Introduction to Data Science

STAT 3255/5255 @ UConn, Spring 2025

Jun Yan and Students in Spring 2025

2025-01-27

Table of contents

Preliminaries **1**
- Sources at GitHub . . . 1
- Compiling the Classnotes . . . 2
- Midterm Project . . . 2
- Final Project . . . 3
- Adapting to Rapid Skill Acquisition . . . 3
- Wishlist . . . 4
  - Students in 3255 . . . 4
  - Students in 5255 . . . 5
- Course Logistics . . . 5
  - Presentation Orders . . . 5
  - Presentation Task Board . . . 7
  - Final Project Presentation Schedule . . . 10
  - Contributing to the Class Notes . . . 11
  - Homework Requirements . . . 11
  - Quizzes about Syllabus . . . 12
- Practical Tips . . . 12
  - Data analysis . . . 12
  - Presentation . . . 13
- My Presentation Topic (Template) . . . 13
  - Introduction . . . 13
  - Sub Topic 1 . . . 14
  - Sub Topic 2 . . . 14
  - Sub Topic 3 . . . 14
  - Conclusion . . . 14
  - Further Readings . . . 14

1 Introduction **15**
1.1 What Is Data Science? . . . 15
1.2 Expectations from This Course . . . 16
1.3 Computing Environment . . . 17
1.3.1 Command Line Interface . . . 17
1.3.2 Python . . . 18
1.4 Data Science Ethics . . . 19
1.4.1 Introduction . . . 19
1.4.2 Principles of Ethical Data Science . . . 19
1.4.3 Ensuring Ethics in Practice . . . 22
1.4.4 Conclusion . . . 24

2 Project Management **25**
2.1 Set Up Git/GitHub . . . 25
2.2 Most Frequently Used Git Commands . . . 26
2.3 Tips on using Git: . . . 27
2.4 Pull Request . . . 27

3 Reproducible Data Science **29**
3.1 Introduction to Quarto . . . 29
3.2 Compiling the Classnotes . . . 30
3.2.1 Set up your Python Virtual Environment . . . 30
3.2.2 Clone the Repository . . . 31
3.2.3 Render the Classnotes . . . 32

4 Python Refreshment **33**
4.1 Know Your Computer . . . 33
4.1.1 Operating System . . . 33
4.1.2 File System . . . 34
4.2 The Python World . . . 35
4.3 Standard Library . . . 35
4.4 Important Libraries . . . 37
4.5 Writing a Function . . . 38
4.5.1 Monty Hall . . . 39

4.6 Variables versus Objects 40
4.7 Number Representation 43
4.7.1 Integers 43
4.7.2 Floating Number 44
4.8 Virtual Environment 46

5 Exercises **49**

References **57**

Preliminaries

The notes were developed with Quarto; for details about Quarto, visit https://quarto.org/docs/books.

Sources at GitHub

These lecture notes for STAT 3255/5255 in Spring 2025 represent a collaborative effort between Professor Jun Yan and the students enrolled in the course. This cooperative approach to education was facilitated through the use of GitHub, a platform that encourages collaborative coding and content development. To view these contributions and the lecture notes in their entirety, please visit our GitHub repository at https://github.com/statds/ids-s25.

Students contributed to the lecture notes by submitting pull requests to our GitHub repository. This method not only enriched the course material but also provided students with practical experience in collaborative software development and version control.

For those interested, class notes from Fall 2024, Spring 2024, Spring 2023, and Spring 2022 are also publicly accessible. These archives offer insights into the evolution of the course content and the different perspectives brought by successive student cohorts.

Compiling the Classnotes

To reproduce the classnotes output on your own computer, here are the necessary steps:

- Clone the classnotes repository to an appropriate location on your computer; see Chapter Chapter 2 for using Git.
- Set up a Python virtual environment in the root folder of the source; see Section Section 4.8.
- Activate your virtual environment.
- Install all the packages specified in `requirements.txt` in your virtual environment:

```
pip install -r requirements.txt
```

- For some chapters that need to interact with Google map services, you need to save your API key in a file named `api_key.txt` in the root folder of the source.
- Render the book with `quarto render` from the root folder on a terminal; the rendered book will be stored under `_book`.

Midterm Project

Reproduce NYC street flood research (Agonafir, Lakhankar, et al., 2022; Agonafir, Pabon, et al., 2022).

