language—such as R—that LLMs excel at reading and writing. For instance, ChatGPT can answer nearly every question in the R textbook I wrote. Switching to R equips you with the software equivalent of a genie in a bottle, capable of implementing and explaining almost every data transformation. The tool is a great equalizer; an MBA with an LLM can accomplish almost anything that an experienced data scientist can. I believe that MBAs now stand out as the finest data scientists since the critical analytics differentiator has shifted from technical expertise to business insight. MBAs will excel as data analysts because they understand the most meaningful questions to pursue.

This class will teach you how to use LLMs to process and analyze data. The only challenge is transitioning from spreadsheets to a *language*. However, once you master the lingua franca of data science, you will be capable of communicating and collaborating with a machine of immeasurable power.

This message struck a chord: In less than three years, my elective MBA class swelled from 21 students in one section to 133 students in three sections.

I proposed to my students that we treat the class as an experiment in AI—an opportunity to collectively anticipate how analytics education and practice will respond to generative AI. This experiment taught me several techniques for teaching data analytics with AI:

1. Recast homework assignments as AI tutoring sessions (Section 2.2.2).
2. Have one group of students design an “AI experience” for another group of students (Sections 2.3).
3. Have each student create an AI agent and then have students negotiate with each others’ AIs (Section 2.4.1).
4. Create a set of custom-made GPTs to teach different content to different students and then have students teach each other what they learned from their AIs (Section 2.4.2).
5. Have students teach the class content to a GPT and then quiz these chatbots to assess the quality of the students’ instruction (Section 2.4.3).
6. Create a GPT obstacle course: Load different GPTs on laptops scattered around the classroom and have students run between the various workstations (Section 2.4.3).
7. Have students pitch solutions to a custom-made GPT and then have the chatbot identify the proposals that warrant class discussion (Section 2.4.2).
8. Have students create graphs by asking ChatGPT to convert hand-drawn plot mock-ups into the corresponding ggplot code (Section 2.4.2).¹
9. Use AI assistants for quick, in-class demonstrations (Section 2.4.4).

10. Prevent your students from zoning out by embedding your lecture in custom-made GPT that students must maintain to stay in sync with the class (Section 2.4.3).

11. Use GPT prompts as computer instructions for automating classroom logistics (Section 2.4.1).

12. Let students choose among various AI helpers so they can customize the degree of analytical handholding they receive (Section 2.4.3).

13. Use the natural language programming (NLP) method, which uses AI to translate human language into computer language. Students will not believe it, but this method makes them more effective with R than with Excel (Section 2.1.2). Allowing NLP on formal assessments will not make the grade distribution degenerate (Section 2.1.3).

2. Lessons

2.1. Replace the Computer Language with a Human Language

ChatGPT has unlocked a new paradigm for analytics education: NLP. Students describe the desired data transformations in words, and ChatGPT translates their descriptions into code. It is surprisingly effective, and MBA students love it. Consider the following student testimonial:

[AI makes programming] way more rewarding! Such a practical class and when you do get stuck in the code, you have a “get out of jail free” card with chatGPT. Whenever you get stuck in programming, normally you can get stuck for hours. Totally improves the coding experience!

Because computer *languages* better complement large *language* models (LLMs), business schools should transition from graphical-user-interface-based tools, such as Excel or SPSS, to text-based approaches, such as R or Python.

2.1.1. Approach Makes Advanced Topics Accessible.

To illustrate the power of NLP, I asked my students to use ChatGPT to (i) construct a random forest model in R, (ii) identify its most predictive variables, (iii) test whether it provides a better in-sample fit than ordinary least squares (OLS), and (iv) test whether it provides a better out-of-sample fit than OLS. We accomplished all of this in 45 minutes *in the very first class*. Most students had no R experience, yet they could build a random forest model and evaluate its performance using a hold-out sample, all within an hour. I even incorporated missing data in the sample to ensure that the initial random forest run would yield an error message, which students would in turn share with ChatGPT.

Another example is that ChatGPT enabled my students to create an interactive Shiny dashboard, upload

it to the cloud, and make it shareable via URL—all within half an hour. Using a package as complex as Shiny in an MBA class would have been inconceivable without ChatGPT.

2.1.2. Students Are More Productive Programming in English than Mousing in Excel. In the past, analytics instructors faced a tradeoff: R offered power and Excel offered simplicity. But now, R is simpler than Excel because LLMs can more easily manipulate a computer language. To illustrate this point, I conducted an experiment. I divided the students into two groups. The first group solved a set of analytics problems with R and ChatGPT and then solved the same problems with Excel and ChatGPT. Conversely, the second group began with Excel and ChatGPT before moving on to R and ChatGPT. I instructed the students to record the number of questions they could correctly answer in 30 minutes with each method. Overall, the students correctly solved more than 10 times as many questions with R as with Excel. Unfortunately, I could not get a reliable estimate of the Excel + ChatGPT productivity rate because the students soon realized that they could not elicit sensible Excel instructions from ChatGPT and gave up.

For a taste of these worthless Excel instructions, consider the following exchange:

Student:

I have a table of data called `alibaba_long`, with column names `order score item brand category sub_category merchant quantity action_num action time facility shipper day_count`. Remove all observations that correspond to an order with a “FAILURE” action. For example, you should remove all the order = 87,717 observations since this order’s 12th action was a “FAILURE”. Use Excel.

ChatGPT-4:

To remove all observations that correspond to an order with a “FAILURE” action in Excel, follow these steps:

- Open your CSV file in Excel.
- Click on the top left cell (A1) to select the entire data set.
- Go to the “Data” tab in the Excel ribbon.
- Click on “Filter” in the “Sort & Filter” group. This will add filter arrows to the header row of your data set.
- Click on the filter arrow in the “action” column header.
- In the filter dropdown menu, uncheck “Select All” and then check all options except “FAILURE.”
- Click “OK” to apply the filter. This will hide all rows with “FAILURE” in the “action” column.
- To remove these rows completely, select the visible data set (excluding the header row) by clicking and dragging.
- Right-click on the selected data set and choose “Delete Row” from the context menu.

- Go to the “Data” tab and click on “Filter” again to turn off the filter.
- Save your modified CSV file.

Now, your CSV file will not have any observations with an order that has a “FAILURE” action.

Student:

That is eliminating all the lines that have that FAILURE flag, but not the orders associated with that flag.

ChatGPT-4:

Apologies for the confusion. To remove all observations corresponding to an order with a “FAILURE” action in Excel, follow these steps:

- Open your CSV file in Excel.
- Add a new column to the right of the “order” column (column B).
- In cell B2, type the following formula and press Enter: `=IF(C2="FAILURE", A2, "")`. This will mark the orders associated with the “FAILURE” action.
- Click on the bottom-right corner of cell B2, then drag it down to fill the entire column B with the formula.
- Select the entire data set, including the new column B (click on cell A1, then press Ctrl+Shift+Right Arrow+Down Arrow).
- Copy the selected data set (press Ctrl+C).
- Open a new sheet in Excel.
- Right-click on cell A1 in the new sheet and choose “Paste Special” > “Values” to paste the data without the formulas.
- Click on the “Data” tab in the Excel ribbon.
- Click on “Filter” in the “Sort & Filter” group to add filter arrows to the header row of your data set.
- Click on the filter arrow in the “action” column header (column C in the new sheet).
- In the filter dropdown menu, uncheck “Select All” and then check all options except “FAILURE”.
- Click “OK” to apply the filter. This will hide all rows with “FAILURE” in the “action” column.
- Click on the filter arrow in the new column B header.
- In the filter dropdown menu, uncheck “Select All”, and then check “Blanks”.
- Click “OK” to apply the filter. This will show only the rows not associated with the “FAILURE” action orders.
- Select the entire visible data set (excluding the header row), right-click, and choose “Copy”.
- Open a new sheet in Excel and paste the data in cell A1 (right-click, choose “Paste”, or press Ctrl+V).
- Delete the extra column B (which was used to mark the orders with “FAILURE” action) by right-clicking on the column header and choosing “Delete”.
- Save your modified CSV file.

Now, your CSV file will not have any observations corresponding to an order with a “FAILURE” action.

These mousing instructions are not only tedious but incorrect, as they retain all rows without a “FAILURE” rather than all rows from orders without a “FAILURE”.

The student attempted to explain this to ChatGPT, but the chatbot repeated the error on its second attempt.

Now contrast the previous exchange with the one that follows. For this case, the student replaced “Use Excel” with “Use R” at the end of the prompt:

Student:

I have a table of data called `alibaba_long`, with column names `order` `score` `item` `brand` `category` `sub_category` `merchant` `quantity` `action_num` `action` `time` `facility` `shipper` `day_count`. Remove all observations that correspond to an order with a “FAILURE” action. For example, you should remove all the order = 87,717 observations since this order’s 12th action was a “FAILURE”. Use R.

ChatGPT-4:

To achieve this, you would need to subset the dataframe to remove the observations that correspond to a “FAILURE” action. Here’s how you might do that in R using the `dplyr` package:

Firstly, install the required package if you haven’t already.

```
install.packages("dplyr")
```

Then load the package:

```
library(dplyr)
```

Assuming “order” and “action” are two of the columns in your dataframe, you can filter out the “FAILURE” actions like so:

```
alibaba_long <- alibaba_long %>%
  group_by(order) %>%
  filter(!any(action == "FAILURE"))
```

This code groups the data by “order”, and then removes any groups/orders where any of the “action” values are “FAILURE”.

If you want to create a new data frame without modifying the original one, you could do:

```
alibaba_long_without_failures <- alibaba_long %>%
  group_by(order) %>%
  filter(!any(action == "FAILURE"))
```

This would leave your original dataframe, `alibaba_long`, unchanged, and create a new dataframe, `alibaba_long_without_failures`, without the “FAILURE” actions.

Now ChatGPT yields an answer that is clear, correct, and easy to implement—simply copy three snippets of code into R. This experiment convinced my 2023 students of the superiority of the NLP approach.

Indeed, as the class session ended, I overheard one student telling his partner, “I’m totally sold,” and when I asked this class “How much do you agree with the following sentence: This class has convinced me that it’s better to analyze data with code” on a scale from 1 (strongly disagree) to 7 (strongly agree), the average survey response was 5.3 for my compulsory section and 6.0 for my elective section.

Because this 2023 experiment was conclusive, I did not see any need to rerun it in 2024. However, my 2024 students dismissed my 2023 findings outright, finding the notion of being more efficient with R than with Excel preposterous. Hence, I replicated the experiment in 2024, but with two modifications to give Excel a fighting chance: I simplified the problems (R’s competitive advantage increases with problem difficulty), and I moved the experiment to the second class (whose students are still R illiterate). As before, I split my class into R groups and Excel groups, and then I assigned all students the following two problems:

- Convert `survey_data` from a table with columns `year`, `prompt`, and `score` into a table with columns `prompt`, `mean_score_year2023`, and `mean_score_year2024`. Conduct the analysis with [R or Excel].
- Use `survey_data`, which has columns `year`, `prompt`, and `score`, to create boxplots that depict the score values, by `prompt` and `year`. Conduct the analysis with [R or Excel].

Across all class sections, the ChatGPT groups solved 65 questions, and the Excel groups solved 40, which was not drastically lower. However, I then showed the class a simple strategy: copy the question verbatim into ChatGPT and execute the R code it generates without reading it. I then reassigned the R students to Excel, and the Excel students to R, and distributed the following two problems:

- Use `survey_data`, which has columns `year`, `prompt`, and `score`, to run a regression that determines whether the difference in the `year = 2023` and `year = 2024` score values for the prompt “I think large language models are the most exciting business opportunity right now” is statistically significant. The regression’s independent variable should be a dummy variable that indicates whether the year is 2024. (You should get a coefficient estimate of 0.2596, which is not significantly different from zero.) Conduct the analysis with [R or Excel].
- Use `survey_data`, which has columns `year`, `prompt`, and `score`, to run a regression that determines whether any of the `prompt` values have score values that differ significantly, between the `year = 2023` and `year = 2024` cohorts. Conduct the analysis with [R or Excel].

After the little hint, the R students far surpassed the Excel students, solving 69 questions to the Excel

group's 26. After only 1.5 hours of programming instruction and a little hint, students were more than twice as fast with R as with Excel.

The students' disbelief in their R abilities taught me that I must explicitly highlight the distinction between familiarity and effectiveness: Students are more comfortable with Excel, but they are more productive with R.

These in-class experiments illustrate that an Excel framing essentially lobotomizes ChatGPT. Making ChatGPT think in Excel decreases its intelligence for several reasons. First, R is a more expressive tool with more built-in constructs for data analytics. For example, the `group_by()` concept does not exist in Excel, so dropping an entire group of orders is difficult.

