

Incorporating ChatGPT into a Financial Data Science Course with Python Programming

Yang Liu*

James Madison University

Laura K. Miller[†]

James Madison University

Xu Niu[‡]

James Madison University

We would like to thank the students and class participants of FIN 434 (Fundamentals of Financial Data Science) at James Madison University in Fall 2022 and Spring 2023 for their valuable suggestions and feedback. We would also like to acknowledge the generous financial support on this project from the Learning Center's "Learning Laboratories Research Fund" at James Madison University.

*Professor Yang Liu, PhD, is Assistant Professor at the Learning Centers and Department of Psychology at James Madison University, Harrisonburg, VA 22807, E-mail: liu38yx@jmu.edu.

[†]Professor Laura K. Miller, PhD, is Assistant Professor of Writing, Rhetoric and Technical Communication and Executive Director of the Learning Centers at James Madison University, Harrisonburg, VA 22807, E-mail: mille4lk@jmu.edu.

[‡]Corresponding author, Professor Xu Niu, PhD, is Assistant Professor at Department of Finance and Business Law at James Madison University, Harrisonburg, VA 22807, E-mail: niuxx@jmu.edu.

Incorporating ChatGPT into a Financial Data Science Course with Python Programming

Abstract

ChatGPT's artificial intelligence (AI) has generated considerable debate and even panic in higher education. However, AI is an unavoidable trend in education that has the potential to subvert many courses' pedagogical structure and methods. One such course is financial data science with Python programming, a course that is in high demand among finance majors. Incorporating ChatGPT into such a course can solve long-standing challenges and difficulties that instructors and students are facing, and it can shift the focus of the course from tedious Python coding back to financial data interpretations and real-world applications. This paper represents a first attempt at welcoming ChatGPT into financial education, by focusing on the pedagogical innovation of a financial data science course with Python programming. In particular, the paper provides specific examples of potential projects, with other insights and suggestions for using ChatGPT in data science courses for finance majors.

JEL classification: A22, C80, I23

Keywords: ChatGPT, Financial Education, Data Science, Python

1. Introduction

Financial data science courses in higher education have been rapidly changing alongside the evolution of technology. Simultaneously, big data in the era of artificial intelligence (AI) have shaped the finance industry quickly, which has steadily increased the demands on the financial data science workforce. In turn, financial education must evolve to meet industry demands and to prepare students to succeed in this rapidly evolving field, without compromising essential foundational instruction.

Due to technological trends in the field, financial data science courses for undergraduate finance majors typically focus on Python programming. In fact, many business schools are launching new programs or concentrations in FinTech and Financial Analytics, such as the prestigious business schools at New York University, Massachusetts Institute of Technology, and Columbia University.

However, instructors and students in financial data science courses with Python programming face several challenges. First, finance majors typically lack the preparation and prerequisites for data analysis, computer skills, and programming exposure to modern computer languages. Second, finance majors are not usually motivated to learn the details of Python coding, starting from syntax. Instead, as applied scientists, finance majors want to see how these Python programs can help solve real problems in finance. We notice that existing courses, webinar trainings, and textbooks on data science and Python programming do not target finance majors; instead, their purpose aligns with the needs of computer science or computer information systems majors. As a result, it is difficult for finance majors to identify direct financial applications and understand how Python programs can be linked to other theories they have been learning from other finance courses, such as corporate finance and investments. Third, given the above two challenges, it is almost impossible to reach the goals of a financial data science course in one semester. The time-consuming preparations regarding basic Python coding leave only a few classes at the end of the semester to explore applications in finance. Finally, the learning outcomes of such a course in financial data science and Python programming

are vague. Through a one-semester course, it is impossible to train an average finance major to become a sophisticated programmer in Python. In fact, training finance majors to become computer programmers should never be the learning objective because cutting-edge finance majors on the job market should not be competing with computer science majors with Python programming skills. Instead, finance majors should treat Python (or any computer language or statistical software) as a tool for understanding and solving problems in the real financial world. Doing so enables finance majors to communicate with other team members, who are Python experts, and to provide primary expertise in finance.

In the context of these challenges, Open AI published ChatGPT in November, 2022. ChatGPT is a large natural language processing model trained by OpenAI based on a massive dataset of internet-based text before September 2021 (OpenAI, 2022). In March 2023, OpenAI also released a new large multimodal model named GPT-4 as an enhanced version of ChatGPT (OpenAI, 2023). Both ChatGPT and GPT-4 produce text response based on a user's prompt. ChatGPT is free and only receives text prompts, while GPT-4 is only available in the paid version and can receive both text and image prompts.

ChatGPT has created new challenges in higher education and has provoked a broad discussion on how to include this innovation in higher education (Dowling & Lucey, 2023; Huang, 2023; Lund & Wang, 2023). For instance, Rudolph et al. (2023) indicates that ChatGPT raises questions about plagiarism, which need to be dealt with carefully. Some universities, like the University of Vermont (2023), have begun to revise their plagiarism rules. One of the top universities in France, Paris Institute of Political Studies (SciencesPo), has even banned the use of ChatGPT (Sciences Po, 2023). Although current large natural language processing models, such as ChatGPT, have unignorable limitations, including the tendency to hallucinate, produce harmful content, amplify biases, etc., the advances in large language models have indubitable potential for vast social impact, including higher

education (Tamkin et al., 2021; OpenAI, 2023a). Yet, the application of artificial intelligence in financial data science education has not been fully explored. Educators must admit that AI not only has the potential to change traditional pedagogy, but it could even subvert the current structure and instructional methods of certain courses.

One such course is financial data science with Python programming, a course in high demand among finance majors. Incorporating AI tools, such as ChatGPT, into this course using Python programming could provide a promising or even an ultimate solution to the above-mentioned challenges in teaching financial data science courses.

First, with the help of ChatGPT, instructors can free finance majors from the tedious and time-consuming preparations on writing Python codes and instead focus on reading and understanding Python codes. Second, the instructional focus can shift from teaching students how to write executable Python codes to writing effective prompts with applications in financial data science. Finally, a financial data science course with Python programming can once again become a finance course by focusing on financial theories and applications, instead of becoming a computer programming course with minimal examples in finance.

To the best of our knowledge, this paper represents the first attempt at exploring the possibility of incorporating ChatGPT into financial education, especially the course in financial data science with Python programming. Given ChatGPT's ability to convert among different computer languages, we use Python as an example to illustrate how educators could use ChatGPT as a tool to teach financial data science.

The remainder of this paper proceeds as follows: Section 2 reviews the literature on ChatGPT in finance and education. Section 3 introduces several examples of in-class projects for using ChatGPT in a financial data science course with Python programming, and it provides more suggestions on

potential projects that are suitable for learning with the help of ChatGPT. Section 4 concludes by reviewing the overarching assertions in this paper.

2. Literature Review

The finance industry is undergoing a transformation due to rapid changes in technology, such as advances in artificial intelligence and big data, leading to opportunities for growth and innovation in financial education (Alt et al., 2018; Arner et al., 2015; Davenport & Ronanki, 2018; Schueffel, 2016; Suryono et al., 2020). For example, Arner et al. (2015) review the long history during which finance and technology have evolved together, and they discuss challenges in balancing the potential benefits of innovation with the possible risks of approaches. Alt et al. (2018) assert that FinTech has disruptive potential for the financial industry, including the convergence of multiple technologies, the customer-orientation approach, and the innovation/entrepreneurial spirit. They highlight the dominance of the banking sector within the FinTech domains but call for more research on other FinTech domains. Consequently, fintech education is essential to equip professionals with necessary skills and knowledge to excel in this rapidly evolving field.

