FINM 32900: Data Science for Finance

Course Syllabus, by Jeremy Bejarano

1 Course Description

"Data Science for Finance" is a hands-on course centered on key data science tools in quantitative finance. Acknowledging the field's wide scope, the course focuses on a common skill set across various data science subfields. That is, this course examines elements of the analytical pipeline, from data extraction and cleaning to exploratory analysis, visualization, and modeling, and finally, publication and deployment. It does so with the aim of teaching the tools and principles behind creating reproducible and scalable workflows, including build automation, dependency management, unit testing, the command-line environment, shell scripting, Git for version control, and GitHub for team collaboration. These skills are taught through case studies, each of which will additionally give students practical experience with key financial data sets and sources such as CRSP and Compustat for pricing and financials, macroeconomic data from FRED and the BEA, bond transactions from FINRA TRACE, Treasury auction data from TreasuryDirect, textual data from EDGAR, and high-frequency trade and quote data from NYSE. Prior experience at an intermediate level with Python and the PyData stack is assumed.

2 Objective

The objective of this course is twofold:

1. *The "Science" in Data Science.* Teach a core set of software tools and principles behind creating modern data analytic workflows that are reproducible and scalable from end to end. The idea is that this set of computational tools will serve as a technical foundation for any subfield of data science that the student wishes to pursue. This is born out of the idea that reproducibility is fundamental to good science.
2. *The "Data" in Data Science.* Give students hands-on experience with a series of important and commonly used financial data sets and/or sources, including CRSP, Compustat, FRED, BEA data, FINRA TRACE, TreasuryDirect, EDGAR, and NYSE TAQ. Not only does each data set require some degree of domain specific knowledge (e.g., to clean and interpret properly), but each data set also poses different computational challenges to use effectively. For example, using EDGAR means working with large amounts of unstructured text or data from NYSE TAQ involves learning to work with big data (approximately 100 GBs per day). Each case study is designed to give students the basic knowledge and tools to use these data sets effectively.

3 Motivation

3.1 What is data science and what is the purpose of this class?

Given the increasing complexity of the computational sciences, the ability to produce readable, reusable, and reproducible code and analyses is increasingly important in order to produce good science and to add value within an organization. Recognizing that data science is a broad term encompassing many fields, this course introduces a core set of practical tools and techniques common to all subfields of data science as they are used within quantitative finance, with a special emphasis on those necessary to build behind creating reproducible and scalable workflows (e.g., reproducible analytical pipelines).

A common characterization of data science describes it as a combination of coding/computer skills, mathematics and statistics, and domain specific knowledge (see **Figure 1**). These computer skills, colloquially referred to as "hacking skills," are called such because they refer to skills that are often not taught in a traditional computer science curriculum. A computer science curriculum might include theoretical courses on algorithms, operating systems, or programming languages, whereas the practical skills related to the computing ecosystem---the command-line environment, shell scripting, data wrangling and visualization, or version control---are often left to the individual to learn on their own. (This was even the motivation for a supplementary CS course at MIT, "The Missing Semester of Your CS Education.") In this course, I provide a structured overview of these important tools and demonstrate, by example, how they can be used to (1) collect, clean, and analyze economic and financial data; (2) facilitate collaboration among groups of researchers; and (3) to create analyses that are easy for outside users to reproduce. All of the examples and/or case studies used to illustrate these concepts and to teach these tools come from common tasks or applications within quantitative finance, including data wrangling, feature engineering, time series forecasting, performance evaluation, portfolio optimization, back testing, and reporting results.