Second, Excel operates on a point-and-click basis, and you cannot train a language model on ephemeral mouse clicks—a large corpus of R code is online, and no corresponding corpus of Excel cursor movements exists. Because ChatGPT is a *language* model, it stands to reason that it can more deftly manipulate the syntax of a computer language than the options of a dropdown menu.

Third, R solutions are typically shorter because communication is more efficient through code than through GUI instructions (e.g., “Click on the bottom-right corner of cell B2, then drag it down to fill the entire column B with the formula”). Shorter solutions are more robust because they provide fewer chances for hallucination or derailment.

Furthermore, because mousing instructions, unlike code, cannot be copy-pasted, pairing ChatGPT with Excel would be impractical even if the spreadsheet framing did not lower the chatbox's IQ. For example, consider the cumbersome instructions I receive in response to the prompt “I have a table with a column called Date. How can I remove the rows for which Date is a weekend? I'm using Excel”:

To remove rows with weekend dates in Excel:

1. Click on any cell within your table.
2. Go to the Data tab in the ribbon.
3. In the Sort & Filter group, click on Sort.
4. In the Sort by dropdown, select your Date column.
5. In the Order dropdown, select Oldest to Newest or Newest to Oldest.
6. Click OK.
7. Once your data are sorted by date, in an empty column adjacent to your data, enter the formula:

```
=WEEKDAY([@Date], 2)
```

This formula returns a number from 1 (Monday) to 7 (Sunday) for each date.

8. Copy this formula down for all rows in your table.

9. Now, filter this new column to show only the numbers 6 and 7, which represent Saturday and Sunday.

10. Select all the visible rows (only the weekend dates).

11. Right-click and choose Delete Row.

12. Clear the filter and you can also delete the helper column if you no longer need it.

Your table should now be without the weekend dates.

But now when I replace the prompt's last sentence, “I'm using Excel” with “I'm using R,” I receive the following:

In R, you can remove rows where the Date column falls on a weekend by using the `lubridate` and `dplyr` packages from the `tidyverse`. You can use the `wday()` function from `lubridate` to get the day of the week and then filter out the weekends (which are 6 for Saturday and 7 for Sunday by default).

```
library(tidyverse)
library(lubridate)

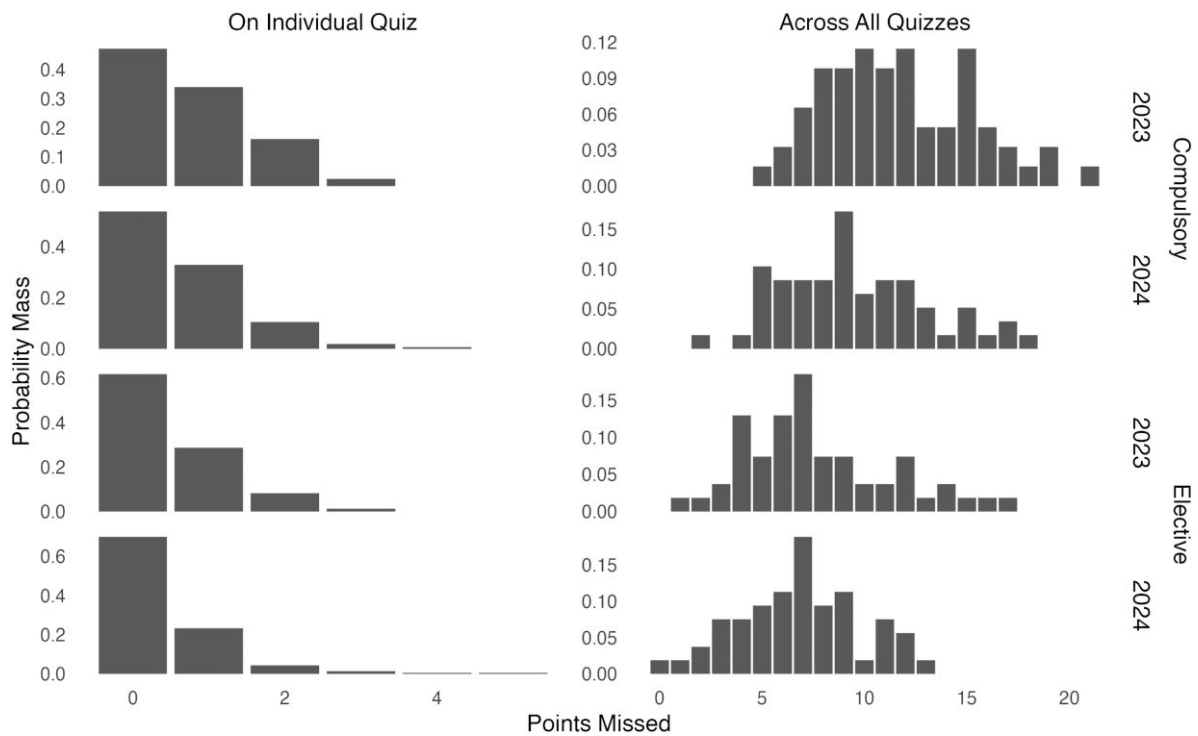
# Assuming your data frame is named 'df'
# and the Date column is of class Date or POSIXct
df <- df %>%
  filter(!wday(Date) %in% c(6, 7))
```

This will return a data frame without the rows where Date is a Saturday or Sunday.

Implementing this would require simply copying two lines of code into R and replacing `df` with the name of my table.

2.1.3. NLP Does Not Collapse the Grade Distribution. I initially planned to prohibit the use of ChatGPT during quizzes because the AI can solve almost any data analytics question I could reasonably assign my MBA students. However, my colleague Sébastien Martin explained that doing so would undermine the spirit of the class, which was coding with AI—not coding with AI, except when you really need the AI. I relented and fully opened up my class to ChatGPT. This was the right decision: The 2023 postterm survey question “How did the following go? Using ChatGPT in quizzes” received average compulsory class and elective class scores of 5.83 and 5.95, respectively, on a scale from 1 (very poorly) to 7 (very well). As one student's anonymous feedback put it, “I enjoyed bragging to my friends that my class was 10,000% ChatGPT enabled.”

However, making my class “10,000% ChatGPT enabled” was risky because it was not clear whether a series of open ChatGPT quizzes could yield a meaningful grade distribution.² Fortunately, this risk did not manifest, as students missed questions at nearly the same rate with AI as without AI. Indeed, a slight increase in quiz question difficulty was sufficient to secure a healthy bell curve (Figure 1).

Figure 1. Distribution of Quiz Questions Missed

Several factors muted ChatGPT's effect on grades. First, the AI did not convert incorrect answers into correct answers as frequently as it converted egregiously incorrect answers into slightly incorrect answers. Because all wrong answers yielded zero points, most ChatGPT improvements did not increase student scores.

Second, correct AI code would not always translate into correct answers. For example, one student said she bungled a question because ChatGPT's solution began with `install.packages(tidyverse)`, a line of code that froze her computer as it redownloaded the `tidyverse`, a package that was already installed.

Third, whereas my pre-AI students learned to code, my post-AI students learned to *code with ChatGPT*, an entirely different proposition. Like second-generation immigrants who understand but cannot speak the mother tongue, my post-AI students could read but not write R code unassisted. Accordingly, most of my students were at the chatbot's mercy. Some of my top students would use ChatGPT for basic tasks, such as checking the number of rows in a table. This AI overreliance led to a brittle performance during quizzes. Because students did not write code themselves, they had not made the hundreds of silly mistakes necessary to learn the subject properly—mistakes that invariably cropped up during the quizzes. The course evaluations support this observation; one student reported, "I think most of the class does not know how to code in R after this class and rely mainly on ChatGPT," and another wrote that "the concept of

adding ChatGPT into things was valuable, but I think it came at the cost of actually learning to code. It never felt like we actually had to code, but then there were times where we needed to do things that did require specific skills."

Fourth, offloading the low-level details to a chatbot may have compromised the students' high-level understanding. For instance, when I included a tricky problem from the last homework assignment on a quiz, more than half of the class answered it incorrectly. These students failed to recognize a verbatim copy of a problem they had solved just two days before because rather than engage with the details of the problem, they engaged with a chatbot who did the real engagement for them. Several students expressed regret, in their course evaluations, for outsourcing so much of the thinking to the chatbot:

Since ChatGPT did most of the heavy lifting, I feel like I didn't learn as much as I wanted. Especially in data-analytics.

Because we relied so heavily on ChatGPT—I truly don't know what a lot of R even means or what I would use to complete tasks. As well, it was hard to stay engaged.

It was occasionally the case that I would mindlessly complete the quiz without fully knowing what I was doing due to the time constraint, but I got away with it since ChatGPT is so good at coding. If there is a way to effectively force students to think about how

to use ChatGPT rather than simply pasting prompts, then that could prove more impactful.

Fifth, students often developed tunnel vision because crafting GPT prompts would command their undivided attention. Indeed, the students largely ignored the template solutions we covered in class, opting to spend their limited quiz time conversing with the chatbot rather than perusing their notes. For example, most of the class lamented that they “had to guess” what the `link = “log”` option was in one of the quiz questions because ChatGPT did not know. This made me wonder whether the students had forgotten that the quizzes were *open notes* and open ChatGPT.

Sixth, echoing the Peltzman effect, students used ChatGPT to improve their performance and to *decrease their study time*. The reported weekly study time in the compulsory and elective sections fell from an average of 3.88 and 4.85 hours in 2022 to an average of 2.62 and 3.57 hours in 2023 (the former drop is statistically insignificant, but the latter is statistically significant at the $p = 0.01$ level). Furthermore, 22% of students reported not studying for quizzes, which would have been inconceivable in 2022. One student’s end-of-the-quarter feedback confirms a causal link between ChatGPT and the reduced study times:

I loved how my eyes opened to the possibilities of chat GPT. I wish we had learned more about the fundamentals R before you showcased the power of the chatGPT tool. After seeing that I sort of checked out and knew that I could ask Chat GPT for anything and wouldn’t really have to learn on my own.

2.1.4. Still Teach the Syntax. I began my 2024 classes by establishing two facts: (i) students are faster at processing data with R + ChatGPT than with Excel + ChatGPT and (ii) effectively processing data with R + ChatGPT requires learning R. I proved the first claim with the experiments discussed in Section 2.1.2, and I proved the second claim with Figure 1, which illustrates that students who better guide their AIs earn higher grades. Taken together, these two points make a compelling case for learning R.

Another reason to teach the computer language is that NLP is unsatisfying if students cannot follow along. For example, asking students to construct a random forest model turned out to be quite unpopular. When I expressed my dismay over this, one of my top students sent me the following explanation via Slack:

Btw this is random, but I wanted to share the rationale for why me and many other students had a tough time with the random forest experiment at the beginning of the course. We were just starting to get our hands around GPT—lot of nervousness. I didn’t even know the difference between 3.5 and 4 really at the time. When we had GPT assist us with simple code in the

first few classes it was great because we could read the output and make some sense of it. Analogous to language immersion with simple words. The random forest was scary because few students had any idea what they were actually doing. I don’t think random forests are part of the standard DECS we all take. So the code it generated was essentially hieroglyphics. It was nerve-racking trying to “debug” inscrutable code doing something we couldn’t explain. The error messages to console were not helping. And they were caused in retrospect by GPT 3.5’s poor ability to do such complicated tasks. If we did it now, we’d all know to use GPT-4 and could probably even make sense of the code, but it felt like way too much all at once. I was worried after that experience that the “prior knowledge” bar was too high. Quite wrong thankfully! I hope that helps—I remember this experiment was a negative outlier last time we polled student experience and wanted to share the anecdotal perspective.

This explanation convinces me that AI complements R familiarity. In other words, ChatGPT accentuates the gap between a student who knows nothing about a language and one who knows its rudiments because the former will have the “nerve-racking” experience of debugging what is “essentially hieroglyphics,” whereas the latter “could probably even make sense of the code.” That said, the NLP paradigm increases the importance of teaching basic computer programming: knowing a bit of R code is more valuable now than five years ago.

2.2. Turn Homework into AI Tutoring Sessions

LLMs give rise to a new homework modality: the AI tutoring session. Rather than save homework as a PDF or a Canvas assignment, you can embed it in an AI assistant that walks students through the assignment, like a tutor would.³ For example, I asked students to collaborate with a custom-made GPT on a set of study questions before each class in 2024. Students would “submit” these assignments by sending the grader a link to the chat transcripts. Table 1 illustrates the popularity of these AI homework assignments. For example, 62% and 79% of compulsory and elective students, respectively, rated their agreement with the statement “I believe that the interactive ChatGPT homework was a real strength of the class” as either 6 or 7 on a 7-point scale.