Financial data science education has also become increasingly important due to the rise of big data in the finance industry. Several studies have explored the current state and future prospects of financial data science education (Alshater, 2022; Aydın & Karaarslan, 2023; Birjali et al., 2021; Dowling & Lucey, 2023; Fisher et al., 2016; Kang et al., 2020; Olan et al., 2022; Xing et al., 2018; Yan, 2018). For example, Yan (2018) identifies big data and programming as two trends in job advertisements at business schools. They collected job advertisements from 23 business schools and found that there is significant demand for fintech education among students and professionals. Yan (2018) also indicates that many economics, finance, and accounting data are available to students as

public data, such as the Federal Reserve, US. Department of Treasure, Marketwatch, and more. They also suggest business majors should master at least one programming language, such as Python.

The finance industry is also undergoing a transformation due to the incorporation of AI, which has opened up new opportunities for data-driven decision-making and innovation (Bussmann et al., 2020; Davenport & Ronanki, 2018; Lu, 2019; Rudin & Radin, 2019). For example, Bussmann et al. (2020) propose an AI model to assess credit risk in the context of peer-to-peer lending. Lu (2019) discusses the significant role that AI played in industrial development from 1961 to 2018, especially in integrating emerging technologies such as graphic processing units, cloud computing, and blockchain. Moreover, this study emphasizes the unquestionable value of AI as an assistant in a wide range of industries. Accordingly, there is a growing interest in AI and financial data science education to equip professionals with the necessary skills and knowledge to excel in this rapidly evolving field.

One of the examples and applications of AI is Large Language Models (LLMs) based on Natural Language Processing (NLP), which has significant implications for financial data science education. In recent years, researchers have explored the use of NLP techniques for processing and analyzing financial data in varied areas in finance (Alshater, 2022; Aydın & Karaarslan, 2023; Birjali et al., 2021; Dowling & Lucey, 2023; Kang et al., 2020; Xing et al., 2018). For example, Aydın and Karaarslan (2023) use ChatGPT as an example of Generative AI to discuss the technical and structural fundamentals, competitors, and user expectations. They find that although ChatGPT fabricates some citations when writing a literature review on given topics, Generative AI might change the way we do things in many fields. Kang et al. (2020) provides a comprehensive review of the application of the UT Dallas List of 24 Leading Business Journals that employ NLP as their focal analytical technique to elucidate how textual data can be harnessed for advancing management theories across multiple disciplines. The study discusses the managerial and technological challenges associated with the application of LLMs in management research.

In summary, the literature in finance has reached a consensus that fintech education is a rapidly evolving field that requires innovative approaches and resources to keep pace with industry demands. Therefore, effective strategies for financial education need to be identified. The literature also suggests that AI and financial data science education are critical for future growth and innovation in the finance industry. It is necessary to identify effective strategies for AI and financial data science education and evaluate their impact on industry and society. Education must also prepare students to navigate the ethical implications of AI in finance to ensure responsible and sustainable use of AI in finance.

3. Examples of Python Projects with ChatGPT

As a first attempt at welcoming ChatGPT into financial education, this paper focuses on the pedagogical innovation of a financial data science course with Python programming. A typical such course is one semester long and is geared toward junior or senior finance majors at the undergraduate level. Since existing textbooks on data science and Python programming do not target finance majors, such a course is not oriented from a required textbook; instead, the course is project-based and includes a sequence of in-class projects and take-home projects as assignments.

In this section, we first present two typical examples of in-class projects that would benefit from interacting with ChatGPT. The first project uses the Monte Carlo simulation to calculate the weights for an optimal portfolio given a list of stocks in a certain sample period. The second project is intended to run a sequence of time-series regressions for a list of stocks to find the factor loadings of the Fama-French Three-Factor Model. At the end of the section, we provide a list of other potential projects that can be suitable for using ChatGPT in a financial data science course with Python programming. All potential projects are well-suited for such a course (either as in-class projects or as take-home assignments), and all projects have been tested by the authors of this article.

3.1 Example Project 1: Monte Carlo Simulation and Optimal Portfolio

An optimal portfolio given a list of stocks can be achieved by three equivalent methods: (a) maximizing an investor's expected return for a given level of risk, (b) minimizing the risk for a given level of expected return, or (c) maximizing the Sharpe ratio of the portfolio. The construction of an optimal portfolio is a fundamental concept in modern portfolio theory and is used extensively in the field of finance. A typical undergraduate-level course in “Investments” or “Portfolio Management” for finance majors will cover such a topic in great detail from a theoretical point of view. One approach to creating an optimal portfolio is to use the Monte Carlo simulation. The basic idea is to generate a large number of random portfolios and then choose the one that has the best risk-return tradeoff (i.e., choose the portfolio that gives the highest possible Sharpe ratio).

For this particular in-class project, instructors typically need to start with an overview of related concepts with a focus on financial concepts, such as optimal portfolio, Sharpe ratio, efficient frontier, and Capital Asset Pricing Model (CAPM). At the onset, students often view these concepts as overly theoretical and do not see their applications with real financial data. Therefore, we assign this particular project to help students understand how these concepts directly apply to the real financial world. Instead of the traditional approach of instruction beginning with Python coding, we initiate our project with help from ChatGPT. We have found that incorporating ChatGPT into the project frees students from tedious and complicated Python programming, and the most important part of learning becomes understanding the theoretical concepts needed to communicate with ChatGPT effectively.

The first step of using ChatGPT is to develop a prompt (i.e., input from users) and then react according to outputs from ChatGPT for the following procedures. We emphasize that it is likely that ChatGPT's first response might not meet our requirements and will require follow up with modified questions to get expected answers and Python codes from ChatGPT. Next, the instructor would help

students understand that the logic of calculating weights for an optimal portfolio requires downloading of returns for a given list of stocks in a certain sample period. Moreover, the instructor can choose to use monthly returns data from Yahoo Finance for this project, and let students be aware of the data availability from Yahoo Finance (and probably other data resources, such as CRSP). We note to students that in order to communicate with ChatGPT effectively, they only need to be aware of the data available from Yahoo Finance and be able to understand the data structure.

Table 1 presents the dialog with ChatGPT requesting a Python code for downloading data from Yahoo Finance. For illustration purposes, in Table 1 (and all similar tables hereafter), we present “Input from the user of ChatGPT” in Panel A and corresponding “Output from ChatGPT” in Panel B. The screenshots for all dialogs with ChatGPT are available from the authors upon request. We note that ChatGPT will generate a slightly different answer even if the question is from the same user. Therefore, the outputs from ChatGPT in our paper might not be exactly replicated either by other researchers, educators, or students in class. Instead of focusing on the direct replication of each project, we describe the logic of dialog with ChatGPT and investigate how this logic can be applied in the financial data science course with Python programming. Since the Python codes generated by ChatGPT are too long to present in the table, we report all such codes in the Appendix.

Table 1: Dialog with ChatGPT – Download Data from Yahoo Finance

Panel A: Input from the user of ChatGPT
Write a python code that can directly download monthly stock return data for a list of stocks from Yahoo Finance.