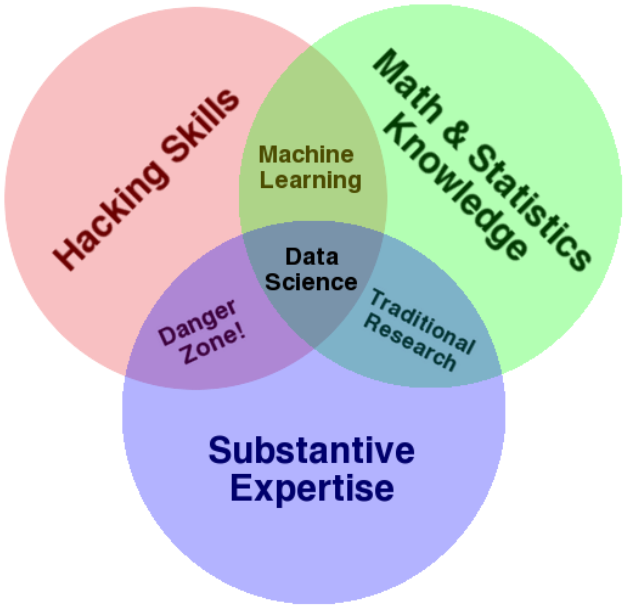

Figure 1: "The Data Science Venn Diagram", by Drew Conway

3.2 How is the course structured?

Furthermore, the series of case studies used in this course is designed to introduce students to a wide range of important financial data sets. Each case study will teach the domain-specific knowledge necessary to properly use and interpret each data set. For example, these case studies may include the following data sources: CRSP and Compustat for pricing and financials, macroeconomic data from FRED and the BEA, bond transactions from FINRA TRACE, Treasury auction data from TreasuryDirect, textual data from EDGAR, and high-frequency trade and quote data from NYSE. Furthermore, each data set comes with its own unique computational challenges to use properly. For example, using EDGAR means working with large amounts of unstructured text or data from NYSE TAQ involves learning to work with big data (approximately 100 GBs per day). Thus, a series of case studies involving each of these data sets will serve as a useful framework around which to teach important data science tools and skills as well as a good way to insert meaningful hands-on experiences into the coursework. Each of these applications will be self-contained, in that the basic statistical or financial theory needed to understand these tasks

will accompany their discussion so long as the students enter the course with prior experience with Python and the PyData stack at an intermediate level.

3.3 Why focus on this specific set of computational tools?

To emphasize the importance of these tools and the principles behind them, I now consider comments made by groups and individuals within various contexts. Within the context of the academic economics research, I refer to the following motivation by economists Matthew Gentzkow and Jesse M. Shapiro in Code and Data for the Social Sciences: A Practitioner's Guide[1]:

> "Though we all write code for a living, few of the economists, political scientists, psychologists, sociologists, or other empirical researchers we know have any formal training in computer science. Most of them picked up the basics of programming without much effort, and have never given it much thought since. Saying they should spend more time thinking about the way they write code would be like telling a novelist that she should spend more time thinking about how best to use Microsoft Word. Sure, there are people who take whole courses in how to change fonts or do mail merge, but anyone moderately clever just opens the thing up and figures out how it works along the way. This manual began with a growing sense that our own version of this self-taught seat-of-the pants approach to computing was hitting its limits…
>
> "Here is a good rule of thumb: If you are trying to solve a problem, and there are multi-billion dollar firms whose entire business model depends on solving the same problem, and there are whole courses at your university devoted to how to solve that problem, you might want to figure out what the experts do and see if you can't learn something from it. This [course] is about translating insights from experts in code and data into practical terms for empirical social scientists."

Within the context of economic statistics and financial regulators, consider this statement by the UK's Office for Statistics Regulation[2]:

> "In 2017 we championed the Reproducible Analytical Pipeline (RAP), a new way of producing official statistics… This approach involved using programming languages to automate manual processes, version control software to robustly manage code and code storage platforms to collaborate, facilitate peer review and publish analysis… . We consider that RAP principles support all three pillars of the Code of Practice for Statistics: trustworthiness, quality and value. Trustworthiness, by increasing transparency; quality, by reducing the risk of manual errors; and value, by enabling analytical time to be spent adding value for users rather than on menial, repetitive tasks. It is for all of these reasons that we are so passionate about promoting the use of RAP."