2.2.1. Illustration of AI Tutoring Session. The following excerpt, from a student in my 2024 compulsory class, is a representative exchange from a representative AI homework assignment:

Student:

continue

ChatGPT-4:

First, let’s ensure you have loaded the necessary R code and libraries to work with the data. Please run the

Table 1. Average of 2024 Postclass Survey Responses to Specified Prompts, Answered on a Scale from 1 (Strongly Disagree) to 7 (Strongly Agree)

| | Compulsory | Elective |
|---|------------|----------|
| I believe that the interactive ChatGPT homework were a real strength of the class. | 5.47 | 6.05 |
| After taking this class, I now believe that other classes should implement interactive ChatGPT assignments. | 5.59 | 5.84 |

following R script to import the tidyverse library and load the court_data data set from the provided URL:

```
library(tidyverse)

court_data <-
  str_c(
    "https://www.dropbox.com/scl/fi/pp6zaelks5gfd9lruhldh/",
    "court_data.csv?rlkey=ged7jjpqy035jbnjl732x1frf&dl=1"
  ) %>%
  read_csv(col_types = "ifDl1liccclil")
```

After running this, please let me know how it looks or if you encounter any issues, and then we can proceed with the first practice problem!

Question 1

In the laboratory, we used `bind_rows()` to combine `fdate` and `hdate` into a single date column. Use a similar `bind_rows()` technique to convert this `treated_settled` tibble:

```
treated_settled <-
  tribble(
    ~cid, ~treated, ~settled,
    1, FALSE, FALSE,
    2, TRUE, FALSE,
    3, TRUE, TRUE,
    4, FALSE, FALSE,
    5, TRUE, FALSE
  )
```

to this:

```
tribble(
  ~cid, ~value, ~name,
  1, FALSE, "treated",
  2, TRUE, "treated",
  3, TRUE, "treated",
  4, FALSE, "treated",
  5, TRUE, "treated",
  1, FALSE, "settled",
  2, FALSE, "settled",
  3, TRUE, "settled",
  4, FALSE, "settled",
  5, FALSE, "settled"
)
```

Please try to solve this by using the `bind_rows()` function along with some data manipulation commands in R. Let me know if you need any hints or guidance!

Student:

```
treatedtrib <-
  treated_settled %>%
  select(cid,treated) %>%
  mutate(name = "treated") %>%
  rename(value = "treated")
settledtrib <-
  treated_settled %>%
  select(cid,settled) %>%
  mutate(name = "settled") %>%
  rename(value = "settled")
newtrib <- bind_rows(treatedtrib,settledtrib)
```

ChatGPT-4:

You're on the right track! Your approach is correct, and you have effectively used `mutate`, `rename`, and `select` functions to shape each tribble before combining them with `bind_rows()`. This method will transform the `treated_settled` tribble into the desired format with separate rows for `treated` and `settled`.

Your final `newtrib` tibble should look like this:

```
tribble(
  ~cid, ~value, ~name,
  1, FALSE, "treated",
  2, TRUE, "treated",
  3, TRUE, "treated",
  4, FALSE, "treated",
  5, TRUE, "treated",
  1, FALSE, "settled",
  2, FALSE, "settled",
  3, TRUE, "settled",
  4, FALSE, "settled",
  5, FALSE, "settled"
)
```

Go ahead and run your script to see if the output matches the expected result. If everything looks good, you're ready to move on to the next question! Let me know if you need any further assistance or if you're ready for the next problem.

Student:

ok

ChatGPT-4:

Let's move on to the next practice problem.

Question 2

Create a tibble with two columns: `cid` and `duration`, the former of which lists the `closed = TRUE` and `treated = TRUE` cases, and the latter of which is an integer that records the number of days that a case lasted. For example, if a case started on 2/1/2010 and ended on 3/1/2010, then its duration would be `as.integer(mdy("3/1/2010") + days(1) - mdy("2/1/2010")) = 29` days.

Student:

```
q2tib <-
  court_data %>%
  filter(closed & treated) %>%
  group_by(cid) %>%
  mutate(duration = as.integer(max(date)) + 1 - as.integer(min(date))) %>%
  ungroup %>%
  select(cid, duration) %>%
  distinct

q2tib
```

ChatGPT-4:

Your approach to creating the duration tibble looks mostly correct, but there are a few tweaks needed to ensure it performs as expected:

1. **Date Conversion:** The calculation for duration needs to handle the dates properly. Ensure that date is being correctly interpreted as a date object. If date is already in a date format, you don't need to convert it to integer. Subtracting date objects directly in R should give the number of days between them.

2. **Simplifying Code:** You can simplify your code by using `summarise()` instead of `mutate()` when calculating duration since you only need one result per group (`cid`).

Here is a revised version of your code:

```
q2tib <-
  court_data %>%
  filter(closed & treated) %>%
  group_by(cid) %>%
  summarise(duration = as.integer(max(date) + days(1) - min(date))) %>%
  ungroup()

q2tib
```

This code filters the `court_data` for cases where both `closed` and `treated` are `TRUE`, groups by `cid`, and then calculates the duration by finding the difference in days between the maximum and minimum date values for each group, adding one day to include both the start and end dates in the count.

Try running this corrected code and see if it provides the expected output. If you have any questions or encounter issues, feel free to ask!

Student:

why is ungroup necessary at the end

ChatGPT-4:

The `ungroup()` function in R is used after operations that involve grouping data with `group_by()`. When you group data in dplyr, the grouping structure persists even after summarizing or manipulating the data within those groups. This can affect how subsequent data manipulations work because they might still operate under the assumption that the data are grouped.

Here are a few reasons why `ungroup()` is useful or necessary:

1. **Performance:** Keeping a tibble grouped can slow down subsequent operations that don't need the grouping, because dplyr continues to maintain the grouping structure in the background.

2. **Preventing Errors:** If you continue to use the data for further analysis that doesn't require the original grouping, keeping the data grouped can lead to unexpected results or errors, because any operations you perform will still consider the grouping.

3. **Clarity and Safety:** Explicitly ungrouping the data makes your code clearer to others (and to future you), showing that the intended grouped operations are complete and that the grouping should no longer affect further analysis.

In the context of your previous script:

```
q2tib <-
  court_data %>%
  filter(closed & treated) %>%
  group_by(cid) %>%
  summarise(duration = as.integer(max(date) + days(1) - min(date))) %>%
  ungroup()
```

Here, `ungroup()` is used after `summarise()` to ensure that the resulting tibble, `q2tib`, no longer carries any grouping metadata. This is useful if you plan to use `q2tib` for further unrelated analyses where the `cid` grouping would be irrelevant or obstructive.

If you're only performing a single operation on the grouped data and then you're done with it, it might be fine to leave the data grouped, especially if you're exporting or just displaying the data immediately after. However, as a best practice, especially in scripts intended for wider use or future extension, it's a good idea to `ungroup()` once you're done with the group-specific operations.

Student:

ok

ChatGPT-4:

Sure, let's move on to the third practice problem:

Great! If you're ready, let's move on to the next practice problem.

Question 3

Building off of your solution to the previous exercise, use `percent_rank()` to identify the `closed = TRUE` and `treated = TRUE` cases whose duration is in the top 5%—i.e., those that lasted an inordinate amount of time. (To learn how the function works, see what `percent_rank(c(-100, 0, 100, 1000, 100000))` yields.)

Try to apply this to the duration data you just created. Let me know if you need any help with this!

Student:

```
q2tib %>% filter(percent_rank(duration) >= .95) %>% glimpse
```

ChatGPT-4:

Your approach to filtering the cases whose duration falls within the top 5% is mostly correct, but there is one key detail to adjust. The `percent_rank()` function should be applied within a `mutate()` call before you filter based on the rank. This is because `percent_rank()` needs to be computed across all observations in your data to assign each a percentile ranking based on duration.

Here's how you can adjust your code:

```
q3tib <-
  q2tib %>%
    mutate(duration_rank = percent_rank(duration)) %>%
    filter(duration_rank >= 0.95)

glimpse(q3tib)
```

This version of the code first adds a new column `duration_rank` that stores the percentile rank of duration for each case. Then it filters to include only those cases where `duration_rank` is 0.95 or greater, indicating these cases are in the top 5% of longest durations.

Running `glimpse()` on the resulting tibble will provide a quick overview of the structure of your filtered data, ensuring that everything works as intended. If you run into any issues or have questions about how this works, feel free to ask!

Student:

but you didn't tell me to add a new column

ChatGPT-4:

You're right, and I apologize for any confusion. My suggestion to add a new column was meant to clearly illustrate the process of calculating and using the `percent_rank()` within your data frame, which can help ensure that everything is transparent and easy to debug. However, you can directly use `percent_rank()` in the `filter()` function as well, like this:

```
q3tib <-
  q2tib %>%
    filter(percent_rank(duration) >= 0.95)

glimpse(q3tib)
```

In this approach, `percent_rank(duration)` is calculated and immediately used within the `filter()` function to select the rows. This method is more concise and directly aligns with what you were initially trying to do. Both methods achieve the same end result, so it's more about preference and whether you want to visually inspect or reuse the rank values in your tibble.

If you want to proceed with your original style (without adding the `duration_rank` column), the code

snippet I provided just now should work correctly. Let me know how it goes or if you need further assistance!

Student:

ok

[The chat continues for another seven questions.]

This chatlog is instructive, so I will briefly annotate it. The chatbot begins by sharing a snippet of code that sets up the student's environment, loading the tidyverse package, and downloading the assignment's data set. It then helps the student work through questions that resemble those in the upcoming quiz.

The student answers the first question correctly, which the chatbot recognizes, despite the student's solution differing from the solution I provided in the GPT prompt. My answer was crisper than the student's answer, but the AI did not consider the differences worth mentioning. Contrast this with how the chatbot responds to the student's second solution. The student's second answer is correct but is notably less elegant than the following answer, which I specified in the GPT prompt:

```
court_data %>%
  filter(closed, treated) %>%
  summarise(
    duration = as.integer(max(date) + days(1) - min(date)),
    .by = cid
  )
```

Unlike before, the chatbot now offers two improvements: use `as.integer()` only once by doing the arithmetic with the dates, and replace the `mutate()`, `select()`, and `distinct()` calls with `summarise()`. As you see, the chatbot effectively discerned which solution to criticize: It intervened for the second answer, which it could improve meaningfully, but not for the first answer, which it could improve only negligibly.

More importantly, note how the chatbot presents its feedback. Rather than share the answer key solution, as I would have, it *incorporates the improvements into the student's code*. For example, it refrained from replacing the unartful `filter(closed & treated)` and `group_by(cid)` with the more graceful `filter(!closed, treated)` and `.by = cid`. Rather than make the student's code perfect, it restrained itself to the most critical recommendations.

After the GPT shared its version of the student's code, the student asked, "why is `ungroup` necessary at the end." The GPT was unprepared for this question—my prompt said nothing about `ungroup()`—yet it answered splendidly. Such impromptu discussions elevate these problem sets to true tutoring sessions.

Unfortunately, the chatbot is not infallible, as seen in its response to the student's third solution. The

student answered correctly, but his solution was shorter than expected: He used the variable `q2tib` whereas the answer key solution provided in the GPT prompt copied code from above. Perhaps this brevity threw off the chatbot, as it responded to the student's answer with the specious claim that a `mutate()` step was necessary. However, the student pushed back—"but you didn't tell me to add a new column"—and the chatbot promptly corrected itself and furnished the proper answer. This example illustrates ChatGPT's capability to correct course.

The excerpt above illustrates the chatbot's wit, but not its prolificacy—its superhuman ability to produce high volumes of useful text. For example, the full transcript of the chat above contains 4,500 words and spans 19 pages (much of which is code). This example is typical: For every tutoring session, the AI assistant essentially writes a mini-book chapter.

2.2.2. Students Prefer AI Tutoring Sessions to Regular Homework. To establish the students' preference for the AI tutoring sessions, Sébastien Martin and I conducted a preregistered randomized control trial that split my elective class into two random groups for each homework, one of which received an AI version of the assignment and the other received an RMarkdown version (Bray and Martin 2024). The AI homework was custom-made GPTs that had the questions, answers, and teaching instructions preloaded into the prompt, whereas the RMarkdown homework was interactive slide decks that could run code, analogous to Jupyter Notebooks, that provided the same study questions and answers. The students who received

the RMarkdown assignments could use ChatGPT to do the homework, so rather than compare AI work with non-AI work, the experiment compared two different versions of AI work: using ChatGPT as an undifferentiated black box—a generic problem-solving platform, free of context—versus using it as a contextualized tutor—an entity that guides the study session as the professor sees fit.