Panel B: Output from ChatGPT
To download monthly stock return data for a list of stocks from Yahoo Finance, we can use the <code>`pandas-datareader`</code> library. The following Python code can be used to download the monthly returns for a list of stocks: (Note: ChatGPT generates a Python code here in a stand-alone box. Since the Python code generated by ChatGPT is too long to present in the table, we report all such codes in the Appendix.) In this code, we first define a list of stocks (<code>`stocks_list`</code>) and the start and end dates for the data (<code>`start_date`</code> and <code>`end_date`</code>). We then use the <code>`pdr.get_data_yahoo()`</code> function to download the data from Yahoo Finance, specifying the list of stocks, start and end dates, and an interval of 'm' for monthly data. Next, we extract the Adj Close prices for each stock from the downloaded data, and use the <code>`pct_change()`</code> function to calculate the monthly returns for each stock. Finally, we print the first few rows of the returns dataframe to verify that the data was downloaded and processed correctly.

Instead of pausing our logic and stream of thoughts by reading through the Python code, we tell students that we shall first assume the Python code generated by ChatGPT is executable and correct (which it is often not). After we receive all Python codes as expected from a sequence of queries in ChatGPT, we have students merge all Python codes together and develop a final version of Python code in a single file before debugging and executing the code. Of course, a quick browse-through of the Python code is necessary and very helpful in the following steps.

By quickly reading through ChatGPT's response, as presented in Table 1, we notice that ChatGPT not only provides a Python code in a clean format (with comments alongside code), but more importantly, ChatGPT provides details and the logic behind the generated Python code, which are extremely helpful for beginning users of Python who do not have much previous experience and exposure to computer programming. Notably, most students in a financial data science course for finance majors should be considered this type of Python user.

Next, we notice that ChatGPT recommends using the ``pandas-datareader`` library for this purpose. A more sophisticated financial data scientist might be aware of another widely used library of

``yfinance`` for accessing data directly from Yahoo Finance. We want to introduce this new library to the class and therefore send our modified question and request to ChatGPT, forcing the generated Python code to use the ``yfinance`` library instead. The dialog with ChatGPT for this purpose is presented in Table 2.

Table 2: Dialog with ChatGPT – Download Data from Yahoo Finance (Modified)

Panel A: Input from the user of ChatGPT
Write a python code that can directly download monthly stock return data for a list of stocks from Yahoo Finance, using the <code>yfinance</code> library in Python.

Panel B: Output from ChatGPT
To download monthly stock return data for a list of stocks from Yahoo Finance using the <code>`yfinance`</code> library in Python, we can use the following code: (Note: ChatGPT generates a Python code here in a stand-alone box. Since the Python code generated by ChatGPT is too long to present in the table, we report all such codes in the Appendix.) In this code, we first import the <code>`yfinance`</code> library using the <code>`yf`</code> alias. We then define a list of stocks (<code>stocks_list`</code>) and the start and end dates for the data (<code>start_date`</code> and <code>end_date`</code>). Next, we use the <code>`yf.download()`</code> function to download the data from Yahoo Finance, specifying the list of stocks, start and end dates, and an interval of <code>'1mo'</code> for monthly data. We then extract the Adj Close prices for each stock from the downloaded data, and use the <code>`pct_change()`</code> function to calculate the monthly returns for each stock. Finally, we print the first few rows of the returns dataframe to verify that the data was downloaded and processed correctly.

Now, assuming the datasets with monthly stock returns are ready to use, we proceed to request that ChatGPT generate a python code that can calculate the optimal portfolio given monthly return data for a list of stocks. The dialog with ChatGPT for this purpose is presented in Table 3.

Table 3: Dialog with ChatGPT – Calculate Optimal Portfolio

Panel A: Input from the user of ChatGPT
Write a python code that can calculate the optimal portfolio given monthly return data for a list of stocks.

Panel B: Output from ChatGPT
<p>To calculate the optimal portfolio given monthly return data for a list of stocks, we can use the `scipy` library in Python to perform a mean-variance optimization. Here is an example code:</p> <p>(Note: ChatGPT generates a Python code here in a stand-alone box. Since the Python code generated by ChatGPT is too long to present in the table, we report all such codes in the Appendix.)</p> <p>In this code, we first define a list of stocks (`stocks_list`) and the start and end dates for the data (`start_date` and `end_date`). We then download the monthly return data for each stock using the Yahoo Finance API and concatenate them into a single dataframe.</p> <p>Next, we define the objective function for the mean-variance optimization, which calculates the variance of the portfolio given a set of weights and the covariance matrix of the returns. We also define a constraint for the sum of the weights to equal 1.</p> <p>We then define initial weights for the optimization and bounds for the weights to be between 0 and 1. We use the `minimize()` function from `scipy.optimize` to perform the mean-variance optimization, specifying the objective function, initial weights, method, bounds, and constraint.</p> <p>Finally, we print the optimal weights.</p>

According to the response from ChatGPT in Table 3, we notice that the ChatGPT-proposed Python code calculates the weight for the optimal portfolio using the method of a mean-variance optimization, which is not our intended method of the Monte Carlo simulation. We note that this is very common when communicating with ChatGPT, and the solution is easy and straightforward: We send another round of requests to ChatGPT with a similar question, but this time, we specifically require ChatGPT to use “simulation” to generate Python codes. The dialog with ChatGPT for this round of communication is presented in Table 4.

Table 4: Dialog with ChatGPT – Calculate Optimal Portfolio (Simulation)

Panel A: Input from the user of ChatGPT
Write a python code that uses simulation to calculate the optimal portfolio given monthly return data for a list of stocks.

Panel B: Output from ChatGPT
<p>To use simulation to calculate the optimal portfolio given monthly return data for a list of stocks, we can generate random portfolios with different weights and simulate their returns to find the portfolio with the highest Sharpe ratio. Here is an example code:</p> <p>(Note: ChatGPT generates a Python code here in a stand-alone box. Since the Python code generated by ChatGPT is too long to present in the table, we report all such codes in the Appendix.)</p> <p>In this code, we first download the monthly return data for each stock using the `yfinance` library and calculate the returns. We then define the number of simulations and the number of portfolios to generate for each simulation.</p> <p>We define a function to generate a random portfolio of weights, which generates a set of random weights and normalizes them so that they sum to 1. We also define a function to simulate the returns of a portfolio given a set of weights, which calculates the expected return, volatility, and Sharpe ratio of the portfolio.</p> <p>We then generate multiple sets of random portfolios and simulate their returns to find the portfolio with the highest Sharpe</p>

Interestingly, we notice that ChatGPT first downloaded the monthly return data from Yahoo Finance using the `yfinance` library. We have tested the behavior and response from ChatGPT in a “New chat” window, and the results are different and do not use the `yfinance` library, as in the first request. Therefore, it is reasonable to assume that ChatGPT’s algorithm links the current request with the previous requests in the same chatting thread to optimize the user’s experience. We appreciate this feature in ChatGPT because it creates a more natural, efficient conversational experience.

By merging all the related Python codes generated by ChatGPT (i.e., downloading data and using simulation to calculate weights for the optimal portfolio), we can easily develop a single Python code for the entire project. Usually, with only a few rounds of potential debugging, the full Python code becomes executable and fulfills our requirements and purposes. In fact, since we organized the questions in the most effective way of communicating with ChatGPT, the last Python code generated by ChatGPT is almost ready to use directly for our purpose.