And, within the context of the finance industry[3],

---

[1] Code and Data for the Social Sciences: A Practitioner's Guide. Matthew Gentzkow, Jesse M. Shapiro, 2014. http://web.stanford.edu/~gentzkow/research/CodeAndData.pdf

[2] Reproducible Analytical Pipelines: Overcoming barriers to adoption: https://osr.statisticsauthority.gov.uk/wp-content/uploads/2021/03/Reproducible-Analytical-Pipelines-Overcoming-barriers-to-adoption.pdf

[3] See Data science jobs in finance: "super smart" people wasting talent and "The ultimate guide for financial data science"

> "We have been in financial data science long enough, when it was called quantitative finance, to see some clear patterns that we believe are detrimental to the career of a financial data scientist: lack of fundamental core knowledge in computer science, lack of SQL knowledge, … lack of visualization abilities to share business insights… Technical knowledge is, in itself, not what's most important; it's an understanding of technical functions within a production environment that truly prove a candidate's worth… Data scientists can be super smart, but if no lines of their code will go into production, it's a waste of talent."

This course is designed to explore the core tools and techniques of data science, with an emphasis on the modern software tools and techniques that make such messy, large-scale projects not only feasible, but also easy to reproduce and reuse. This includes shell scripting, build automation, dependency management, unit testing, version control with Git, and effective collaboration with GitHub. We will explore these tools from the perspective of a quantitative researcher or data scientist working in finance, drawing on real-world applications from finance and economics, with an emphasis on hands-on work with actual data sets.

4 Notes and Course Reference Material

This course will use notes specific to this course, to be distributed via GitHub. In addition to these lecture notes, I will draw from the following references which you may find useful.

- The Missing Semester of Your CS Education, MIT Lecture Notes
- Code and Data for the Social Sciences: A Practitioner's Guide, by Matthew Gentzkow and Jesse M. Shapiro
- QuantEcon Data Science Lectures
- American Economic Association Unofficial Guide on Replication

5 Prerequisites

Students should have at least basic programming ability in Python, as well as basic knowledge of probability, statistics, and econometrics. Lectures will feature live programming exercises in class, so students should have a WiFi-enabled laptop to bring to class.

Software to be used in class. Before the first class, please make sure to install the required software and sign up for the required services.

Students will need to install the following software of their laptop. Each of these pieces of software are free:

- Anaconda distribution of Python (Individual Edition)
- Visual Studio Code (NOT Visual Studio. Visual Studio Code is different from Visual Studio)
- Git
- GitKraken
- TeX Live
- Docker
- MobaXterm (Home Edition)

Students should also sign up for an account with the following websites. We will use free versions of each of these services:

- GitHub
- IPUMS CPS
- Wharton Research Data Services (WRDS) Apply for access through the University of Chicago. Contact FINM office.

6 Assessment

Grades will be based on 5 coding assignments (50%), a midterm (20%), a final group project (25%), and participation (5%). Assignments will be submitted via GitHub classroom.

- Assignments will be submitted individually and will be graded using GitHub's automated testing tools.
- The final project will be completed in groups. Students will choose the project from among a few options provided at the beginning of the quarter. The project will be graded not only on how well it accomplishes the assigned data cleaning and analysis task, but will be primarily graded on whether (1) the steps to reproduce it are fully automated and well documented, (2) the code is written in a clean and reusable fashion, and (3) the results are presented clearly and presented in a way that convinces the reader that the results are correct.
- The participation grade will depend on the positive impacts that a student has on the class. These include participating in in-class discussions and/or answering questions on the class GitHub page (or on Canvas). Students are in no way penalized for giving wrong answers in these in-class discussions nor is there any penalty for asking for help—asking for help is often the best way to learn!

7 Course Outline

The course outline below may change as time allows.

Unit 1: What is Data Science? What are reproducible analytical pipelines and why are they so important?