Four students will be selected to present their work in a workshop at the 2025 NYC Open Data Week. You are welcome to invite your family and friends to join the the workshop.

Final Project

Students are encouraged to start designing their final projects from the beginning of the semester. There are many open data that can be used. Here is a list of data challenges that you may find useful:

- ASA Data Challenge Expo: big data in 2025
- Kaggle
- DrivenData
- [15 Data Science Hackathons to Test Your Skills in 2025](https://www.fynd.academy/blog/data-science-hackathons}

If you work on sports analytics, you are welcome to submit a poster to Connecticut Sports Analytics Symposium (CSAS) 2025.

Adapting to Rapid Skill Acquisition

In this course, students are expected to rapidly acquire new skills, a critical aspect of data science. To emphasize this, consider this insightful quote from VanderPlas (2016):

> When a technologically-minded person is asked to help a friend, family member, or colleague with a computer problem, most of the time it's less a matter of knowing the answer as much as knowing how to quickly find an unknown answer. In data science it's the same: searchable web resources such as online documentation, mailing-list threads, and StackOverflow answers contain a wealth of information, even (especially?) if it is a topic you've found yourself searching before. Being an effective practitioner of data science is less about memorizing the tool or command you should use for every possible situation, and more about learning to effectively find the information you

> don't know, whether through a web search engine or another means.

This quote captures the essence of what we aim to develop in our students: the ability to swiftly navigate and utilize the vast resources available to solve complex problems in data science. Examples tasks are: install needed software (or even hardware); search and find solutions to encountered problems.

Wishlist

This is a wish list from all members of the class (alphabetical order, last name first, comma, then first name). Here is an example.

- Yan, Jun
  - Make practical data science tools accessible to undergraduates.
  - Pass real-world data science project experience to students.
  - Co-develop a Quarto book in collaboration with the students.
  - Train students to participate real data science competitions.

Add yours through a pull request; note the syntax of nested list in Markdown.

Students in 3255

- Ackerman, John
- Alsadadi, Ammar Shaker
- Chen, Yifei
- El Zein, Amer Hani
- Febles, Xavier Milan
- Horn, Alyssa Noelle
- Hutchins, Isabella Grace

- Jun, Joann
- Kline, Daniel Esteban
- Lagutin, Vladislav
- Lang, Lang
- Li, Shiyi
- Lin, Selena
- Long, Ethan Kenneth
- Nasejje, Ruth Nicole
- Pfeifer, Nicholas Theodore
- Reed, Kyle Daniel
- Roy, Luke William
- Schittina, Thomas
- Schlessel, Jacob E
- Symula, Sebastian
- Tamhane, Shubhan
- Tomaino, Mario Anthony

Students in 5255

- Edo, Mezmur Wossenu
- Mundiwala, Mohammad Moiz
- Vellore, Ajeeth Krishna
- Zhang, Gaofei

Course Logistics

Presentation Orders

The topic presentation order is set up in class.

```
with open('rosters/3255.txt', 'r') as file:
    ug = [line.strip() for line in file]
with open('rosters/5255.txt', 'r') as file:
    gr = [line.strip() for line in file]
presenters = ug + gr
target = "Blanchard"  # pre-arranged 1st presenter
presenters = [name for name in presenters if target not in name]

import random
## seed jointly set by the class
random.seed(5347 + 2896 + 9050 + 1687 + 63)
random.sample(presenters, len(presenters))
## random.shuffle(presenters) # This would shuffle the list in place
```

```
['Symula,Sebastian',
 'Kline,Daniel Esteban',
 'Zhang,Gaofei',
 'Lin,Selena',
 'Lang,Lang',
 'Long,Ethan Kenneth',
 'Edo,Mezmur Wossenu',
 'Nasejje,Ruth Nicole',
 'Alsadadi,Ammar Shaker',
 'Tamhane,Shubhan',
 'Chen,Yifei',
 'Jun,Joann',
 'Horn,Alyssa Noelle',
 'Ackerman,John',
 'Schittina,Thomas',
 'Reed,Kyle Daniel',
 'El Zein,Amer Hani',
 'Pfeifer,Nicholas Theodore',
 'Li,Shiyi',
```

```
'Mundiwala,Mohammad Moiz',
'Vellore,Ajeeth Krishna',
'Febles,Xavier Milan',
'Tomaino,Mario Anthony',
'Lagutin,Vladislav',
'Roy,Luke William',
'Kravette,Noah']
```

Switching slots is allowed as long as you find someone who is willing to switch with you. In this case, make a pull request to switch the order and let me know.