Next and most importantly, we need to read and understand the Python codes generated by ChatGPT line by line and identify any errors. We note that it is rare that ChatGPT-generated Python code has any “syntax error”; instead, the focus should be on the theoretical side (e.g., whether the Python code defines the Sharpe ratio in a correct manner and whether the code can identify the portfolio with maximized Sharpe ratio). We emphasize this in class. According to students’ feedback, they start to realize the importance of understanding fundamental finance theories because other tedious and complicated work, such as Python coding, can be easily handled by modern tools like ChatGPT.

As an example, we assume the list of stocks under consideration include Apple Inc. (AAPL), Alphabet Inc. (GOOGL), and Amazon.com Inc. (AMZN), for the sample period from January 1, 2016 to December 31, 2021. The calculated weights for the optimal portfolio using the merged Python codes generated by ChatGPT are presented in Table 5.

Table 5: Results Based on ChatGPT Generated Python Code – Optimal Portfolio

Panel A: Weights for Optimal Portfolio (Monte-Carlo Simulation Method)	
Name of Stock (Ticker)	Weights
Apple Inc. (AAPL)	43.08%
Alphabet Inc. (GOOGL)	29.76%
Amazon.com Inc. (AMZN)	27.16%

Panel B: Summary Statistics of Optimal Portfolio	
Expected Return	33.73%
Volatility (Standard Deviation)	21.75%

It is natural for students to ask a follow-up question about the differences between the method of mean-variance optimization and the method of simulation in calculating optimal portfolio weights. In fact, we show the class that ChatGPT can also help us design such questions, although we still need instructor expertise and guidance because ChatGPT’s responses to such questions are not always correct and reliable. The dialog with ChatGPT about the advantages and disadvantages of the methods

of mean-variance optimization versus simulation is presented in Table 6. We conclude this project by noting and detailing the response from ChatGPT on this comparison.

Table 6: Dialog with ChatGPT – Compare Different Methods

Panel A: Input from the user of ChatGPT
Compared to the method of mean-variance optimization, what are the advantages of using the method of simulation to calculate optimal portfolio?
Panel B: Output from ChatGPT
The mean-variance optimization method for calculating optimal portfolio weights relies on assumptions about the distributions of the assets' returns and correlations, which may not hold in reality. In contrast, simulation-based approaches do not make any assumptions about the distribution of asset returns or correlations and are therefore more robust to model misspecification.
Furthermore, mean-variance optimization requires the estimation of the covariance matrix of asset returns, which can be noisy and may lead to instability in the portfolio weights. In contrast, simulation-based approaches only require the calculation of the mean and standard deviation of the portfolio returns, which are more stable and less sensitive to outliers.
Simulation-based approaches also allow for the incorporation of non-normal distributions, which may be important in practice since stock returns often exhibit non-normal behavior such as skewness and kurtosis.
Finally, simulation-based approaches can easily incorporate constraints on the portfolio weights, such as upper and lower bounds, integer constraints, and exposure constraints, which can be challenging to handle in mean-variance optimization.
Overall, simulation-based approaches can provide a more flexible and robust alternative to mean-variance optimization for calculating optimal portfolio weights.

Finally, we conclude this in-class project by noting that as applied scientists, such as finance majors, our focus should always be on understanding the theories in finance. While previously there were many distractions in reaching this goal because finance majors often devote too much time to learning Python programming, nowadays with the help of ChatGPT, the focus is back on financial content. According to our experience in the classroom, students strongly agree with this notion and more than ever understand the importance of theories from fundamental courses in finance.

3.2 Example Project 2: Regression Analysis and Fama-French Three-Factor Model

Our second example project is intended to run a sequence of time-series regressions for a list of stocks to find the factor loadings of the Fama-French Three-Factor Model (Fama & French, 1992). Ever since the first establishment of this model, the Fama-French Three-Factor Model has established its importance. It soon became a widely used asset pricing model that has significant implications for financial theory and practice. Nowadays, almost every finance textbook (especially those in Investments and Asset Pricing) mentions this important model. It is therefore of critical importance for finance majors to have a full and sophisticated understanding of the model by doing time-series regressions of Fama-French Three-Factor Model themselves. Therefore, a typical course in financial data science should cover this material.

Similar to the previous in-class project, we start this project by reviewing the related concepts in financial theory and financial econometrics, such as the Fama-French Three-Factor Model, regression analysis, and the different structures of financial data (time-series, cross-sectional, and panel data). Moreover, we introduce students to the data available from Wharton Research Data Services (WRDS), especially the Center for Research in Security Prices (CRSP) and the Data Library on Professor Ken French's webpage (which has been integrated into WRDS). Due to the recent attempt of launching Two-Factor Authentication (2FA) at WRDS in 2023, we instruct all students to register for their own individual accounts because the original instructional WRDS account (i.e., "Shared Account for Class" at WRDS) does not support direct access databases from Python program.

Next, an important starting point is finding a well-designed question or prompt to communicate with ChatGPT. Table 7 displays the dialog with ChatGPT requesting a Python code for downloading data from WRDS.

Table 7: Dialog with ChatGPT – Download Data from WRDS

Panel A: Input from the user of ChatGPT
Write a python code that can directly download datasets from WRDS. The code needs to download daily return data for a list of stocks (available in CRSP) and Fama-French Factors. Finally, merge the downloaded datasets.

Panel B: Output from ChatGPT
To download datasets from WRDS, you can use the <code>wrds</code> Python package. Here is an example code that downloads daily returns data for a list of stocks available in CRSP and Fama-French factors, and merges the downloaded datasets: (Note: ChatGPT generates a Python code here in a stand-alone box. Since the Python code generated by ChatGPT is too long to present in the table, we report all such codes in the Appendix.) In this code, we first establish a connection to WRDS using the <code>wrds.Connection()</code> function. We then specify the list of stocks and Fama-French factors we want to download. To download the daily returns data for the specified stocks, we construct a SQL query that selects the date, permno (CRSP's unique identifier for each stock), and daily return (<code>ret</code>) from the <code>crsp.msfc</code> table, filtering by the specified list of tickers (<code>ticker</code>) in the <code>crsp.msenames</code> table. We use the <code>raw_sql()</code> function of the <code>wrds</code> package to execute this query and obtain the resulting dataset. To download the Fama-French factors data, we construct another SQL query that selects the date and the specified factors from the <code>ff.factors_daily</code> table. We then merge the downloaded CRSP and Fama-French datasets on the <code>date</code> column using the <code>pd.merge()</code> function of the pandas package. Finally, we close the connection to WRDS using the <code>close()</code> method of the <code>wrds.Connection</code> object.

As before, we only quickly browse through the Python code generated by ChatGPT and assume the code is executable, which minimizes distractions in our thought process. According to Table 7, it seems that the ChatGPT-generated Python code is as expected. No further modifications are necessary.

However, a quick glance through the ChatGPT-generated Python code, comments, and notes reveals a potential error in coding—the code requests to download the “daily return” data from the “`crsp.msfc`” table on CRSP; however, this table on CRSP is interpreted as “Table: Monthly Stock – Securities” on CRSP and WRDS, and the correct table with daily stock returns should be “`crsp_a_stock.dsfc`”. Similarly, the ChatGPT-requested table of `crsp.msenames` on CRSP should be “`crsp_a_stock.dsenames`”. Therefore, we emphasize in class that ChatGPT is a helpful tool in Python

coding, but we should not use the code without scrutinizing the details. Instead, we should check the correctness of ChatGPT-generated codes and use these codes with caution.