In this first lecture, we discuss the importance of reproducibility within the field of data science. We'll start with a discussion of various cases in which the tools of data science add value within the business context. We'll then begin discussing the suite of tools that we will use in this class to produce reproducible analytical pipelines. Recall, the end goal is for students to become familiar with the set of computational tools necessary to create their own projects that are reproducible from end to end. I will demo this by showing two such projects.

The tools for this first unit, as an example, we will construct our own fully-automated analysis from a template that I provide. We will learn how to construct all aspects of the pipeline as the course progresses. The focus in this first unit will be on generating automated reports in LaTeX, with data that is pulled from the web, using functions that are covered by unit tests, using code that is tracked by Git.

Summary

- **Software Tools:** Unit testing with PyTest, the basics of Git, and continuous integration via GitHub (used here for automatic grading on GitHub), Simple Shell Scripts (we'll use a proper build system later), LaTeX (from template for now), conda environments
- **Featured Data Source:** Nasdaq Data Link and Ken French Data Library
- **Math/Stats/Finance Tools:** Performance analysis and backtesting trading strategies. Alpha vs Beta
- **Case Study:**
    - Is There a Replication Crisis in Finance?
    - Open Source Asset Pricing (An example of easy reproducibility and transparency—also an example of a public project accepting pull requests on GitHub.)
    - The Limits of p-Hacking: Some Thought Experiments
    - Venn by Two Sigma
- **TA Session:** The shell environment, shell commands, absolute vs relative paths. (Enough to understand running PyTest to test HW)

Unit 2: A Review of Time Series Forecasting, plus a Discussion of Build Systems

Many of the concepts in data science require a firm understanding of the basics of time series forecasting. In this lecture, we'll review a few of the core concepts and discuss some major pitfalls practitioners may encounter (e.g., look-ahead bias, non-stationarity, spurious correlations).

This review will lead us into a discussion about presenting the results of our analysis, generating charts and tables, and creating a system that will automatically assemble the output of our code into a neatly organized report.

Summary

- **Software Tools:** Statsmodels, Markdown within Jupyter Notebooks, Best Practices for Tracking Jupyter Notebooks in Git, JupyText, GNU Make, PyDoit
- **Featured Data Source:** FRED and FRED-MD
- **Math/Stats/Finance Tools:** A few basic time series forecasting concepts—ARMA models, stationarity, impulse response functions, SARIMAX, out of sample testing
- **Case Study:** Monash Time Series Forecasting Repository

Unit 3: Data Query Automation, plus a Discussion of Environment Variables and Secrets

This week will focus on a common first step in the reproducible analytical pipeline: pulling and cleaning data from remote repositories. This often involves using SQL to pull data from relational data bases. Since this automation often involves pulling data from secure data bases, it's important to manage properly manage secrets such as API keys and passwords. We will discuss methods for properly storing these secrets, while maintaining full automation. This will lead us into a broader discussion of environment variables broadly.

As our case study, we'll automate the steps used to pull data from CRSP and Compustat to construct commonly used factors in linear multifactor asset pricing models. We'll discuss how multifactor asset pricing models play a large part of modern asset pricing. This will emphasize the real-world importance of linear multi-factor asset pricing models by discussing the history and investment philosophy from researchers at Dimensional Fund Advisors and AQR.

Discuss the following paper and discuss some of the basic analyses that are used in this framework.

- Cochrane, John H. "Portfolio advice for a multifactor world." Economic Perspectives, 1999, vol. 23, issue Q III, 59-78
- Dimensional Fund Advisors: "Why Active Investing is a Negative Sum Game"
- Value Investing and AQR — "Devil is in the HML Details"

Summary

- **Software Tools:** Environment variables, the command-line environment, protecting API keys and other secrets in shared project, First Steps with SQL, linearmodels
- **Featured Data Source:** CRSP US Stock Databases and Compustat Financials, both accessed via WRDS
- **Math/Stats/Finance Tools:** Linear multi-factor asset pricing models

Unit 4: Creating Reusable Tools and Social Coding with GitHub, plus a Discussion about Tidy Data

This week will focus on creating reusable tools and how to document them. We'll discuss file and folder structure within a project, how to create your own simple Python modules and best practices for doing so. We'll discuss how to automate the generation of documentation.