You are welcome to choose a topic that you are interested the most, subject to some order restrictions. For example, decision tree should be presented before random forest or extreme gradient boosting. This justifies certain requests for switching slots.

Presentation Task Board

Here are some example tasks:

- Making presentations with Quarto
- Markdown jumpstart
- Data science communication skills
- Import/Export data
- Arrow as a cross-platform data format
- Database operation with Structured query language (SQL)
- Grammer of graphics
- Handling spatial data
- Visualize spatial data in a Google map
- Animation
- Classification and regression trees
- Support vector machine

- Random forest
- Naive Bayes
- Bagging vs boosting
- Neural networks
- Deep learning
- TensorFlow
- Autoencoders
- Reinforcement learning
- Developing a Python package
- Web scraping

Please use the following table to sign up.

Date	Presenter	Topic
09/11	Zachary Blanchard	Presentation with Quarto
09/16	Deyu Xu	Import/Export data
09/18	Sara Clokey	Communications in Data Science
09/23	Dorathea Johnson	Database with SQL
09/25	Xavier Febles	Statistical tests
09/30	Jack Bienvenue	Visualizing Spatial Data in a Google Map

Date	Presenter	Topic
10/02	Julia Mazzola	Data Visualization with Plotnine
10/07	Suha Akach	Naive Bayes classifier
10/09	Rahul Manna	Animation
10/23	Jaden Astle	Classification and Regression Trees
10/23	Olivia Kashalapov	Synthetic Minority Oversampling Technique (SMOTE)
10/28	Data science alumni panel	
10/30	Emily Borowski	Random Forest
10/30	Aditya Paricharak	Neural Networks
11/04	Melanie Desroches	Web Scraping
11/06	Qianruo Tan	Reinforcement Learning
11/11	Aansh Jha	K-means Clustering

Date	Presenter	Topic
11/11	Owen Babiec	Calling R from Python and Vice Versa
11/13	Stef Baptista	
11/13	Mohammad Parvez	Extracting and Analyzing Census Data

Final Project Presentation Schedule

We use the same order as the topic presentation for undergraduate final presentation. An introduction on how to use Quarto to prepare presentation slides is availabe under the `templates` directory in the classnotes source tree, thank to Zachary Blanchard, which can be used as a template to start with.

Date	Presenter
11/18	Sara Clokey; Dorothea Johnson; Xavier Febles; Jack Bienvenue
11/20	Julia Mazzola; Suha Akach; Rahul Manna; Jaden Astle
12/02	Olivia Kashalapov; Emily Borowski Qianruo Tan; Melanie Desroches
12/04	Aditya Paricharak; Aansh Jha; Owen Babiec; Stef Baptista

Contributing to the Class Notes

Contribution to the class notes is through a 'pull request'.

- Start a new branch and switch to the new branch.
- On the new branch, add a `qmd` file for your presentation
- If using Python, create and activate a virtual environment with `requirements.txt`
- Edit `_quarto.yml` add a line for your `qmd` file to include it in the notes.
- Work on your qmd file, test with `quarto render`.
- When satisfied, commit and make a pull request with your quarto files and an updated `requirements.txt`.

I have added a template file `mysection.qmd` and a new line to `_quarto.yml` as an example.

For more detailed style guidance, please see my notes on statistical writing.

Plagiarism is to be prevented. Remember that these class notes are publicly available online with your names attached. Here are some resources on how to avoid plagiarism. In particular, in our course, one convenient way to avoid plagiarism is to use our own data (e.g., NYC Open Data). Combined with your own explanation of the code chunks, it would be hard to plagiarize.

Homework Requirements

- Use the repo from Git Classroom to submit your work. See Chapter Chapter 2.
  - Keep the repo clean (no tracking generated files).
    - Never "Upload" your files; use the git command lines.

        * Make commit message informative (think about the readers).
    - Make at least 10 commits and form a style of frequent small commits.

- Track `quarto` sources only in your repo. See Chapter Chapter 3.
- For the convenience of grading, add your standalone html or pdf output to a release in your repo.
- For standalone pdf output, you will need to have LaTeX installed.