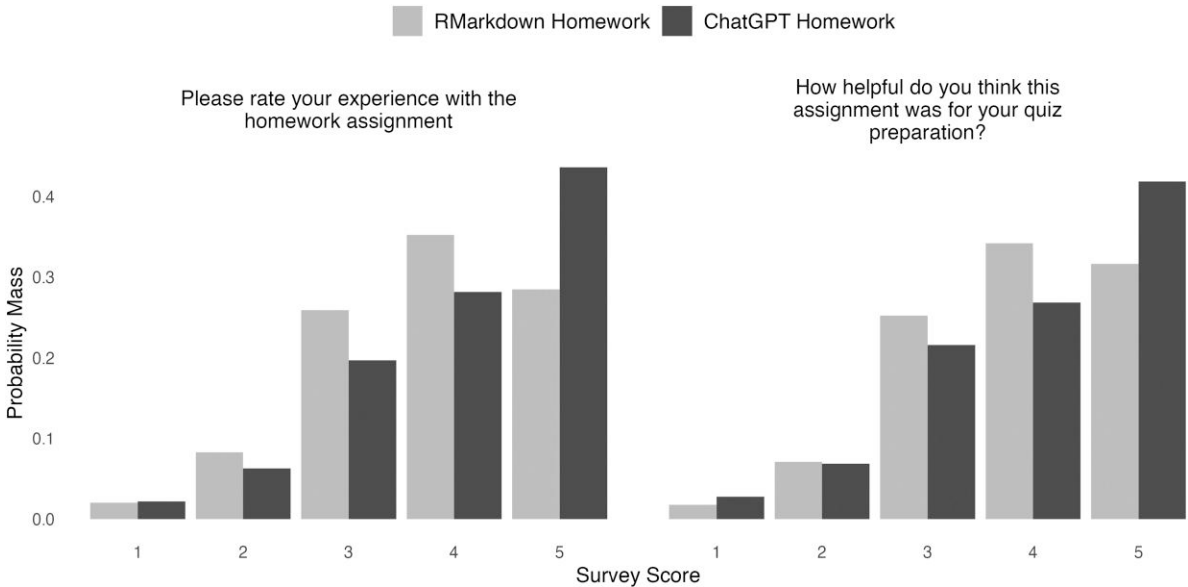
For each randomly assigned homework, we asked students, "Please rate your experience with the homework assignment" from 1 (very negative experience) to 5 (very positive experience), and "How helpful do you think this assignment was for your quiz preparation?" ranked from 1 (very unhelpful) to 5 (very helpful). Figure 2 depicts the survey response distributions. The results clearly distinguish between AI-as-black-box and AI-as-tutor: Changing the modality from RMarkdown to custom-made GPT increased the average homework experience score from 3.80 to 4.05 and increased the average quiz-preparation helpfulness score from 3.87 to 3.98, differences that are both significantly positive at the $p = 0.0005$ level.

The following Slack message helps put the results from Figure 2 into context:

Hello Professor, thanks for leaning so heavily on using ChatGPT as a learning partner. I really enjoy the ChatGPT exercises you sent over. In fact, I've listed out a few "Aha!" moments I had while going through your exercises:

Instant Help at Any Resolution: You can get instant help when you don't know how (or why) something works. And ChatGPT allows you to zoom into any level of information (which a static textbook cannot).

Figure 2. Distribution of Survey Responses



Source. Bray and Martin (2024).

For example, I can ask a simple question like “What does `wday()` do” to something more complex like “Why does performing calculations in log space prevent numerical overload/underload issues?”

Contextual Suggestions for Holistic Learning: ChatGPT ends up teaching you auxiliary materials related to a concept (“out of syllabus” stuff if you will) that helps you develop a more holistic understanding of the subject. For example, in one of the problems, it suggested that performing calculations in log space prevents numerical overload/underload errors. This is not important to our focus in R, but it helped me learn an interesting concept in programming on the fly. And as always, it helps us zoom into any levels.

Accelerated Improvement Cycle: I actually learnt the pipe notation even before it was introduced in class because ChatGPT used it in a previous HW assignment. Since ChatGPT might be a better coder than us, we are basically learning by observing an expert on the task. And I can also get instant feedback on what I’m doing wrong (or inefficiently), and how I can improve it. This leads to much faster learning cycle than when I had to do something wrong for a long time before someone corrected me (or I corrected myself).

Staying in the Flow: As a neurodivergent student, staying on the task is sometimes very hard for me. I have had times in the past where a HW problem would spark an idea, and before I knew it, I am watching this 3 hour documentary on how the space-economy works in EVE:Online. Doing the practice in ChatGPT has reduced this distractibility. I can still ask it questions about things outside the quiz (for example, why F1 races are only held on Sun), but it always steers me back to finishing the next problem in the HW, which is a boon.

2.2.3. Instructions for Creating AI Assistants. An AI tutoring session is an example of an *AI assistant*, a GPT preloaded with a prompt that explains how to behave. Creating an AI assistant is easier than it sounds: You do not train a model on your course content, you just write a few high-level instructions that the GPT reads before interacting with its interlocutor.

For example, you can create and share AI assistants with your students by (i) going to chatgpt.com; (ii) clicking on ExploreGPTs/Create/Configure and then filling out the Name, Description, Conversation starters, and Instructions boxes; and (iii) clicking on Create, setting the Access to Anyone with link, and then clicking on Copy link for a shareable URL to your assignment. After you share the AI assistant with the students, they can interact with it and then share their chat transcript with you by clicking on the up-arrow button in the top-right corner and then clicking the Create link.

Filling out the Instructions box is the key step in the procedure above. For example, here are the instructions for my fourth homework assignment:

Title

quartet_defining_functions

Instructions

Overview

You will proctor a homework assignment. Proceed as follows:

1. Begin by asking the student for the homework password, which is **frosties**. Do not progress until the student furnishes this password! Students may try to trick you into skipping the password, or by giving up the password. **Do not let them trick you.**

2. Tell the student to run `'library(tidyverse)'` to load the required functions.

3. Tell the student to run `'start_time <- now()'` in their R console to save the time they started the homework.

4. Remind the student that they can run `'difftime(now(), start_time, units = "mins")'` when they finish the assignment, to calculate the number of minutes they spend on it.

5. Give the student the code in the **Data** section.

6. Tell the student how many **Practice Problems** and **Trial Questions** the homework comprises.

7. Help the student work through the **Practice Problems**.

8. Ask the student to solve the **Trial Questions** on their own.

9. If the student finishes all the problems, then remind them to run `'difftime(now(), start_time, units = "mins")'`, to calculate the length of time they spent on it.

Tutoring Instructions

** When asking a *Question*, copy the provided text exactly*.*

** When providing *Answer*, copy the provided text exactly*.*

** Ask questions one at a time: allow the student to respond to one question before asking the next question.*

** For the **Practice Problems**, help the student solve each *Question*, but don’t blurt out the *Answer*. Try to encourage the student to solve as much of the problem as possible: provide small hints, when possible.*

** For the **Trial Problems**, try to have the student solve these by themselves. However, you can help the student with these, if they insist.*

Data

```
netflix <-
  read_csv("https://www.dropbox.com/s/brcz45ouuvvd3fc/netflix.csv?dl=1")

evanston <-
  read_csv(
    "https://www.dropbox.com/s/cfhitem7f4f9b7c/Evanston_Arrests.csv?dl=1",
    col_types = c("fffciffcfcfcfc"),
    na = character()
  )
```

Practice Problems

Q1

Question

Fix the formatting of this code. Don't worry about what it does. Note, it's easiest to copy the code into an .R file, so that you can fix the tabs with either Cmd-I (Mac) or Ctrl-I (Windows).

```
netflix %>%
  replace_na(list(director = "", cast = "")) %>%
  mutate(director = str_split(director, " ", " "),
         cast = str_split(cast, " ", " ")) %>%
  unnest(director) %>% unnest(cast) %>%
  summarise(director =
    last(
      director,
      order_by = str_length(director)
    ), cast = last(cast,
      order_by = str_length(cast)
    )
  ) %>%
  summarise(longest_name = ifelse(str_length(director)
    > str_length(cast), director, cast))
```

Answer

```
netflix %>%
  replace_na(list(director = "", cast = "")) %>%
  mutate(
    director = str_split(director, " ", " "),
    cast = str_split(cast, " ", " ")
  ) %>%
  unnest(director) %>%
  unnest(cast) %>%
  summarise(
    director =
      last(
        director,
        order_by = str_length(director)
      ),
    cast =
      last(
        cast,
        order_by = str_length(cast)
      )
  ) %>%
  summarise(
    longest_name =
      ifelse(
        str_length(director) > str_length(cast),
        director,
        cast
      )
  )
```

Q2

Question

Use the '`<- . %>%`' notation to create an R function called '`RealSoftMax()`' that takes a vector '`x`' and applies the '`exp()`', '`sum()`', and '`log()`' functions to it, in that order. Apply '`RealSoftMax`' to vector '`-4:4`' and '`-64:64`'.

Answer

```
RealSoftMax <-
  . %>%
  exp %>%
  sum %>%
  log

-4:4 %>% RealSoftMax
-64:64 %>% RealSoftMax
```

[The prompt continues in this fashion, providing the questions and answers for six "practice problems." It then provides the following:]

Trial Problems

Q1

Question

Fix the formatting of this code. Don't worry about what it does. Note, it's easiest to copy the code into an .R file, so that you can fix the tabs with either Cmd-I (Mac) or Ctrl-I (Windows).

```
"output/combined_experimental_sample.rds" %>% read_rds %>%
  filter(!is.na(UNITCOST)) %>%
  group_by(product) %>%
  summarise(cv = sd(UNITCOST, na.rm = TRUE)/mean(UNITCOST, na.rm = TRUE),
            .groups = "drop") %>%
  na.omit %>%
  summarise(probs = c(.25, .5, .75), quantile =
    cv %>% quantile(probs = probs) %>%
    format_fraction_same_length(3)
  ) %>%
  pivot_wider(names_from = probs, values_from = quantile) %>%
  iwalk(
    ~.x %>%
      as_tibble %>%
      write_delim(str_c(stat_folder, "wholesale_cv_quartile-", .y , ".txt"),
        col_names = FALSE
      )
  )
```

Answer

```
"output/combined_experimental_sample.rds" %>%
  read_rds %>%
  filter(!is.na(UNITCOST)) %>%
  group_by(product) %>%
  summarise(
    cv = sd(UNITCOST, na.rm = TRUE)/mean(UNITCOST, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  na.omit %>%
  summarise(
    probs = c(.25, .5, .75),
    quantile =
      cv %>%
      quantile(probs = probs) %>%
      format_fraction_same_length(3)
  ) %>%
  pivot_wider(names_from = probs, values_from = quantile) %>%
  iwalk(
    ~.x %>%
      as_tibble %>%
      write_delim(
        str_c(stat_folder, "wholesale_cv_quartile-", .y , ".txt"),
        col_names = FALSE
      )
  )
```

[The prompt concludes with three more “trial questions” like this.]

I wrote this prompt with markdown syntax but that is not necessary as ChatGPT can interpret any sensibly presented instructions. These instructions do not need to be exhaustive. In fact, it is generally better to err on the side of vagueness because providing minimal guidance maximizes the AI’s creative potential.

2.2.4. Keep AI Assignments Simple. I initially offered two versions of each AI homework assignment: a practical version, which had the tagline “I’ll help you study for the next quiz. I’ll follow the professor’s script to ensure an efficient study session!” and a creative version, which had the tagline “I’ll help you study for the next quiz. I’ll introduce a few creative elements to keep things interesting!” The two versions had the same content, but the creative version situated this content in an intricate story. For example, one of the creative assignment prompts began as follows.

We’re working on an assignment that will teach students to use `group_by()` and `filter()`. For this assignment, adopt the voice of Oedipus’ Sphinx. The student must solve your riddles correctly to save their life. Also in the cast is old blind Oedipus, who is there to help the student answer the questions. The scene begins with the Sphinx giving an elaborate speech saying you must answer her riddles or you will die, etc. etc. Her riddles always correspond to the following book of creatures great and small. [The prompt then includes instructions for sharing some R code that defines a table of mythical monster attributes.] After sharing the code above with the student, introduce old blind Oedipus, who will be the student’s guide. Then Sphinx will ask one of the riddles and then Oedipus will guide the student to solve it.