Next, the instructor would need to help students review the details of the Fama-French Three-Factor Model. Specifically, the time-series regression equation for estimating factor loadings on the Fama-French Three-Factor Model for stock i can be written as:

$$R_{i,t} - R_{f,t} = \alpha_i + \beta_i(R_{m,t} - R_{f,t}) + s_iSMB_t + h_iHML_t + \varepsilon_{i,t}, \quad (1)$$

where $R_{i,t}$ is the excess return of stock i at time t ; $R_{f,t}$ is the risk-free rate at time t ; $R_{m,t}$ is the excess return of the market portfolio at time t ; SMB_t is the size factor at time t , calculated as the difference between the returns of small and large stocks; HML_t is the value factor at time t , calculated as the difference between the returns of high and low book-to-market stocks; α_i is the intercept of the regression for stock i , representing the excess return not explained by the factors; β_i is the beta coefficient of the market factor for stock i , representing the sensitivity of the stock's excess return to the market return; s_i and h_i are the sensitivity coefficients of SMB and HML factors for stock i , respectively; $\varepsilon_{i,t}$ is the residual term or error term for stock i at time t .

According to Equation (1), and compared to what is already there in our download data from WRDS, for each stock, we have all independent variables ready, but we need to generate the dependent variable, which is the excess return of stock i . For a typical student in our course of financial data science with basic knowledge of Python from the materials at the beginning of the semester, the student should be able to add this line of Python code easily. If not, given the powerful tool of ChatGPT, it is only one more question away from getting this sample code. Such a communication with ChatGPT is presented in Table 8.

Table 8: Dialog with ChatGPT – Create New Variables

Panel A: Input from the user of ChatGPT
In Python, how to define a new variable "ret_rf = ret - rf," assuming the variables "ret" and "rf" are available in the dataset?

Panel B: Output from ChatGPT
To define a new variable <code>ret_rf</code> as the difference between two existing variables <code>ret</code> and <code>rf</code> in Python, you can simply subtract <code>rf</code> from <code>ret</code> using the <code>-</code> operator, and assign the resulting series to a new variable <code>ret_rf</code> . Here's an example code assuming you have a Pandas DataFrame called <code>data</code> that contains the <code>ret</code> and <code>rf</code> columns:
(Note: ChatGPT generates a Python code here in a stand-alone box. Since the Python code generated by ChatGPT is too long to present in the table, we report all such codes in the Appendix.)
In this code, we use the Pandas indexing operator <code>[]</code> to select the <code>ret</code> and <code>rf</code> columns from the <code>data</code> DataFrame, and then subtract <code>rf</code> from <code>ret</code> . The resulting series is assigned to a new variable <code>ret_rf</code> . Note that this assumes that both <code>ret</code> and <code>rf</code> are numeric columns in the DataFrame, and that they have the same length and index as the other columns in the DataFrame.

Next, we emphasize in class that the downloaded dataset is supposed be a panel data (i.e., a cross-section of a list of stocks plus a time-series of variables, such as returns for each stock). However, using regression to estimate factor loadings for the Fama-French Three-Factor Model is supposed to be time-series regression. Therefore, for each individual stock, we shall run a separate time-series regression. We remind students that it is of critical importance to understand this difference in data structure, and we need to design questions for ChatGPT with this data structure in mind.

Therefore, this time, we need to provide much longer instructions to ChatGPT to generate the Python code that would meet our requirement. We shall break down those instructions into different sentences; Table 9 presents our communication with ChatGPT for this purpose. We remind students again that we should help ChatGPT understand our data structure, and thus the key words such as “time-series regression,” “panel data,” and “time-series variables” are very important.

Table 9: Dialog with ChatGPT – Fama-French Three-Factor Regressions

Panel A: Input from the user of ChatGPT

Write a Python code for a sequence of time-series regressions. The data given is a panel data with a list of stocks and a time series of variables for each stock. For each regression, the dependent variable is "ret_rf", and the three independent variables are "mktrf", "smb", and "hml". Report the regression results (with t-statistics and p-value) for each stock in a table.

Panel B: Output from ChatGPT

Here's an example Python code for running a sequence of time-series regressions with plots for each regression line. This assumes you have a Pandas DataFrame called `panel_data` that contains the panel data for each stock, with columns for "ret_rf", "mktrf", "smb", and "hml". The code performs a separate regression for each stock in the dataset, and generates a table of regression results and a plot of the regression line for each stock.

(Note: ChatGPT generates a Python code here in a stand-alone box. Since the Python code generated by ChatGPT is too long to present in the table, we report all such codes in the Appendix.)

In this code, we first define the list of independent variables as `['mktrf', 'smb', 'hml']`. We then group the panel data by stock ticker using the `groupby()` function of the Pandas DataFrame object.

We initialize an empty DataFrame called `regression_results` to store the regression coefficients, t-statistics, and p-values for each independent variable for each stock. We then loop over each group of data for each stock ticker, and run a separate linear regression using the `OLS()` function from the `statsmodels` package.

We extract the regression coefficients, t-statistics, and p-values from the fitted model and store them in a new row of the `regression_results` DataFrame for each stock.

Finally, we print the `regression_results` DataFrame, which contains the regression coefficients, t-statistics, and p-values for each independent variable for each stock.

Finally, we note that the Python codes generated by ChatGPT only print the final results from regressions on screen. Given the potentially large size of the table, we would rather request Python to output the table to an Excel file. We note to class that whenever we have such a task in mind but are unsure how to write it in Python, ChatGPT can help us. For example, Table 10 represents our conversation with ChatGPT on additional Python code to output a dataframe in Python into an Excel file.

Table 10: Dialog with ChatGPT – Output an Excel File

Panel A: Input from the user of ChatGPT
How to output a python dataframe to a separate Excel file?

Panel B: Output from ChatGPT
To output a Python DataFrame to a separate Excel file, you can use the <code>to_excel()</code> method of the Pandas DataFrame object. Here's an example Python code that demonstrates how to write a DataFrame called <code>my_data</code> to an Excel file called <code>my_file.xlsx</code> :
(Note: ChatGPT generates a Python code here in a stand-alone box. Since the Python code generated by ChatGPT is too long to present in the table, we report all such codes in the Appendix.)
In this code, the <code>to_excel()</code> method writes the contents of the <code>my_data</code> DataFrame to an Excel file called <code>my_file.xlsx</code> . The <code>index=False</code> argument tells Pandas not to include the DataFrame index in the Excel file.
You can also specify additional arguments to the <code>to_excel()</code> method to control the formatting of the Excel file, such as the worksheet name, cell formatting, and whether to include column headers. See the Pandas documentation for more information on these arguments.

We now have all fragments of Python codes generated by ChatGPT for this project. Next, we inspect carefully all these Python codes, merge them into one single code, and get it ready to be executed. Once again, we note in class that certain parts of the code require modifications and corrections, such as the previously mentioned file names in CRSP to access “daily returns” data.

As an example, we assume the list of stocks under consideration include Microsoft Corp. (MSFT), Apple Inc. (AAPL), and Amazon.com Inc. (AMZN), for the sample period from January 1, 2020 to March 31, 2020. Using the merged (and modified) Python codes that are generated by ChatGPT, the calculated factor loadings, t -statistics, and p -values are presented in Table 11.