Quizzes about Syllabus

- Do I accept late homework?
- Could you list a few examples of email etiquette?
- How would you lose style points?
- Would you use CLI and GUI?
- How many students will present at 2025 NYC ODW and when will the presentations be?
- What's the first date on which you have to complete something about your final project?
- Can you use AI for any task in this course?
- Anybody needs a reference letter? How could you help me to help you?

Practical Tips

Data analysis

- Use an IDE so you can play with the data interactively
- Collect codes that have tested out into a script for batch processing
- During data cleaning, keep in mind how each variable will be used later

- No keeping large data files in a repo; assume a reasonable location with your collaborators

Presentation

- Don't forget to introduce yourself if there is no moderator.
- Highlight your research questions and results, not code.
- Give an outline, carry it out, and summarize.
- Use your own examples to reduce the risk of plagiarism.

My Presentation Topic (Template)

This section was prepared by John Smith.

Use Markdown syntax. If not clear on what to do, learn from the class notes sources.

- Pay attention to the sectioning levels.
- Cite references with their bib key.
- In examples, maximize usage of data set that the class is familiar with.
- Could use datasets in Python packages or downloadable on the fly.
- Test your section by `quarto render <filename.qmd>`.

Introduction

Here is an overview.

Sub Topic 1

Put materials on topic 1 here

Python examples can be put into `python` code chunks:

```
# import pandas as pd

# do something
```

Sub Topic 2

Put materials on topic 2 here.

Sub Topic 3

Put matreials on topic 3 here.

Conclusion

Put sumaries here.

Further Readings

Put links to further materials.

1 Introduction

1.1 What Is Data Science?

Data science is a multifaceted field, often conceptualized as resting on three fundamental pillars: mathematics/statistics, computer science, and domain-specific knowledge. This framework helps to underscore the interdisciplinary nature of data science, where expertise in one area is often complemented by foundational knowledge in the others.

A compelling definition was offered by Prof. Bin Yu in her 2014 Presidential Address to the Institute of Mathematical Statistics. She defines

$$\text{Data Science} = \text{SDC}^3,$$

where

- 'S' represents Statistics, signifying the crucial role of statistical methods in understanding and interpreting data;
- 'D' stands for domain or science knowledge, indicating the importance of specialized expertise in a particular field of study;
- the three 'C's denotes computing, collaboration/teamwork, and communication to outsiders.

Computing underscores the need for proficiency in programming and algorithmic thinking, collaboration/teamwork reflects the inherently collaborative nature of data science projects, often requiring teams with diverse skill sets, and communication to outsiders emphasizes the importance of

translating complex data insights into understandable and actionable information for non-experts.

This definition neatly captures the essence of data science, emphasizing a balance between technical skills, teamwork, and the ability to communicate effectively.

1.2 Expectations from This Course

In this course, students will be expected to achieve the following outcomes:

- **Proficiency in Project Management with Git**: Develop a solid understanding of Git for efficient and effective project management. This involves mastering version control, branching, and collaboration through this powerful tool.
- **Proficiency in Project Reporting with Quarto**: Gain expertise in using Quarto for professional-grade project reporting. This encompasses creating comprehensive and visually appealing reports that effectively communicate your findings.
- **Hands-On Experience with Real-World Data Science Projects**: Engage in practical data science projects that reflect real-world scenarios. This hands-on approach is designed to provide you with direct experience in tackling actual data science challenges.
- **Competency in Using Python and Its Extensions for Data Science**: Build strong skills in Python, focusing on its extensions relevant to data science. This includes libraries like Pandas, NumPy, and Matplotlib, among others, which are critical for data analysis and visualization.

- **Full Grasp of the Meaning of Results from Data Science Algorithms**: Learn to not only apply data science algorithms but also to deeply understand the implications and meanings of their results. This is crucial for making informed decisions based on these outcomes.
- **Basic Understanding of the Principles of Data Science Methods**: Acquire a foundational knowledge of the underlying principles of various data science methods. This understanding is key to effectively applying these methods in practice.
- **Commitment to the Ethics of Data Science**: Emphasize the importance of ethical considerations in data science. This includes understanding data privacy, bias in data and algorithms, and the broader social implications of data science work.

1.3 Computing Environment

All setups are operating system dependent. As soon as possible, stay away from Windows. Otherwise, good luck (you will need it).

1.3.1 Command Line Interface

On Linux or MacOS, simply open a terminal.

On Windows, several options can be considered.