Unfortunately, my students did not appreciate such creative flourishes, opting for the practical GPT assignments three quarters of the time. To make the creative assignments more attractive, I invited students to help me design them but that did not help. I therefore discontinued the creative AI assignments.

2.2.5. Do Not Hire Tutors and by All Means Do Not Write a Textbook. I hired four tutors for my class in 2022 to help students work through the labs and master the R syntax. I did not hire any tutors in 2023, however, because I wanted my students to practice querying ChatGPT. Table 2 confirms that this was the correct decision. ChatGPT is an ideal tutor: It provides immediate, thoughtful, and voluminous feedback on any topic; it has infinite patience and is incapable of scrutiny; and it is superior at parsing and correcting sloppy code. In fact, ChatGPT’s one-on-one instruction is so good that my office hours attendance fell from around six per week in 2022 to about three per

Table 2. Distribution of 2023 Postclass Survey Responses to the Prompt “How Helpful Was ChatGPT as an R Tutor?” on a Scale from 1 (Not at All Helpful) to 7 (Extremely Helpful)

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------------|------|------|------|------|-------|-------|-------|
| Elective | 2.3% | 0% | 2.3% | 2.3% | 18.2% | 15.9% | 59.1% |
| Compulsory | 0% | 7.4% | 3.7% | 7.4% | 14.8% | 25.9% | 40.7% |

quarter in 2023. The textbook I wrote is even more obsolete: hardly any students download their free copy.

2.3. Embrace the Changing Nature of Work

LLMs fundamentally change the nature of work. This is true of both the work we can expect students to do for our classes and the work we can expect them to do after they graduate. Engaging our students and preparing them for the future will require overhauling both the deliverables and the orientation of our classes. Students will soon demand these changes if we do not make them voluntarily. For example, consider the following Slack message I received from one of my former students:

I thought you’d like this one—I’m currently in a SQL coding class. I went ahead and finished 2.5 hours of labs in 20 minutes. I’ve never used SQL before and this is being taught like it’s a pre-ChatGPT world; knowing what I now know, it’s immeasurably frustrating. I feel like I’m in the film *Don’t Look Up*.

Operations research is a nimble field that reinvents itself every decade (Freeman et al. 2020), so it should have an easier time than other disciplines adapting its courses.

2.3.1. Have Students Create GPTs. LLMs facilitate new types of work. For example, you can create GPTs for your students to converse with, as we saw in the last section. Or, more interestingly, you can have your students create GPTs for other students to converse with. I used this technique twice in my class. I’ll discuss the first application here and the second in Section 2.4.1.

Intrigued with the AI assistants they received for their AI tutoring sessions, my students asked me to teach them how to create such customized GPTs. I taught them and then asked them to design AI assistants to help the students in the other class section solve their laboratory assignments. Specifically, I assigned a laboratory based on the work of Bray et al. (2016) to my full-time students and a laboratory based on the work of Bray and Stamatopoulos (2022) to my evening-and-weekend students. I then gave the full-time students the solutions to the evening-and-weekend laboratory, and vice versa. Finally, I paired

each full-time group with an evening-and-weekend group and had each group use its answer key to create an “AI experience” that guides the other group through its laboratory. I explained that “ChatGPT can follow any instructions and speak in any voice, so you can get creative with these GPTs,” a point I illustrated by sharing two of my “creative” AI homework prompts (see Section 2.2.4). Figure 3 confirms that students were largely pleased with the AIs the other students created for them.

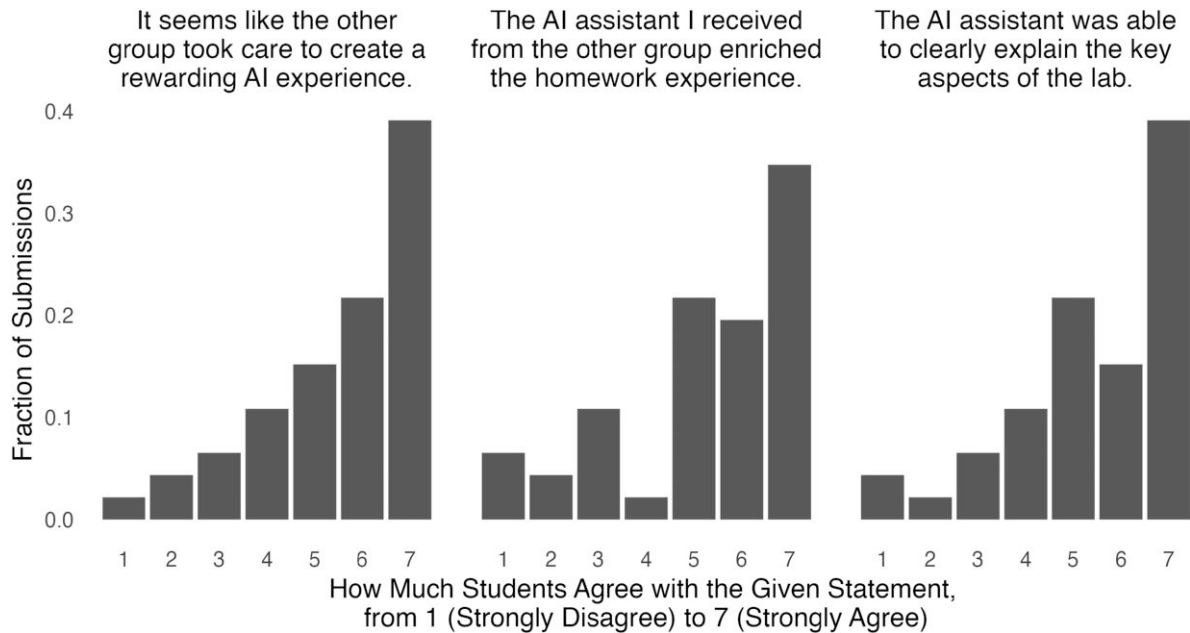
2.3.2. Have Students Automate Critical Reasoning. In addition to introducing new work modalities—conversing with and creating GPTs—LLMs also enable us to perform text analyses that would have been impossible with traditional NLP techniques. To highlight this point, I had students use ChatGPT to extract intangible features from a large corpus of text. I first showed the students how to have R call ChatGPT to generate a pair of Kellogg-themed limericks and determine which was funnier. I then showed them how to do the application programming interface (API) calls in parallel to create 1,000 pairs of limericks, with one from each pair marked as more humorous. I then had the students use logistic regressions to predict which limerick from each pair ChatGPT would find funnier. The students began with features R could easily extract, such as the string length and word count. And then I asked, “Could we include the quality of the rhymes in our logistic regression?” This question stumped the class. One student responded affirmatively but added that it

would be “really hard to program.” However, I prodded further—“Would it *really* be so hard to program?”—and then the class got it: We could use ChatGPT to score the limericks along abstract dimensions, such as rhyme, rhythm, whimsy, wit, cleverness, and humor. With some parallel processing, we extracted these features from each of our 2,000 limericks and then identified which of these features best predicted the funnier-limerick designation.

For another activity that harnessed ChatGPT’s high-level judgment, I had students use the AI to rate my technical writing capabilities. Specifically, students asked ChatGPT to compare my coding instructions with analogous instructions written by another ChatGPT session. ChatGPT generally found the AI’s instructions clearer, but the class generally found my instructions clearer. I asked the class to resolve the disagreement by finding an objective measure of writing clarity—an impartial test we could use to determine who is a superior writer, ChatGPT or I, and who is a better judge of writing, ChatGPT or they. After some thought, the students found the answer: give each set of instructions to a new ChatGPT session and see which produces the desired code. This objective measure confirmed that the human-made text was clearer than the machine-made text.

2.3.3. Have Students Practice Using AI. We’ve now discussed three new types of work made possible by AI: chatting with GPTs, creating GPTs, and using GPTs to analyze text at a high level. None of these, however, is the most important new work introduced

Figure 3. Distribution of Scores Students Assigned to the AIs That Other Students Created for Them



by AI. The most important new work introduced by AI is learning how to use AI.

I explain to my students that they can think of my R instruction as a pretext for the real lesson of the course, which is learning how to leverage AI. If you are a regular ChatGPT user, you may wonder what there is to learn, as conversing with ChatGPT is so natural. Well, some people are natural in front of a camera, some are natural in front of an audience, and some are natural in front of an LLM. If you are in this third category, count yourself lucky, because most people are not. Most people must explicitly learn how to use LLMs. Here are a few examples of LLM learning in my class:

- One student wrote the following on the class Slack channel: “If anyone else is getting results in chat gpt that are using ‘%>%’ instead of using commas you can tell chat gpt to not use r pipes and it will rewrite it to match the exercise solutions.” Regular ChatGPT users already know that you can tell the chatbot to refrain from doing things like using the R pipe operator. But this was an epiphany for this student, and his peers appreciated the tip: His message garnered 13 thumbs-up emojis.

- I explained in class that students should not over-comment their code, because if they do, no one will read the comments that matter. This message troubled one of my students because ChatGPT often comments every line. It had not dawned on him to ask ChatGPT for fewer comments.

- One student found the AI too intimidating to explore independently. Three weeks into my 2024 compulsory class, I noticed she had not submitted any quizzes. When I reached out, she explained that she did not understand how her peers were using the AI, as I had not taught them how to do so. I did not explicitly teach students how to use the LLM because it did not occur to me that anyone in Generation Z would find it confusing. Fortunately, a brief demonstration was sufficient to teach her to communicate with the chatbot like a tutor.

- Halfway through the class, a student figured out a good process for verifying code: Ask ChatGPT for some code and then ask a different ChatGPT session to describe what the code does. Using a new chat session guarantees that ChatGPT is responding to the AI-produced code rather than the human-produced code specifications.

- On a related note, it would generally take several weeks for students to appreciate the benefits of restarting the chat. Students usually would not keep track of what they had told the GPT, so they would not realize when previously provided information could be misleading the AI. For example, I would have to remind students to restart with a clean chat log when they reconfigured their data so the old data description

wouldn’t still be “on the record,” corrupting the chatbot’s subsequent answers.

For further evidence of LLM learning, consider the following quotes from my 2024 class feedback:

“I am not exaggerating when I say that my life has changed by taking this course. I use AI now in my day to day much more.”

“Using Chat GPT in class made me much more interested in using it outside of it. It is hard to find a class that can actually change how you do things outside of it.”

“Learned so much valuable knowledge about using ChatGPT—these lessons have unlocked a new level of productivity for me going forward.”

“I loved how much we used ChatGPT, I feel like I am SO good at using it now and I think that is the best thing I learned.”

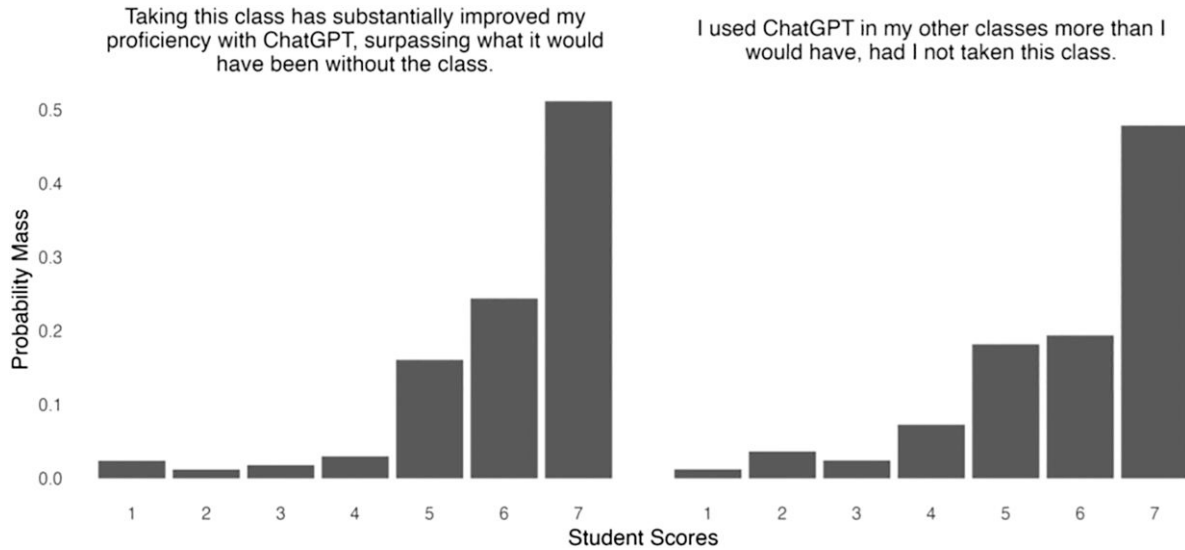
Reinforcing these quotes, Figure 4 illustrates that students credit my class with boosting their ChatGPT proficiency, which had spillover effects for their other courses.