We'll also discuss advanced version control with Git, pull requests, and how to divide up work on GitHub.

We'll also discuss the philosophy of "tidy data." We'll apply this philosophy to cleaning US Treasury auction data and explore pricing patterns in Treasury securities (e.g., the on-the-run/off-the-run spread).

Summary

- **Software Tools:** Advanced Version Control with Git, Social Coding with Git and GitHub, GitHub Pull Requests, GitHub Issue Tracker, Python Modules, Generating Documentation with Sphinx or Jupyter Book
- **Featured Data Source:** CRSP US Treasury Database and TreasuryDirect (Web APIs Securities)
- **Math/Stats/Finance Tools:** Basic fixed income mathematics
- **Case Study:** Gürkaynak, Refet S., Brian Sack, and Jonathan H. Wright. "The US Treasury yield curve: 1961 to the present." Journal of monetary Economics 54, no. 8 (2007): 2291-2304.

Unit 5: Visualization

In this unit, we'll discuss the art of generating good visualizations. We'll practice generating a variety of charts seen in finance and economics. We'll also include an overview of various Python plotting libraries.

Here, we'll also include a discussion of including the Bloomberg terminal in our workflows, discussing the Bloomberg Python API, as well as some 3rd party wrappers (xbbg or blp for bql queries)

Summary

- **Software Tools:** Matplotlib, Seaborn, Plotly, plotnine (ggplot2), Vega-Altaire, xbbg
- **Featured Data Source:** Bloomberg Terminal via Bloomberg Python API

Unit 6: Working with Remote Machines, plus a discussion of Text Mining

In this unit, we'll discuss tools for working with remote machines. This includes how to offload computation to a remote machine as well as how to pull data from a remote. We'll discuss tools such as a secure shell (SSH), tools for pulling data from the web (wget), and tools for copying files back and forth (scp or rsync). To demonstrate some of these concepts, we'll explore some examples of text mining on SEC filings.

Summary

- **Featured Data Source:** SEC Filings via EDGAR as well as the WRDS SEC Analytics Suite (accessed via the WRDS server)
- **Software Tools:** Working with remote machines (API access vs SSH), Running a Jupyter notebook on a remote machine (Browser, with Python extension of VS Code, but not with SSH extension of VS Code), ssh, wget, rsync, scp, the SLURM scheduler
- **Math/Stats/Finance Tools:** sentiment analysis

Unit 7: ~~Big~~ Medium Data in Python

Here, we'll discuss the challenges associated with data sets that become progressively larger. In particular, we'll discuss the challenges associated with "medium-size data." Our case study will focus on the current limitations of Pandas for such data sets. For comparison, we'll explore alternatives such as Polars and PySpark. In our data demo, we'll explore census data and demonstrate how Pandas can fail on the IPUMS data when the extraction is too large. We'll use this data in a discussion of GDP and Inflation forecasting. We'll also discuss a puzzle associated with wage growth during the Great Recession and the idea of creating a better series with which to forecast business cycles.

Summary

- **Software Tools:** Advanced data wrangling with Pandas, comparison of Pandas with Polars and PySpark
- **Math/Stats/Finance Tools:** Macroeconomic forecasting, Controlling for changes in sampling over time
- **Featured Data Source:** Current Population Survey (CPS) from the US Census Bureau, via IPUMS

- **Case Study:** Two Sigma - pandas at a Crossroads

Week 8: Big Data, featuring NYSE Trade and Quote Data

In this unit, we'll now encounter our first truly large data set. Such data sets require extra creativity when working with them. We will explore a data set of trades and quotes from the NYSE. Given the number of trades and quotes that the NYSE processes daily, this data set grows very quickly. We'll discuss strategies for calculating basic intraday sample statistics as well as more complex statistics, such as calculating National Best Bid and Offer (NBBO) at any given time. We'll access this data via the WRDS cloud.