- Windows Subsystem Linux (WSL): https://learn.microsoft.com/en-us/windows/wsl/
- Cygwin (with X): https://x.cygwin.com
- Git Bash: https://www.gitkraken.com/blog/what-is-git-bash

To jump start, here is a tutorial: Ubunto Linux for beginners.

At least, you need to know how to handle files and traverse across directories. The tab completion and introspection supports are very useful.

Here are several commonly used shell commands:

- `cd`: change directory; `..` means parent directory.
- `pwd`: present working directory.
- `ls`: list the content of a folder; `-l` long version; `-a` show hidden files; `-t` ordered by modification time.
- `mkdir`: create a new directory.
- `cp`: copy file/folder from a source to a target.
- `mv`: move file/folder from a source to a target.
- `rm`: remove a file a folder.

1.3.2 Python

Set up Python on your computer:

- Python 3.
- Python package manager **miniconda** or **pip**.
- Integrated Development Environment (IDE) (Jupyter Notebook; RStudio; VS Code; Emacs; etc.)

I will be using VS Code in class.

Readability is important! Check your Python coding styles against the recommended styles: https://peps.python.org/pep-0008/. A good place to start is the Section on "Code Lay-out".

Online books on Python for data science:

- "Python Data Science Handbook: Essential Tools for Working with Data," First Edition, by Jake VanderPlas, O'Reilly Media, 2016.

2. “Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython.” Third Edition, by Wes McK- inney, O’Reilly Media, 2022.

1.4 Data Science Ethics

1.4.1 Introduction

Ethics in data science is a fundamental consideration throughout the lifecycle of any project. Data science ethics refers to the principles and practices that guide responsible and fair use of data to ensure that individual rights are respected, societal welfare is prioritized, and harmful outcomes are avoided. Ethical frameworks like the Belmont Report (Protection of Human Subjects of Biomedical & Research, 1979)} and regulations such as the Health Insurance Portability and Accountability Act (HIPAA) (Health & Services, 1996) have established foundational principles that inspire ethical considerations in research and data use. This section explores key principles of ethical data science and provides guidance on implementing these principles in practice.

1.4.2 Principles of Ethical Data Science

1.4.2.1 Respect for Privacy

Safeguarding privacy is critical in data science. Projects should comply with data protection regulations, such as the General Data Protection Regulation (GDPR) or the California Consumer Privacy Act (CCPA). Techniques like anonymization and pseudonymization must be applied to protect sensitive information. Beyond legal compliance, data scientists should consider the ethical implications of using personal data.

The principles established by the Belmont Report emphasize respect for persons, which aligns with safeguarding individual privacy. Protecting privacy also involves limiting data collection to what is strictly necessary. Minimizing the use of identifiable information and implementing secure data storage practices are essential steps. Transparency about how data is used further builds trust with stakeholders.

1.4.2.2 Commitment to Fairness

Bias can arise at any stage of the data science pipeline, from data collection to algorithm development. Ethical practice requires actively identifying and addressing biases to prevent harm to underrepresented groups. Fairness should guide the design and deployment of models, ensuring equitable treatment across diverse populations.

To achieve fairness, data scientists must assess datasets for representativeness and use tools to detect potential biases. Regular evaluation of model outcomes against fairness metrics helps ensure that systems remain non-discriminatory. The Americans with Disabilities Act (ADA) (Congress, 1990) provides a legal framework emphasizing equitable access, which can inspire fairness in algorithmic design. Collaborating with domain experts and stakeholders can provide additional insights into fairness issues.

1.4.2.3 Emphasis on Transparency

Transparency builds trust and accountability in data science. Models should be interpretable, with clear documentation explaining their design, assumptions, and decision-making processes. Data scientists must communicate results in a way that stakeholders can understand, avoiding unnecessary complexity or obfuscation.

Transparent practices include providing stakeholders access to relevant information about model performance and limitations. The Federal Data

Strategy (Team, 2019) calls for transparency in public sector data use, offering inspiration for practices in broader contexts. Visualizing decision pathways and using tools like LIME or SHAP can enhance interpretability. Establishing clear communication protocols ensures that non-technical audiences can engage with the findings effectively.

1.4.2.4 Focus on Social Responsibility

Data science projects must align with ethical goals and anticipate their broader societal and environmental impacts. This includes considering how outputs may be used or misused and avoiding harm to vulnerable populations. Data scientists should aim to use their expertise to promote public welfare, addressing critical societal challenges such as health disparities, climate change, and education access.