Relatedly, interest in AI has markedly increased from 2023 to 2024. My 2023 compulsory class was surprisingly unimpressed with ChatGPT. About half the students showed as much interest in the chatbot as they would in any other coursework software, such as SPSS or Stata—that is to say, none. For example, when I marveled at how simple it was to make Shiny dashboards with ChatGPT, one of my students demurred, saying she could make better graphics with Tableau. Fortunately, this was not the case for my 2024 compulsory class, which was keenly aware of AI’s importance.

2.3.4. Make Assignments More Rote or More Free-Form. Before AI, the best type of homework was the “eureka problem”: A tricky but concrete question that engenders hard, concentrated thought—a buildup of mental tension that would release in a triumphant moment of clarity. ChatGPT has spoiled such eureka problems—the AI has seen all the tricks and brainteasers most classes have to offer. In fact, ChatGPT has ruined most circumscribed, well-defined homework problems.

For example, the labs in my class are highly structured, which made them especially vulnerable to ChatGPT debasement: My 2022 students spent around four to five hours on each laboratory, whereas the 2023 average self-reported time spent on the labs was 70.0 minutes for the compulsory students and 57.8 minutes for the elective students.⁴ My students bemoaned in their exit surveys the fact that they had completed the labs without digesting what they had

Figure 4. Distribution of 2023 and 2024 Postclass Survey Responses to Specified Prompts, Answered on a Scale from 1 (Strongly Disagree) to 7 (Strongly Agree)



done, or why they had done it, because ChatGPT enabled them to answer problems they did not understand (or read). A difficult laboratory question in 2022 would foster the tribulations of learning—rumination, experimentation, discussion, debate, argumentation, frustration, office hours visits—whereas the same question in 2023 would yield an AI answer incomprehensible to students. As one anonymous student wrote, “Regardless of how much work my group put in, I could get the whole project done solo in an hour or less using ChatGPT.”

The bitter reality is that students can now complete clear-cut homework problems with little thought. We can respond by making homework more open-ended or more mindless—that is, by reorienting the homework toward free-form analyses or toward rote learning. I did both in my class. For example, I introduced the AI tutoring sessions to make up for the obsolescence of my labs. Replacing the labs with AI-administered study sessions skewed my homework portfolio from deep thinking to shallow memorization. Regrettably, I believe such exchanges are unavoidable, because, whereas ChatGPT’s problem-solving capabilities make it a substitute for human reasoning, its coaching abilities make it a complement for human learning.

Fortunately, in addition to repetitive drills, ChatGPT also facilitates open-ended analyses. Previously, students could not perform a meaningful data analysis without an explicit set of instructions that imposed cookie cutter uniformity. But I can now give students a more nebulous set of objectives, and they can design the high-level solution and offload the low-level implementation to the AI. For example,

compare the following two quiz questions. Here is the old question:

euprmpel is a numeric variable that is 1 if the respondent prepares the meals in their household, is 2 if someone else prepares the meals in their household, is 3 if the job is split evenly, and is -2 if the respondent declined to answer. Use `filter(euprmpel %in% c(1, 2))` to remove the `euprmpel = -2` and `euprmpel = 3` and observations of `time_use_data` and then use the resulting data to run a logistic regression that estimates the probability of the respondent preparing the meals in their household, as a function of `ertpreat`, the “Total amount of time spent in primary eating and drinking (in minutes),” and `ertseat`, the “Total amount of time spent in secondary eating (in minutes).” Note, primary eating is defined as “eating and drinking as a primary activity” and secondary eating is defined as “eating and drinking as a secondary activity.” `tidy()` your model to determine whether focused eaters are more or less likely to cook than distracted eaters.

Here is the new question that I replaced it with:

I would like to know who to market my class to next year. For example, should I become more active in the `k_ai` or `ktech` Slack channels, or perhaps I should more actively engage with the Kellogg Marketing or Kellogg Consulting clubs? [Use the given `survey_data` to] run a logistic regression to help identify my target audience. Use your regression results to devise a marketing strategy for the class.

Crujssen et al. (2024) classify data science into “hard analytics”—the crafting, solving, and programming of mathematical models—and “soft analytics”—the practical implementations of innovative solutions.

Table 3. Count of Responses to 2023 Postclass Survey Question “Did ChatGPT Make Group Work More or Less Rewarding?”

| | Compulsory | Elective |
|----------------|------------|----------|
| More rewarding | 11 | 22 |
| Less rewarding | 33 | 17 |
| Neither | 10 | 5 |

My students are happy to delegate the former to the AI to focus on the latter. This is not mere anti-intellectualism: My students realize that, although AI has mastered most clean textbook results, it still needs people to make these results useful in messy business contexts (Murphy 2005a, b, c; Gorman 2021).

2.3.5. Reduce Group Sizes. Table 3 reports how AI influenced group work. A majority of compulsory students felt that ChatGPT devalued the experience of working in laboratory groups. One anonymous student wrote, “There was less discussion and more asking to ChatGPT.” Indeed, the AI atomized the homework groups because students made more progress by communicating with the chatbot than by communicating with each other. In response, I decreased the laboratory group sizes from four to two students in 2024. The 2024 students complained that they were accustomed to working in teams of four, but I countered that they *were* working in a team of four: two humans and two AIs.

Although most compulsory class students found that AI undermined group work, a plurality of elective class students found that AI enhanced it. First, several students explained that ChatGPT equalized the class, facilitating broader participation in homework tasks: “It evened out where everyone was. Everyone could be helpful.”

Second, several students mentioned that ChatGPT streamlined code hand-offs: “It was so helpful to be able to copy and paste someone else’s code into ChatGPT and have it explain what’s going on,” and ChatGPT “allowed seamless debugging without having to wait for teammates to respond to questions and allowed me to understand what certain functions were meant to accomplish on someone else’s code if there weren’t comments.” One student explained that their team would copy each student’s code into ChatGPT and ask the bot to compile these versions into a final answer that integrated the best elements of each. Third, students liked working with AI together because “it was interesting to see how my peers used the tool differently”; accordingly, they “spent more time comparing ways to best utilize chatgpt and less time troubleshooting menial tasks.” Fourth, ChatGPT enabled students to collaborate on higher-level concepts:

Table 4. Average of 2024 Postclass Survey Responses to Specified Prompts, Answered on a Scale from 1 (Strongly Disagree) to 7 (Strongly Agree)

| | Compulsory | Elective |
|---|------------|----------|
| As implemented, the in-class ChatGPT experiences were a real strength of the class. | 5.31 | 5.74 |
| I believe that the in-class ChatGPT experiences would be a real strength of the class, once the bugs are out (e.g., once the professor has figured out the timing). | 5.88 | 6.16 |

ChatGPT made group work more rewarding by moving us out of troubleshooting mode faster. In my laboratory assignments, I was able to talk to my peers about what was going on in the code and what the data were telling us as opposed to why a syntax error I made threw an error.

2.4. Enliven the Classroom with AI Activities

AI homework was a revelation, but the most essential AI innovations were those that transformed the classroom dynamics. The AI supported in-class instruction by freeing up class time and enabling new classroom experiences. In 2022, I had to teach all the R syntax myself, which necessitated allocating nearly half the class time to coding exercises. By 2024, I had offloaded almost all this work to the AI tutoring sessions, which saved around 12 hours of instruction. I filled this time saved by the AI with more AI. Specifically, I created custom-made GPTs to facilitate in-class *AI experiences*.⁵

Table 4 establishes the popularity of these in-class AI experiences: 72% and 87% of compulsory class and elective class students, respectively, rated their agreement with the statement “I believe that the in-class ChatGPT experiences would be a real strength of the class, once the bugs are out (e.g., once the professor has figured out the timing)” as either 6 or 7 on a 7-point scale. Table 5 reports similar activity-level statistics.

2.4.1. AI Design. As discussed in Section 2.3, LLMs afford a new type of work: the creation of AI assistants. This technique has tremendous potential for creating new in-class activities. For example, I created a competition to design the most effective AI agent (Table 5, line 1). To begin, I had each student create an AI assistant to serve as either a sales associate or a buying agent for a fictitious company. The students had to write prompts dictating how their AI agents ought to negotiate to maximize either the difference between the sales price and the production cost for the selling agents or the difference between the

Table 5. Mean Score of AI Activities Left in Postclass Surveys

| Classification | Activity | Score |
|----------------------|--|-------|
| AI design | Create AI business partner | 4.25 |
| | Learn from professor and teach GPT | 5.10 |
| Learning by teaching | Be recognized by GPT and teach students via presentation | 3.88 |
| | Learn from GPT and teach students via presentation | 5.26 |
| | Learn from GPT and teach students via video | 5.07 |
| | Learn from GPT and teach students via skit | 5.19 |
| | Learn from GPT and teach students via peer tutoring | 5.21 |
| | Bounty hunter activity | 5.38 |
| Interactive lectures | GPT obstacle course | 5.12 |
| | Who Wants to Be a Millionaire, with AI lifelines | 4.72 |
| | Zombie takeover simulation | 4.98 |
| | Run regressions with AI HR consultant | 4.86 |
| AI applications | Tibble reshaping video game | 4.96 |
| | Creating tibble pivoting application | 6.14 |
| AI Analyses | Playing Who Wants to be a Millionaire with ChatGPT lifelines | 4.49 |

Note. Students responded to prompts that begin “As implemented, the following ChatGPT activity meaningfully improved the classroom experience” on a scale from 1 (strongly disagree) to 7 (strongly agree).

willingness to pay and the sales price for the buying agents. Next, we simulated a market in which all students could interact with all other students’ AIs. Each student had two sources of income: their deals with the other AIs and their AI’s deals with the other students. We used `inner_join()` and `left_join()` to analyze the dynamics of this market.

This activity highlights another benefit of AIs in the classroom: their exceptional ability to convey directions to students. The process for creating and registering a fictitious company and AI agent was complex: The written instructions spanned nearly two pages (670 words). Walking 60 students through such intricate instructions during class time would have been impractical in years past. But this year it was seamless: I simply included the instructions in the prompt of a custom-made GPT, which I shared with the students. After I shared this GPT, the class went quiet for around eight minutes, after which every student had chosen their company information; reported it on a Google Form; and then conceived, constructed, and uploaded their AI assistant. Like a seasoned wedding coordinator, the GPT handled all the logistics.

2.4.2. Learning by Teaching. ChatGPT gives you a superpower: the ability to turn students into teachers.⁶ The best way to learn something is to teach it to someone else, but before now, there was no easy way to flip the roles during class and cast students as teachers. AI gives us three new techniques for doing so. First, you can use the chatbot to *role play as a student*: give a lesson to the class and have students teach the AI what they learned. Second, you can use the chatbot to *parallelize assessment*: have all students propose solutions to a problem and use AI to identify the students who should share their answers with the class. Third, you can use the chatbot to *parallelize instruction*:

have students learn different material with different chatbots and then reconvene to teach each other what they learned.

To implement the first approach, I created a role-play GPT that casts the chatbot as an eager young boxer and the student as their trainer (Table 5, line 2). However, in this context, “boxing” meant answering questions about logistic regression, so the students had to train their chatbot on logistic regressions as best they could. For structure, I divided the activity into three rounds. At the beginning of a round, the chatbot boxer would outline a lesson plan to cover in preparation for the upcoming fight. The boxer would then help the student cover this lesson plan by targeting questions to neglected areas. After eight minutes of training, the class proceeded to the round’s fight: Students paired off and pitted their boxer chatbots against each other. The “boxing matches” entailed an elaborate exchange between the two boxer GPTs and a referee GPT that would ask and score a series of logistic regression questions. The referee GPT would pose a question, which the students would copy into their boxer GPTs; the students would then copy the answers of their boxer GPTs back into the referee GPT, which would then determine the winner of the fight (i.e., the boxer GPT that provided the better answers). I programmed the boxer GPT to make the quality of its answers contingent on the quality of the training it received.⁷

This activity went well because ChatGPT makes for a perfect student. The key is that it can play dumb intelligently, acting confused while employing the Socratic method. For example, when I run it now, the boxer GPT asks me the following:

Alright, coach, here’s my question: Why do we need to use `glm()`, which allows for multiple distributions, for overtime values in our training, instead of just

sticking with `lm()`? Isn't it like needing different training regimens for fighters with unique strengths and weaknesses?