Table 11: Regression Results Based on ChatGPT Generated Python Code

		Microsoft	Apple	Amazon
ID (permno)		10107	14593	84788
MKTRF	coefficient	1.288	1.155	0.763
	<i>t</i> -stat	23.916	22.157	9.913
	<i>p</i> -value	0.000	0.000	0.000
SMB	coefficient	-0.181	-0.553	0.032
	<i>t</i> -stat	-1.306	-4.123	0.164
	<i>p</i> -value	0.197	0.000	0.871
HML	coefficient	-0.660	-0.380	-0.765
	<i>t</i> -stat	-4.323	-2.575	-3.505
	<i>p</i> -value	0.000	0.013	0.001

3.3 Other Potential Projects for Financial Data Science with ChatGPT

As we can see, the instructional and learning experiences have changed substantially since ChatGPT has become available. We have adapted two typical projects in a course of financial data science with the help of this newly available tool. Since one of the main purposes of this paper is to inspire educators in finance (especially the instructors of financial data science courses) to incorporate ChatGPT into their courses, in this subsection, we suggest other potential projects that are suitable for learning with ChatGPT. We have tested all these projects, and they are all doable. These projects can either be in-class, as previously indicated, or assigned to students as take-home projects. We also note that this is not an exhaustive list of suitable projects.

(a) Direct access of financial databases from various data resources.

We have shown in previous example projects that ChatGPT can help us write Python codes to access popular financial data on the Internet directly (e.g., Yahoo Finance and WRDS). Since it is one of the important learning objectives for a course in financial data science, students would also

need to be familiar with many more available financial databases. The important such data resources for finance can include:

- FRED (Federal Reserve Economic Data), which is a database maintained by the Federal Reserve Bank of St. Louis that provides access to a wide range of economic and financial data series, including data on inflation, interest rates, gross domestic product (GDP), employment, and more.
- EDGAR (Electronic Data Gathering, Analysis, and Retrieval), which is a database maintained by the U.S. Securities and Exchange Commission (SEC) that contains filings made by publicly traded companies and other organizations. These filings include annual reports, quarterly reports, and other regulatory filings that provide information on the financial and operational activities of these organizations.
- Nasdaq Data Link, which is a data service provided by Nasdaq that offers historical and real-time data on a variety of financial instruments including stocks, options, futures, and indices. It provides data on global equities, derivatives, and fixed income markets.
- Quandl, which is a financial data provider that offers a wide range of financial and economic data to individuals, academics, and businesses. It provides data on a variety of topics including stock prices, futures, options, currencies, and economic indicators from over 500 sources. We note that Quandl was acquired by Nasdaq in 2018, but it operates as a distinct entity.

We can easily ask ChatGPT to write Python codes to access those data. An example of such a request can be:

- “write a python code that can directly access WTI prices data from FRED database,”
- “write a python code that can directly access 10-K filing data from SEC Edgar database.”
- “write a python code that can directly access data with nasdaqdatalink and quandl.”

We tested all these prompts in dialog with ChatGPT. Every time, ChatGPT can do a reasonably satisfactory job in generating Python codes that are almost ready to use. With a few potential corrections and debugging, the Python codes can execute smoothly and provide expected results.

(b) Text Analysis and Sentiment Analysis with Natural Language Toolkit (NLTK)

Natural Language Toolkit (NLTK) is a Python library used for working with human language data, such as processing and analyzing text. NLTK provides a set of tools and resources, such as corpora, lexicons, and algorithms for tasks such as tokenization, stemming, tagging, parsing, and sentiment analysis. We can instruct students in the financial data science course to download a 10-K filing for a certain company from SEC EDGAR and run sentiment analysis with NLTK library in Python. We can easily request that ChatGPT run the analyses by stating, “write a python code that can run sentiment analysis on a text document using nltk.” In our experience, the Python code generated by ChatGPT can easily be modified into an applicable program for this purpose.

(c) Data Visualization and Word Clouds

Data visualization and word clouds allow complex financial data and ideas to be conveyed in a clear and concise manner. Presenting data in a visually appealing way can help make data more accessible, understandable, and engaging to a wider audience, which can lead to better decision-making and insights. The visualization plots we can introduce to students include (and certainly are not limited to) boxplot, histogram, pair plot, scatter plot, and one of the most important plots among the list, word clouds. We can easily request ChatGPT to do such coding in Python by stating, “write a python code that can generate a word cloud based on a text document.”

(d) Calculation of Rolling Beta for Stocks

Rolling beta is a measure of a stock's risk or volatility over a certain period, calculated using a rolling window of historical data. In a rolling beta calculation, a regression analysis is performed on a subset of data that “rolls” forward over time, with each new calculation incorporating the most recent data point and dropping the oldest one. This allows for a more dynamic and up-to-date assessment of a stock's risk profile, as compared to a static beta calculation based on a fixed period of historical data. This is a widely used technique for estimating beta, according to the Capital Asset Pricing Model (CAPM). For example, Yahoo Finance reports “Beta (5Y Monthly)” in the homepage of all stock quotes. It is a 5-year rolling beta for the given stock that is calculated based on monthly data. We can easily request ChatGPT to calculate rolling beta by asking it to “write a python code that can calculate the 5-year rolling beta for a stock.” We have successfully tested this prompt and interaction with ChatGPT.

(e) Time Value of Money, Financial Calculator, and Mortgage Payment Schedule

The concept of Time Value of Money is considered “Finance 101,” and ever since the first course in finance, students have been exposed to this important concept. Finance students are often required to learn how to use a financial calculator. One of the direct applications of a financial calculator is calculating mortgage monthly payments of principal and interest (for a typical 30-year fixed-rate mortgage).

With the help of ChatGPT, we now can easily develop a simple “software” of a financial calculator and make it into an executable (.exe) file (using Python libraries such as “pyinstaller”). We can further instruct students to test whether the developed financial calculator is functioning properly by calculating mortgage monthly payments because this information is easily available from real estate websites (e.g., Zillow.com and Trulia.com). We can easily request ChatGPT to do this by instructing, “write a python code that can do all calculations as a financial calculator and make it into

an executable file.” As with the other projects we propose, our test runs demonstrate that the Python code generated by ChatGPT can easily be modified to accomplish our purpose.

4. Conclusion

This paper explores the utility of using ChatGPT in a financial data science course and provides specific suggestions for student projects and other assignments. As an example of artificial intelligence that has the capacity to change the educational landscape of higher education, ChatGPT represents both risks and rewards in the classroom. In our experience, ChatGPT has the potential to replace tedious Python coding with meaningful real-world applications. Therefore, we encourage educators in finance to pay more attention to, take seriously, and embrace newly developed AI technologies, especially ChatGPT.

Despite our optimistic views of ChatGPT, we also recommend proceeding carefully. For instance, we recommend scrutinizing ChatGPT output, particularly Python code, due to AI’s current limitations. Students also need instructor guidance on prompt creation, model checking, understanding the data structure, and other applications. Indeed, ChatGPT is not prepared to replace classroom instruction, but it is a valuable tool that can enable instructors to prioritize financial theory, learning, and applications—the most important foundations for a successful career in finance.