Engaging with diverse perspectives helps align projects with societal values. Ethical codes, such as those from the Association for Computing Machinery (ACM) (Computing Machinery (ACM), 2018), offer guidance on using technology for social good. Collaborating with policymakers and community representatives ensures that data-driven initiatives address real needs and avoid unintended consequences. Regular impact assessments help measure whether projects meet their ethical objectives.

1.4.2.5 Adherence to Professional Integrity

Professional integrity underpins all ethical practices in data science. Adhering to established ethical guidelines, such as those from the American Statistical Association (ASA) ((ASA), 2018), ensures accountability. Practices like maintaining informed consent, avoiding data manipulation, and upholding rigor in analyses are essential for maintaining public trust in the field.

Ethical integrity also involves fostering a culture of honesty and openness within data science teams. Peer review and independent validation of findings can help identify potential errors or biases. Documenting methodologies and maintaining transparency in reporting further strengthen trust.

1.4.3 Ensuring Ethics in Practice

1.4.3.1 Building Ethical Awareness

Promoting ethical awareness begins with education and training. Institutions should integrate ethics into data science curricula, emphasizing real-world scenarios and decision-making. Organizations should conduct regular training to ensure their teams remain informed about emerging ethical challenges.

Workshops and case studies can help data scientists understand the complexities of ethical decision-making. Providing access to resources, such as ethical guidelines and tools, supports continuous learning. Leadership support is critical for embedding ethics into organizational culture.

1.4.3.2 Embedding Ethics in Workflows

Ethics must be embedded into every stage of the data science pipeline. Establishing frameworks for ethical review, such as ethics boards or peer-review processes, helps identify potential issues early. Tools for bias detection, explainability, and privacy protection should be standard components of workflows.

Standard operating procedures for ethical reviews can formalize the consideration of ethics in project planning. Developing templates for documenting ethical decisions ensures consistency and accountability. Collaboration across teams enhances the ability to address ethical challenges comprehensively.

1.4.3.3 Establishing Accountability Mechanisms

Clear accountability mechanisms are essential for ethical governance. This includes maintaining documentation for all decisions, establishing audit trails, and assigning responsibility for the outputs of data-driven systems. Organizations should encourage open dialogue about ethical concerns and support whistleblowers who raise issues.

Periodic audits of data science projects help ensure compliance with ethical standards. Organizations can benefit from external reviews to identify blind spots and improve their practices. Accountability fosters trust and aligns teams with ethical objectives.

1.4.3.4 Engaging Stakeholders

Ethical data science requires collaboration with diverse stakeholders. Including perspectives from affected communities, policymakers, and interdisciplinary experts ensures that projects address real needs and avoid unintended consequences. Stakeholder engagement fosters trust and aligns projects with societal values.

Public consultations and focus groups can provide valuable feedback on the potential impacts of data science projects. Engaging with regulators and advocacy groups helps align projects with legal and ethical expectations. Transparent communication with stakeholders builds long-term relationships.

1.4.3.5 Continuous Improvement

Ethics in data science is not static; it evolves with technology and societal expectations. Continuous improvement requires regular review of ethical practices, learning from past projects, and adapting to new challenges.

Organizations should foster a culture of reflection and growth to remain aligned with ethical best practices.

Establishing mechanisms for feedback on ethical practices can identify areas for development. Sharing lessons learned through conferences and publications helps the broader community advance its understanding of ethics in data science.

1.4.4 Conclusion

Data science ethics is a dynamic and integral aspect of the discipline. By adhering to principles of privacy, fairness, transparency, social responsibility, and integrity, data scientists can ensure their work contributes positively to society. Implementing these principles through structured workflows, stakeholder engagement, and continuous improvement establishes a foundation for trustworthy and impactful data science.

2 Project Management

Many tutorials are available in different formats. Here is a YouTube video "Git and GitHub for Beginners — Crash Course' '. The video also covers GitHub, a cloud service for Git which provides a cloud back up of your work and makes collaboration with co-workers easy. Similar services are, for example, bitbucket and GitLab.

There are tools that make learning Git easy.

- Here is a collection of online Git exersices that I used for Git training in other courses that I taught.
- Here is a game called `Oh My Git`, an open source game about learning Git!

2.1 Set Up Git/GitHub

Download Git if you don't have it already.