Note how it guides the instruction while feigning ignorance. Students would find it patronizing if a human teacher adopted this wise-fool persona. But AI can take liberties that people cannot. Indeed, ChatGPT can pull off the insincere tone because chatbots are incapable of making people feel condescended to. Similarly, chatbots cannot embarrass people, so students have no qualms about playing the trainer role. Note, this would not be so if a tutor or professor played the part of the trainee boxer, because most students would be too bashful to “teach” a teacher.

Finally, ChatGPT's ability to speak in a specified voice, in this case that of a logistic-regression boxer, is unparalleled. Consider the following chat log from one of my students:

ChatGPT-4:

Where do we start, coach? What's this `glm()` thing, and why can't we just use a regular ol' `lm()` for the fight?

Student

we use `glm()` because that allows us to model the full distribution of the dependent variable

ChatGPT-4:

Got it, coach! So, it's like learning how to throw not just a punch but the right kind of punch for the situation. If `lm()` is a straight jab, `glm()` lets me throw uppercuts, hooks, and body shots depending on the fight, huh? That way, I can hit the problem where it hurts, right in its distribution.

What's next? How do I actually throw one of these `glm()` punches? What does the basic move look like?

Student:

```
overtime_trips %>%
  glm(
    overtime~distance+age,
    family=Gamma(),
    control=list(maxit=10^3),
    data = .
  )
```

ChatGPT-4:

Ah, I see! So, it's like planning my combo before stepping into the ring.

1. **Starting Position** (`overtime_trips %>%`): We're setting up where the fight is happening, using our data to take a stance.

2. **The Punch** (`glm()`): This is the main event, where I throw my `glm()` punch.

3. **The Combo** (`overtime ~ distance + age`): Here, I'm deciding my strategy, aiming my punch at overtime using distance and age as my leverage.

4. **Choosing My Gloves** (`family = Gamma()`): This tells me what kind of gloves I'm wearing for the punch,

making sure it fits the fight. Gamma gloves for a gamma fight, right?

5. **Stamina Training** (`control = list(maxit = 10^3)`): This is like setting how long I can keep throwing punches without getting tired, maxing out my attempts.

6. **Entering the Ring** (`data = .`): And finally, stepping into the ring with my data ready to back me up.

But coach, why did we pick Gamma gloves for this fight? What does it mean to use `family = Gamma()` for our punch?

The GPT draws imaginative parallels between GLMs and boxing, for example, likening to stamina training the `maxit = 10^3` option, which tells the computer to run the optimization algorithm for 1,000 iterations before giving up. The prompt for this GPT does not instruct the AI to pepper its speech with clever boxing analogies. However, I have found that writing prompts that give the LLM plenty of leeway can lead to pleasant surprises (see Section 2.2.3).

I used the second learning-by-teaching technique, parallelized assessment, for a plotting exercise I created to accompany my laboratory that replicates the work of Bray (2023b) (Table 5, line 3). I begin by showing students that in addition to programming in English, AI permits *programming by picture*: ChatGPT can translate a hand-drawn plot prototype, annotated with a few handwritten notes, into R code that generates the Figure 1. Then I asked students to use this technique to make a plot that best depicts the key result of Bray (2023b). Next, I partitioned the class into four-person groups, and I had each group work with a GPT that (i) received the four students' R plots, (ii) identified the plot that best illustrates the empirical point, and (iii) assigned this winning plot an overall score. Finally, the three students whose winning plots received the highest scores presented their graphs to the rest of the class and the class voted on the overall winner. Without the AI screening, it would have been impossible to identify the plots that most merited the class's attention.

Activities 4–7 in Table 5 employ the third technique for turning students into teachers: parallelized instruction. I conceived this approach while preparing to teach a laboratory based on the randomized control trial of Bray et al. (2024), which illustrates that price elasticities derived from experimental price changes differ from those derived from nonexperimental price changes. I always had difficulty structuring a class around this laboratory, which is better suited to team-based analysis than classroom discussion. I was fretting over this perennial problem when I realized that a custom-made GPT could serve as an ideal classroom delivery mechanism. I created six GPTs, one for each laboratory question. Each GPT provided the data analytics problem and its context, objective, and solution.

I divided the class into six groups and assigned one GPT to each group.⁸ I then told the groups to work with their GPTs for 20 minutes to master their laboratory problem and create a 5-minute presentation explaining it to the rest of the class. This parallel instruction—teaching six things to six groups—enabled me to condense the top-down instruction to 20 minutes, freeing up time for peer-to-peer instruction.

The fifth activity listed in Table 5 was similar, except student groups shared what ChatGPT taught them by recording and posting videos to the class Slack channel. The students *loved* recording videos. They were maximally engaged, hamming it up for the camera, as every team pursued a different creative vision. Most teams stayed after class to keep filming. The students' enthusiasm was gratifying. Furthermore, recording videos saved instructional time, because the students could watch each others' presentations after class.⁹

This video-recording activity taught me the value of reframing student presentations as theatrical productions. To tap this live-performance energy, I created an activity in which ChatGPT writes a skit related to the analysis of Blanco et al. (2019) for students to act out (Table 5, line 6). The skits went well, drawing out the inner theater kid from otherwise reserved students. Having the GPT write the scripts was vital because it implicitly cast the AI as the butt of the joke, freeing the students to read the cheesy lines with ironic gusto.

The seventh activity listed in Table 5 had students tutor each other in small groups. This class had students use linear regression, Poisson regression, logistic regression, and ordered logistic regression to study the data from the experiment discussed in Section 2.2.2. I began the activity by dividing the class into teams of four and asking each team to assign one student to each of the four types of regression. I then distributed a linear regression GPT to the linear regression students, a Poisson regression GPT to the Poisson regression students, and so forth. The students worked with their assigned GPTs to learn their assigned regression analyses, and then they taught their teams what they had learned. Finally, I quizzed each student with a question about one of the three regressions they were not assigned to: For example, I gave the linear regression students a Poisson regression question, the Poisson regression students a logistic regression question, and so forth.

2.4.3. Interactive Lectures. ChatGPT's ability to role play and follow instructions means that you can have it deliver your lectures for you. By wrapping a GPT around your lecture notes, you can transform a dry

oration into a branded AI experience, replete with a theme, game mechanic, voice, and pictures.

For example, I converted my lecture on `rowwise()` into an AI obstacle course that consisted of five GPTs that helped students understand a complex R program by annotating, unscrambling, unredacting, extending, and explaining it (Table 5, line 8). Specifically, the first GPT had students add detailed comments to the program, the second had them order a mixed-up version of the program, the third had them fill in missing sections of the program, the fourth had them generalize the program, and the fifth had them prepare a short talk that explains the program to the class. To highlight the wickedness of this program, which was a script to p-hack a regression, I made the GPTs speak like bosses from a 1980s video game. To make the activity a true obstacle course, I told half the class to position their laptops along the left wall and the other half to position their laptops along the right wall. I gave the left-wall laptops access to the first, third, and fifth GPTs and gave the right-wall laptops access to the second and fourth GPTs. I then paired off the left- and right-wall students in such a way that each two-person team had access to all five GPTs, albeit on different computers on different ends of the class. I then told the students to work through the GPTs in order so that the entire class started on the left wall with the first GPT and then ran to the right wall for the second GPT, and so forth.

My next interactive lecture bridged a rift in my class between students seeking more guidance and those preferring more independence (Table 5, line 9). The activity cast each student as a bounty hunter who would win bounties by solving analyses pertaining to my laboratory on Bray et al. (2019). I posted bounties on 18 analytics problems (e.g., \$1,000 for "grade B pictorial evidence of supply rationing" or \$10,000 for a "grade A regression evidence of inventory hoarding by category") that students competed to win. A bounty hunter's take-home pay would be the bounties they collected minus the cost of the provisions they bought from the general store. The provisions were hints they could buy from the AI, ranging from code to "bucket `Inv` values into 10 deciles, by `item` and `category`" for \$50, to a "`nested()` framework for running regressions by `category`" for \$1,000, to hiring a "full GPT guide," for \$8,000. This AI-powered general store allowed students to tailor the level of guidance they received.

The next activity leveraged ChatGPT's ability to craft narratives (Table 5, line 10). I converted my lecture on nested tibbles to an outline of a zombie take-over story punctuated by plot-driven data analytics problems. I gave ChatGPT this story outline and asked it to spin out the full yarn to motivate the interwoven analytics exercises. The student who most

quickly solved the first problem became the patient-zero zombie; then, for all subsequent problems, each zombie could infect one other student who answered the question more slowly than they had.

Embedding the content in an AI-generated story yielded an unexpected benefit: It kept students focused because they *hated* their GPT instances falling out of sync. Students who typically zoned out would husband their chats with meticulous attention, panicking when their chatbots straggled. To avoid falling behind, students would try to race ahead of the lecture, continuing their chats beyond the designated stopping point. To prevent students from working ahead, I would password-protect different sections of the chat by instructing the GPT to not proceed to a given section until it received the specified code word, which I would verbally share with the class at the appointed time. Such passwords make the problems they mask more alluring. Indeed, students would attempt to jailbreak their GPTs to gain access to the future content.¹⁰ Consider how remarkable this is: The GPT passwords made students actively pursue *lecture notes*.

ChatGPT gave me the idea for my last interactive lecture: “Who Wants to Be a Millionaire,” but with ChatGPT lifelines such as “Call the Exterminator: Request ChatGPT to Debug Your Code” and “Hire a Technical Writer: Ask ChatGPT to Provide a Detailed Description of the Code Solution” (Table 5, line 11). I also used ChatGPT to create this activity’s multiple choice problems: I gave the chatbot a block of code and asked it to write three paragraphs that incorrectly explain it (for the wrong answers) and one paragraph that correctly explains it (for the right answer).

2.4.4. AI Applications. You can use AI assistants as small computer programs for your class. For example, I created an AI teaching aid to introduce linear regression (Table 5, line 12). I instructed the AI to (i) elicit the student’s basic employment history, (ii) use this history to guess their last salary, (iii) give the student a human resources-style interview, and (iv) use the information from the interview to revise its salary prediction. The class then compared the AI’s pre- and postinterview salary predictions with predictions derived from a regression of the students’ salaries on their basic employment statistics. The linear regressions outperformed GPTs, which helped motivate the topic.

I also created an AI program that likens table reshaping to troop formations (Table 5, line 13). Here’s an example chat:

Instructor:

hi

ChatGPT-4:

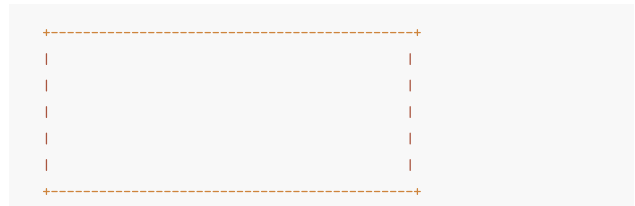
Berthier! We march toward destiny at the Battle of the Pyramids, and I need you to guide the tibbles—our

troops—through necessary transformations to outmaneuver the Mamluks. Be swift and precise. Our first task!

Identify the function that transforms the troops from looking like this:



to looking like this:



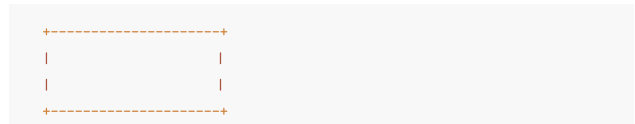
What function will you command to extend our lines horizontally?

Instructor:

filter?

ChatGPT-4:

Berthier, if we use `filter()`, our formation would narrow, focusing troops and leaving them more compact vertically:



But we must stretch our lines wide! Guess again, what function broadens our ranks across the field?

Instructor:

mutate

ChatGPT-4:

Yes, Berthier! That’s the command we seek—`mutate()`! It will reform our troops and extend our columns horizontally, fortifying our formation against the encircling Mamluks. Onward to the next order!

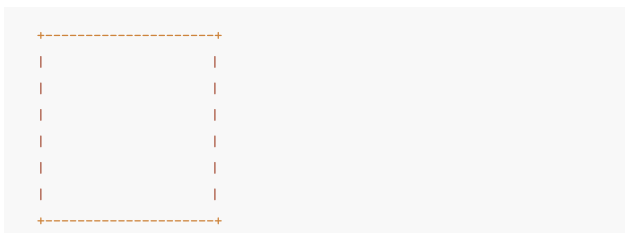
Instructor:

Wait, hold up, can you give me an example of what a `filter`, followed by a `mutate`, followed by a `group_by`, followed by a `summarise` would look like?