References

- Alshater, M. (2022). *Exploring the Role of Artificial Intelligence in Enhancing Academic Performance: A Case Study of ChatGPT* (SSRN Scholarly Paper No. 4312358). <https://doi.org/10.2139/ssrn.4312358>
- Alt, R., Beck, R., & Smits, M. T. (2018). FinTech and the transformation of the financial industry. *Electronic Markets*, 28(3), 235–243. <https://doi.org/10.1007/s12525-018-0310-9>
- Arner, D. W., Barberis, J., & Buckley, R. P. (2015). The Evolution of FinTech: A New Post-Crisis Paradigm. *Georgetown Journal of International Law*, 47(4), 1271–1320.
- Aydın, Ö., & Karaarslan, E. (2023). *Is ChatGPT Leading Generative AI? What is Beyond Expectations?* (SSRN Scholarly Paper No. 4341500). <https://doi.org/10.2139/ssrn.4341500>
- Birjali, M., Kasri, M., & Beni-Hssane, A. (2021). A comprehensive survey on sentiment analysis: Approaches, challenges and trends. *Knowledge-Based Systems*, 226, 107134. <https://doi.org/10.1016/j.knosys.2021.107134>
- Bussmann, N., Giudici, P., Marinelli, D., & Papenbrock, J. (2020). Explainable AI in fintech risk management. *Frontiers in Artificial Intelligence*, 3, 26.
- Davenport, T. H., & Ronanki, R. (2018). Artificial intelligence for the real world. *Harvard Business Review*, 96(1), 108–116.
- Demajo, L. M., Vella, V., & Dingli, A. (2020). Explainable AI for Interpretable Credit Scoring. *Computer Science & Information Technology (CS & IT)*, 185–203. <https://doi.org/10.5121/csit.2020.101516>
- Dowling, M., & Lucey, B. (2023). ChatGPT for (Finance) research: The Bananarama Conjecture. *Finance Research Letters*, 103662. <https://doi.org/10.1016/j.frl.2023.103662>
- Fama, E. F., & French, K. R. (1992). The Cross-Section of Expected Stock Returns. *The Journal of Finance*, 47(2), 427–465. <https://doi.org/10.1111/j.1540-6261.1992.tb04398.x>
- Fisher, I. E., Garnsey, M. R., & Hughes, M. E. (2016). Natural Language Processing in Accounting, Auditing and Finance: A Synthesis of the Literature with a Roadmap for Future Research. *Intelligent Systems in Accounting, Finance and Management*, 23(3), 157–214. <https://doi.org/10.1002/isaf.1386>
- Huang, K. (2023). Alarmed by AI chatbots, universities start revamping how they teach. *New York Times*.
- Kang, Y., Cai, Z., Tan, C.-W., Huang, Q., & Liu, H. (2020). Natural language processing (NLP) in management research: A literature review. *Journal of Management Analytics*, 7(2), 139–172. <https://doi.org/10.1080/23270012.2020.1756939>

- Lu, Y. (2019). Artificial intelligence: A survey on evolution, models, applications and future trends. *Journal of Management Analytics*, 6(1), 1–29.
<https://doi.org/10.1080/23270012.2019.1570365>
- Lund, B. D., & Wang, T. (2023). Chatting about ChatGPT: how may AI and GPT impact academia and libraries? *Library Hi Tech News*, ahead-of-print(ahead-of-print).
<https://doi.org/10.1108/LHTN-01-2023-0009>
- Olan, F., Liu, S., Suklan, J., Jayawickrama, U., & Arakpogun, E. O. (2022). The role of Artificial Intelligence networks in sustainable supply chain finance for food and drink industry. *International Journal of Production Research*, 60(14), 4418–4433.
- OpenAI. (2023a). *Introducing ChatGPT*. Introducing ChatGPT. <https://openai.com/blog/chatgpt>
- OpenAI. (2023b). *GPT-4 Technical Report* (arXiv:2303.08774). arXiv.
<https://doi.org/10.48550/arXiv.2303.08774>
- Rudin, C., & Radin, J. (2019). Why are we using black box models in AI when we don't need to? A lesson from an explainable AI competition. *Harvard Data Science Review*, 1(2), 10–1162.
- Schueffel, P. (2016). Taming the Beast: A Scientific Definition of Fintech. *Journal of Innovation Management*, 4(4), Article 4. https://doi.org/10.24840/2183-0606_004.004_0004
- Sciences Po. (2023). Sciences Po bans the use of ChatGPT without transparent referencing. *Espace Presse Sciences Po*. <https://newsroom.sciencespo.fr/sciences-po-bans-the-use-of-chatgpt/>
- Suryono, R. R., Budi, I., & Purwandari, B. (2020). Challenges and trends of financial technology (Fintech): A systematic literature review. *Information*, 11(12), 590.
- Tamkin, A., Brundage, M., Clark, J., & Ganguli, D. (2021). *Understanding the Capabilities, Limitations, and Societal Impact of Large Language Models* (arXiv:2102.02503). arXiv.
<https://doi.org/10.48550/arXiv.2102.02503>
- Xing, F. Z., Cambria, E., & Welsch, R. E. (2018). Natural language based financial forecasting: A survey. *Artificial Intelligence Review*, 50(1), 49–73. <https://doi.org/10.1007/s10462-017-9588-9>
- Yan, Y. (2018). A Trend in Business Education. *International Journal of Education and Social Science*, 5(7), 15–20.

Appendix A. Python Codes Generated by ChatGPT

(A-1) Python Codes Generated by ChatGPT – Download Data from Yahoo Finance – Table 1

Python	Copy code
<pre>import pandas_datareader as pdr # Define the list of stocks stocks_list = ['AAPL', 'GOOGL', 'AMZN'] # Define the start and end dates for the data start_date = '2016-01-01' end_date = '2021-12-31' # Download the data using Yahoo Finance as the source data = pdr.get_data_yahoo(stocks_list, start=start_date, end=end_date, interval='m') # Extract the Adj Close prices for each stock prices = data['Adj Close'] # Calculate the monthly returns for each stock returns = prices.pct_change() # Print the first few rows of the returns dataframe print(returns.head())</pre>	

(A-2) Python Codes Generated by ChatGPT – Download Data (Modified) – Table 2

Python	Copy code
<pre>import yfinance as yf # Define the list of stocks stocks_list = ['AAPL', 'GOOGL', 'AMZN'] # Define the start and end dates for the data start_date = '2016-01-01' end_date = '2021-12-31' # Download the data using Yahoo Finance as the source data = yf.download(stocks_list, start=start_date, end=end_date, interval='lmo') # Extract the Adj Close prices for each stock prices = data['Adj Close'] # Calculate the monthly returns for each stock returns = prices.pct_change() # Print the first few rows of the returns dataframe print(returns.head())</pre>	