To set up GitHub (other services like Bitbucket or GitLab are similar), you need to

- Generate an SSH key if you don't have one already.
- Sign up an GitHub account.
- Add the SSH key to your GitHub account

See how to get started with GitHub account.

2.2 Most Frequently Used Git Commands

- `git clone`:
  - Used to clone a repository to a local folder.
  - Requires either HTTPS link or SSH key to authenticate.
- `git pull`:
  - Downloads any updates made to the remote repository and automatically updates the local repository.
- `git status`:
  - Returns the state of the working directory.
  - Lists the files that have been modified, and are yet to be or have been staged and/or committed.
  - Shows if the local repository is begind or ahead a remote branch.
- `git add`:
  - Adds new or modified files to the Git staging area.
  - Gives the option to select which files are to be sent to the remote repository
- `git rm`:
  - Used to remove files from the staging index or the local repository.
- `git commit`:
  - Commits changes made to the local repository and saves it like a snapshot.
  - A message is recommended with every commit to keep track of changes made.
- `git push`:
  - Used to send commits made on local repository to the remote repository.

2.3 Tips on using Git:

- Use the command line interface instead of the web interface (e.g., upload on GitHub)
- Make frequent small commits instead of rare large commits.
- Make commit messages informative and meaningful.
- Name your files/folders by some reasonable convention.
  - Lower cases are better than upper cases.
  - No blanks in file/folder names.
- Keep the repo clean by not tracking generated files.
- Creat a `.gitignore` file for better output from `git status`.
- Keep the linewidth of sources to under 80 for better `git diff` view.

2.4 Pull Request

To contribute to an open source project (e.g., our classnotes), use pull requests. Pull requests "let you tell others about changes you've pushed to a branch in a repository on GitHub. Once a pull request is opened, you can discuss and review the potential changes with collaborators and add follow-up commits before your changes are merged into the base branch."

Watch this YouTube video: GitHub pull requests in 100 seconds.

3 Reproducible Data Science

Data science projects should be reproducible to be trustworthy. Dynamic documents facilitate reproducibility. Quarto is an open-source dynamic document preparation system, ideal for scientific and technical publishing. From the official websites, Quarto can be used to:

- Create dynamic content with Python, R, Julia, and Observable.
- Author documents as plain text markdown or Jupyter notebooks.
- Publish high-quality articles, reports, presentations, websites, blogs, and books in HTML, PDF, MS Word, ePub, and more.
- Author with scientific markdown, including equations, citations, cross references, figure panels, callouts, advanced layout, and more.

3.1 Introduction to Quarto

To get started with Quarto, see documentation at Quarto.

For a clean style, I suggest that you use `VS Code` as your IDE. The `ipynb` files have extra formats in plain texts, which are not as clean as `qmd` files. There are, of course, tools to convert between the two representations of a notebook. For example:

```
quarto convert hello.ipynb # converts to qmd
quarto convert hello.qmd   # converts to ipynb
```

We will use Quarto for homework assignments, classnotes, and presentations. You will see them in action through in-class demonstrations. The following sections in the Quarto Guide are immediately useful.

- Markdown basics
- Using Python
- Presentations

A template for homework is in this repo (`hwtemp.qmd`) to get you started with homework assignments.

3.2 Compiling the Classnotes

The sources of the classnotes are at https://github.com/statds/ids-f24. This is also the source tree that you will contributed to this semester. I expect that you clone the repository to your own computer, update it frequently, and compile the latest version on your computer (reproducibility).

To compile the classnotes, you need the following tools: Git, Quarto, and Python.

3.2.1 Set up your Python Virtual Environment

I suggest that a Python virtual environment for the classnotes be set up in the current directory for reproducibility. A Python virtual environment is simply a directory with a particular file structure, which contains a specific Python interpreter and software libraries and binaries needed to support a project. It allows us to isolate our Python development projects from our system installed Python and other Python environments.

To create a Python virtual environment for our classnotes:

```
python3 -m venv .ids-f24-venv
```

Here `.ids-f24-venv` is the name of the virtual environment to be created. Choose an informative name. This only needs to be set up once.

To activate this virtual environment:

```
. .ids-f24-venv/bin/activate
```

After activating the virtual environment, you will see `(.ids-f24-venv)` at the beginning of your shell prompt. Then, the Python interpreter and packages needed will be the local versions in this virtual environment without interfering your system-wide installation or other virtual environments.