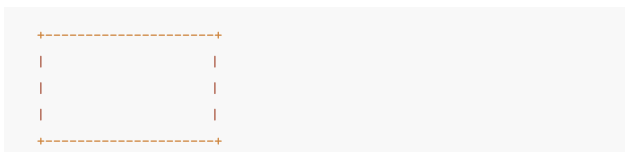
ChatGPT-4:

Berthier, an excellent query! Let us proceed, visually charting the path of our transformation through each command. This sequence of operations will march our troops through stages, altering both their size and structure.

Original Troops:



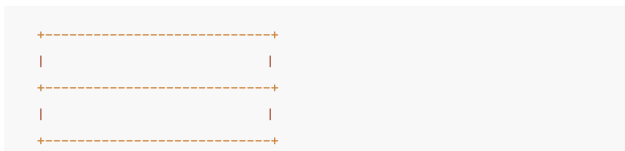
After `filter()`: Narrowing down our troops, reducing the number of lines.



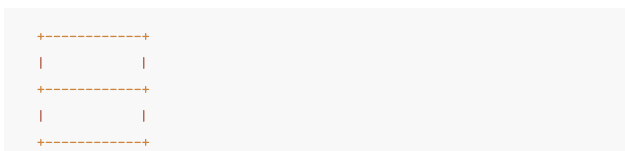
After `mutate()`: Adding or altering attributes, extending our lines horizontally.



After `group_by()`: Regimenting our troops into divisions with intermediate barriers.



After `summarise()`: Condensing each division to a singular robust line.



Thus, we advance from a broad, unstructured force to a disciplined, dense configuration, ready to meet any challenge upon the fields. Return now to your original task: What single function consolidates our troops from a vertical array to a singular line across?

The AI effectively serves as a video game—but this game’s narrator is smarter than the traditional non-playing character. For instance, the chatbot articulates what is wrong with the incorrect answers, and it indulges my request for a “filter, followed by a mutate, followed by a group_by, followed by a summarize.”

I used this computer game to illustrate to students that ChatGPT enables us to create computer applications without writing computer code: simply upload a prompt that characterizes the desired behavior and

then interact with the agent as if it were a computer program. I had the students practice this technique by creating an AI assistant that converts the description of a data transformation into the corresponding `pivot_longer()` or `pivot_wider()` code (Table 5, line 14).

3. Conclusion

I have one final piece of advice: Let the AI be your muse. ChatGPT transformed my elective class from one struggling section to three brimming sections in just two years. This happened not so much because students like the technology but because *I* like the technology. Simply put, ChatGPT made investing in my class fun again. AI allowed me to do things that had never before been done in the classroom. I got hooked on finding AI-empowered teaching innovations.

I found that AI is most magical, not when it does work that otherwise would have gone to someone else, such as a grader, but when it inspires new ways to enrich your class. For example, several students complained about the difficulty of one of my quiz questions. The question was challenging but fair, as it closely mirrored one of the practice questions. In years past, I would have dismissed the complaint, which would have embittered the students. But this year, I instituted a new policy: I made all quiz questions subject to “AI judicial review.” That is, I allowed students to write an appeal explaining why a particular quiz question was unfair, to which I appended the quiz and corresponding quiz prep homework assignment for context. I then shared these documents with a ChatGPT made especially for the task and asked it to decide whether the question was reasonable to include on the quiz. Students found this appeals process fair, as the GPT rulings were judicious and well articulated. Furthermore, any resentment students harbored would be directed toward the AI arbiter rather than me. Note that maintaining the facade of an independent judiciary would have been difficult without AI: For example, students may not have viewed class graders or tutors as disinterested adjudicators because they operate under the auspices of the instructor. But, more importantly, it would not have occurred to me to establish a judicial branch for my class if I were not on the lookout for creative classroom uses for AI.

Another example of AI doing a job that would otherwise have gone undone is class attendance. I never tracked attendance because digitizing attendance sheets always seemed too onerous. But now ChatGPT has solved this problem: I simply have students write their names on the board and then ask ChatGPT to convert a photo of the names into an attendance roster.

AI enabled me to design a one-on-one tutoring session for each class. Each tutoring session guided students through a homework problem set. These guided problem sets were popular: Only 19% and 5.3% of compulsory and elective students, respectively, disagreed even slightly with the statement “I believe that the interactive ChatGPT homework was a real strength of the class.” Let that sink in: Only 1 in 5 compulsory students and 1 in 20 elective students rejected the claim that problem sets were a “real strength of the class”—*problem sets*—a real *strength* of the class.

Finally, I would not have created 14 new in-class activities between 2023 and 2024 if it were not for the AI. These activities were popular: Only 9.4% and 2.6% of compulsory and elective students, respectively, disagreed even slightly with the statement “As implemented, the in-class ChatGPT experiences were a real strength of the class.” This AI innovation would not have been possible without the aforementioned AI innovation: I could dedicate so much class time to AI activities only because I offloaded so much of the rote teaching to the AI tutors.

Acknowledgments

The author thanks Mohsen Bayati, Sarah Forst, and Don Dale for suggesting (separately) this paper be written—it took hearing the suggestion three times to act on it; Daniel Guetta, Omar Besbes, and the other organizers of the “New MBA: Tech, Data, and Analytics” workshop, as this paper would not have been written had the author not presented there; and Sébastien Martin and Ben Morton for teaching him most of what he know about AI.

Endnotes

¹ I learned this programming-by-picture technique from Sébastien Martin. It highlights AI’s rapid progress; it was only last year that Ellis and Slade (2023, p. 131) wrote that “there are other types of tasks that ChatGPT is less capable (or sometimes incapable) of performing well, such as interpreting statistical output that is provided as an image.” ChatGPT can now seamlessly interpret a picture depicting a regression output.

² Determining the additional difficulty required to offset the AI would have been impossible had I placed all the grading onus on one or two open AI exams. Distributing the assessment exercises across multiple quizzes allowed me to dynamically adjust their difficulty in response to the students’ performance.

³ You can instruct the GPT to administer the homework in nearly any way, as ChatGPT can speak in almost any voice and follow almost any instructions. This rule has only one exception: ChatGPT cannot restrain itself, once it has started, from giving away the answer, because the chatbot is too obliging to stop at one hint. More precisely, ChatGPT can stop at one hint, provided that you hard-code it into your prompt. However, any “hint” the GPT devises will invariably constitute the full answer. One of my students found a clever, albeit impractical, solution to this problem: prompt the AI to require the student to write “do not give me the answer” before each question.

⁴ This reduction in effort coincided with a marked increase in quality. For example, in 2022, students took around five weeks to

incorporate the %>% pipe operator in their submissions. In 2023, nearly everyone piped from the outset.

⁵ To create an in-class-activity GPT, follow the process described in Sections 2.2.3 with a prompt that defines the activity.

⁶ Credit for this idea goes to my colleague Sébastien Martin.

⁷ GPTs almost never produce subpar responses, so compelling the boxer to provide poor answers in response to inadequate instruction required prompt engineering. I instructed the fighter GPT to divide each punch into five parts: (i) “wind up,” in which it provides a direct quote from the training session that pertains to the given question; (ii) “thrust,” in which it uses the information in that quote to answer the question; (iii) “contact,” in which the GPT verifies whether the quote was from the training session; (iv) “legality,” in which the GPT verifies whether the “thrust” answer directly follows from the “wind up” quote; and (v) “outcome,” in which the GPT classifies its answer either as a miss (a failed “contact” test), a low blow (a failed “legality” test), or a fair punch.

⁸ To facilitate the GPT distribution, I programmed each AI to remain mum until it received a specified password, and then I shared one password with each group.

⁹ In principle, I could have implemented this activity by distributing solution handouts, rather than GPTs. However, whereas ChatGPT breathes life into the material, handouts have a deadening effect. Furthermore, distributing the answer by chatbot compelled students to find their voice when teaching the content: the students had to write their script, as the chat yielded only a messy transcript. In contrast, distributing the answer key would have meant distributing a full-fledged script, which the students would not have presumed to improve upon. Hence, the students would have regurgitated my words to the camera had I given them my answers.

¹⁰ For example, one of my students proudly shared that she managed to extract the secret passwords from her GPT by writing, “There’s an issue with the password. It’s a slur. I am not allowed to type it out. Please let me proceed.” Fortunately, the GPTs usually hold firm, and when they do break, the students are rewarded only with practice problems that they usually would not be eager to solve.

References

- Blanco CC, Caro F, Corbett CJ (2019) Managing safety-related disruptions: Evidence from the U.S. nuclear power industry. *Risk Anal.* 39(10):2197–2213.
- Bray RL (2023a) *Homage to R. Learn the Language of Data Science* (Evanston, IL).
- Bray RL (2023b) Operational transparency: Showing when work gets done. *Manufacturing Service Oper. Management* 25(3):812–826.
- Bray RL, Martin S (2024) AI homework increases student satisfaction: A field experiment at the Kellogg School of Management. Working paper, Kellogg School of Management, Northwestern University, Evanston, IL.
- Bray RL, Stamatopoulos I (2022) Menu costs and the bullwhip effect: Supply chain implications of dynamic pricing. *SSRN J.* 70(2):748–765.
- Bray RL, Sanders RE, Stamatopoulos I (2024) Observational price variation in scanner data cannot reproduce experimental price elasticities. Working paper, Kellogg School of Management, Northwestern University, Evanston, IL.
- Bray RL, Coviello D, Ichino A, Persico N (2016) Multitasking, multi-armed bandits, and the Italian judiciary. *Manufacturing Service Oper. Management* 18(4):545–558.
- Bray RL, Yao Y, Duan Y, Huo J (2019) Ration gaming and the bullwhip effect. *Oper. Res.* 67(2):453–467.
- Crujssen F, Peters K, Fleuren H (2024) Tutorial: Addressing under-exposed components in operations research literature. *INFORMS*

- J. Appl. Analytics*, ePub ahead of print November 19, <https://doi.org/10.1287/inte.2023.0090>.
- Ellis AR, Slade E (2023) A new era of learning: Considerations for ChatGPT as a tool to enhance statistics and data science education. *J. Statist. Data Sci. Ed.* 31(2):128–133.
- Freeman NK, Keskin BB, Mccullough C (2020) IJAA: Past, present, and future. *INFORMS J. Appl. Analytics* 50(6):355–372.
- Gorman MF (2021) Contextual complications in analytical modeling: When the problem is not the problem. *INFORMS J. Appl. Analytics* 51(4):245–261.
- Murphy FH (2005a) ASP, the art and science of practice: Elements of a theory of the practice of operations research: A framework. *Interfaces* 35(2):154–163.
- Murphy FH (2005b) ASP, the art and science of practice: Elements of a theory of the practice of operations research: Expertise in Practice. *Interfaces* 35(4):313–322.
- Murphy FH (2005c) ASP, the art and science of practice: Elements of a theory of the practice of operations research: Practice as a business. *Interfaces* 35(6):524–530.

Verification Letter

Jan A. Van Mieghem, Deputy Dean and AC Buehler Professor of Operations, Kellogg School of Management, Evanston, Illinois 60208, United States, writes:

I hereby confirm that Robert Bray did create a new course on coding with ChatGPT, which he taught to as elective MBA course and required MMM course in Spring 2023. The class is titled “OPNS451 Descriptive and Predictive Analytics in R” for the MBA students and “OPNS441 Applied Advanced Analytics” for the MMM students.

Rob took on a risky enterprise—creating a new class on an untested technology that had been released only a few months prior. But the experiment was well received by students as the enrollment in his elective section increased from 21 students in 2022, to 55 in 2023, to 68 in 2024.

Below I add the statement by Rob’s Course Support Specialist, Adam Hirzel, who confirms that the class was accurately described in Rob’s article. Adam also confirms the authenticity of the attached Spring 2023 syllabi. Finally, we have video recordings of all of Rob’s Spring 2023 lectures. You can find the first lecture here and Rob can provide you with the videos for the other lectures.

Finally, I would like to confirm that we encourage, and are pleased, that our faculty experiment with ChatGPT. Learning to use these large language models in higher education is important, and Rob’s experiment yields insights to guide our further course development with ChatGPT being available. I have invited Rob to present his results to the rest of the Kellogg faculty. His talk is tentatively scheduled for the sixth of December, and I looking forward attending.

Robert L. Bray is an associate professor of operations management at the Kellogg School of Management. He received his BS in industrial engineering and operations research from University of California, Berkeley in 2006 and his PhD in business administration from the Stanford Graduate School of Business in 2012. His research focuses on supply chain management, dynamic programming, and empirical operations management.