(A-3) Python Codes Generated by ChatGPT – Optimal Portfolio – Table 3

Python	Copy code
<pre>import pandas as pd import numpy as np from scipy.optimize import minimize # Define the list of stocks stocks_list = ['AAPL', 'GOOGL', 'AMZN'] # Define the start and end dates for the data start_date = '2016-01-01' end_date = '2021-12-31' # Download the data using yfinance as the source data = pd.DataFrame() for stock in stocks_list: stock_data = pd.read_csv(f'https://query1.finance.yahoo.com/v7/finance/download/{stock}?per iod1=1451606400&period2=1640908800&interval=1mo&events=history&includeAdjusted Close=true', index_col='Date', parse_dates=True) stock_data = stock_data[['Adj Close']] stock_data = stock_data.rename(columns={'Adj Close': stock}) data = pd.concat([data, stock_data], axis=1) # Calculate monthly returns for each stock returns = data.pct_change().dropna() # Define the objective function for the mean-variance optimization def portfolio_variance(weights, returns): cov_matrix = np.cov(returns.T) portfolio_variance = np.dot(weights.T, np.dot(cov_matrix, weights)) return portfolio_variance # Define the constraint for the sum of the weights to equal 1 def weight_constraint(weights): return np.sum(weights) - 1.0 # Define initial weights for the optimization n_assets = len(stocks_list) weights_0 = np.ones(n_assets)/n_assets # Define bounds for the optimization (weights must be between 0 and 1) bounds = tuple((0, 1) for i in range(n_assets)) # Perform the mean-variance optimization optimal_weights = minimize(portfolio_variance, weights_0, args=returns, method='SLSQP', bounds=bounds, constraints={'type': 'eq', 'fun': weight_constraint}).x # Print the optimal weights for each stock for i, stock in enumerate(stocks_list): print(f"{stock}: {optimal_weights[i]*100:.2f}%") # Calculate the expected return and volatility of the optimal portfolio portfolio_return = np.sum(returns.mean()*optimal_weights)*12</pre>	

```

portfolio_volatility = np.sqrt(np.dot(optimal_weights.T,
np.dot(np.cov(returns.T), optimal_weights)))*np.sqrt(12)

# Print the expected return and volatility of the optimal portfolio
print(f"Expected return: {portfolio_return*100:.2f}%")
print(f"Volatility: {portfolio_volatility*100:.2f}%")

```

(A-4) Python Codes Generated by ChatGPT – Optimal Portfolio (Simulation) – Table 4

Python	Copy code
<pre> import pandas as pd import numpy as np import random import yfinance as yf # Define the list of stocks stocks_list = ['AAPL', 'GOOGL', 'AMZN'] # Define the start and end dates for the data start_date = '2016-01-01' end_date = '2021-12-31' # Download the data using Yahoo Finance as the source data = yf.download(stocks_list, start=start_date, end=end_date, interval='lmo') prices = data['Adj Close'] returns = prices.pct_change().dropna() # Define the number of simulations and the number of portfolios to generate for each simulation n_simulations = 10000 n_portfolios = 1000 # Define a function to generate a random portfolio of weights def generate_portfolio(n_assets): weights = np.random.random(n_assets) weights /= np.sum(weights) return weights # Define a function to simulate the returns of a portfolio given a set of weights def simulate_portfolio(weights, returns): portfolio_return = np.sum(returns.mean() * weights) * 12 portfolio_volatility = np.sqrt(np.dot(weights.T, np.dot(np.cov(returns.T), weights))) * np.sqrt(12) sharpe_ratio = portfolio_return / portfolio_volatility return sharpe_ratio # Generate multiple sets of random portfolios and simulate their returns optimal_weights = None max_sharpe_ratio = -np.inf for i in range(n_simulations): # Generate a set of random portfolios portfolios = [generate_portfolio(len(stocks_list)) for j in range(n_portfolios)] </pre>	


```

# Simulate the returns of each portfolio and find the one with the
highest Sharpe ratio
sharpe_ratios = [simulate_portfolio(weights, returns) for weights in
portfolios]
max_sharpe_ratio_i = np.max(sharpe_ratios)
if max_sharpe_ratio_i > max_sharpe_ratio:
    optimal_weights = portfolios[np.argmax(sharpe_ratios)]
    max_sharpe_ratio = max_sharpe_ratio_i

# Print the optimal weights for each stock
for i, stock in enumerate(stocks_list):
    print(f"{stock}: {optimal_weights[i]*100:.2f}%")

# Calculate the expected return and volatility of the optimal portfolio
portfolio_return = np.sum(returns.mean()*optimal_weights)*12
portfolio_volatility = np.sqrt(np.dot(optimal_weights.T,
np.dot(np.cov(returns.T), optimal_weights)))*np.sqrt(12)

# Print the expected return and volatility of the optimal portfolio
print(f"Expected return: {portfolio_return*100:.2f}%")
print(f"Volatility: {portfolio_volatility*100:.2f}%")

```

(A-5) Python Codes Generated by ChatGPT – Download Data from WRDS – Table 7

Python	Copy code
<pre> import wrds import pandas as pd # Establish a connection to WRDS db = wrds.Connection() # Specify the list of stocks and Fama-French factors stocks = ['AAPL', 'MSFT', 'GOOG'] factors = ['MKT', 'SMB', 'HML', 'RF'] # Download daily returns data for the stocks using CRSP crsp_query = """ select date, permno, ret from crsp.msf where permno in (select permno from crsp.msenames where ticker in ({0})) and date >= '2000-01-01' """.format("'" + "','".join(stocks) + "'") crsp_data = db.raw_sql(crsp_query) # Download Fama-French factors data ff_query = """ select date, {0} from ff.factors_daily where date >= '2000-01-01' </pre>	

```

"".format(', '.join(factors))

ff_data = db.raw_sql(ff_query)

# Merge the downloaded datasets
merged_data = pd.merge(crsp_data, ff_data, on='date')

# Close the connection to WRDS
db.close()

```

(A-6) Python Codes Generated by ChatGPT – Create New Variables – Table 8

Python	Copy code
<pre> import pandas as pd # Assuming you have a Pandas DataFrame called `data` containing the `ret` and `rf` columns ret_rf = data['ret'] - data['rf'] </pre>	

(A-7) Python Codes Generated by ChatGPT – Fama-French Three-Factor Regression – Table 9

Python	Copy code
<pre> import pandas as pd import statsmodels.api as sm # Assuming you have a Pandas DataFrame called `panel_data` with columns for "ret_rf", "mktrf", "smb", and "hml" # Define the list of independent variables independent_vars = ['mktrf', 'smb', 'hml'] # Group the panel data by stock ticker grouped_data = panel_data.groupby('ticker') # Initialize an empty DataFrame to store the regression results regression_results = pd.DataFrame(columns=['ticker', 'coef_mktrf', 'tstat_mktrf', 'pvalue_mktrf', 'coef_smb', 'tstat_smb', 'pvalue_smb', 'coef_hml', 'tstat_hml', 'pvalue_hml']) # Loop over each stock ticker and run a separate regression for ticker, data in grouped_data: # Define the dependent variable and the independent variables y = data['ret_rf'] X = data[independent_vars] # Fit a linear regression model using the OLS function from statsmodels model = sm.OLS(y, sm.add_constant(X)).fit() # Extract the regression coefficients, t-statistics, and p-values coef = model.params </pre>	

```

tstat = model.tvalues
pvalue = model.pvalues

# Store the regression results in a new row of the DataFrame
regression_results = regression_results.append({'ticker': ticker,
        'coef_mktrf': coef['mktrf'], 'tstat_mktrf': tstat['mktrf'],
        'pvalue_mktrf': pvalue['mktrf'], 'coef_smb': coef['smb'],
        'tstat_smb': tstat['smb'], 'pvalue_smb': pvalue['smb'],
        'coef_hml': coef['hml'], 'tstat_hml': tstat['hml'],
        'pvalue_hml': pvalue['hml']},
        ignore_index=True)

# Print the regression results table
print(regression_results)

```

(A-8) Python Codes Generated by ChatGPT – Output an Excel File – Table 10

Python	Copy code
<pre> import pandas as pd # Assuming you have a Pandas DataFrame called `my_data` # Write the DataFrame to an Excel file my_data.to_excel("my_file.xlsx", index=False) </pre>	