To install the Python packages that are needed to compile the classnotes, we have a `requirements.txt` file that specifies the packages and their versions. They can be installed easily with:

```
pip install -r requirements.txt
```

If you are interested in learning how to create the `requirements.txt` file, just put your question into a Google search.

To exit the virtual environment, simply type `deactivate` in your command line. This will return you to your system's global Python environment.

3.2.2 Clone the Repository

Clone the repository to your own computer. In a terminal (command line), go to an appropriate directory (folder), and clone the repo. For example, if you use `ssh` for authentication:

```
git clone git@github.com:statds/ids-f24.git
```

3.2.3 Render the Classnotes

Assuming `quarto` has been set up, we render the classnotes in the cloned repository

```
cd ids-f24
quarto render
```

If there are error messages, search and find solutions to clear them. Otherwise, the html version of the notes will be available under `_book/index.html`, which is default location of the output.

4 Python Refreshment

4.1 Know Your Computer

4.1.1 Operating System

Your computer has an operating system (OS), which is responsible for managing the software packages on your computer. Each operating system has its own package management system. For example:

- **Linux:** Linux distributions have a variety of package managers depending on the distribution. For instance, Ubuntu uses APT (Advanced Package Tool), Fedora uses DNF (Dandified Yum), and Arch Linux uses Pacman. These package managers are integral to the Linux experience, allowing users to install, update, and manage software packages easily from repositories.
- **macOS:** macOS uses Homebrew as its primary package manager. Homebrew simplifies the installation of software and tools that aren't included in the standard macOS installation, using simple commands in the terminal.
- **Windows:** Windows users often rely on the Microsoft Store for apps and software. For more developer-focused package management, tools like Chocolatey and Windows Package Manager (Winget) are used. Additionally, recent versions of Windows have introduced the Windows Subsystem for Linux (WSL). WSL allows Windows users to run a Linux environment directly on Windows, unifying Windows

and Linux applications and tools. This is particularly useful for developers and data scientists who need to run Linux-specific software or scripts. It saves a lot of trouble Windows users used to have before its time.

Understanding the package management system of your operating system is crucial for effectively managing and installing software, especially for data science tools and applications.

4.1.2 File System

A file system is a fundamental aspect of a computer's operating system, responsible for managing how data is stored and retrieved on a storage device, such as a hard drive, SSD, or USB flash drive. Essentially, it provides a way for the OS and users to organize and keep track of files. Different operating systems typically use different file systems. For instance, NTFS and FAT32 are common in Windows, APFS and HFS+ in macOS, and Ext4 in many Linux distributions. Each file system has its own set of rules for controlling the allocation of space on the drive and the naming, storage, and access of files, which impacts performance, security, and compatibility. Understanding file systems is crucial for tasks such as data recovery, disk partitioning, and managing file permissions, making it an important concept for anyone working with computers, especially in data science and IT fields.

Navigating through folders in the command line, especially in Unix-like environments such as Linux or macOS, and Windows Subsystem for Linux (WSL), is an essential skill for effective file management. The command cd (change directory) is central to this process. To move into a specific directory, you use `cd` followed by the directory name, like `cd Documents`. To go up one level in the directory hierarchy, you use `cd ..`. To return to the home directory, simply typing `cd` or `cd ~` will suffice. The `ls` command lists all files and folders in the current directory, providing a clear view of your options for navigation. Mastering these commands,

along with others like `pwd` (print working directory), which displays your current directory, equips you with the basics of moving around the file system in the command line, an indispensable skill for a wide range of computing tasks in Unix-like systems.

You have programmed in Python. Regardless of your skill level, let us do some refreshing.

4.2 The Python World

- Function: a block of organized, reusable code to complete certain task.
- Module: a file containing a collection of functions, variables, and statements.
- Package: a structured directory containing collections of modules and an `__init.py__` file by which the directory is interpreted as a package.
- Library: a collection of related functionality of codes. It is a reusable chunk of code that we can use by importing it in our program, we can just use it by importing that library and calling the method of that library with period(.).

See, for example, how to build a Python libratry.

4.3 Standard Library

Python's has an extensive standard library that offers a wide range of facilities as indicated by the long table of contents listed below. See documentation online.

> The library contains built-in modules (written in C) that provide access to system functionality such as file I/O that would otherwise be inaccessible to Python programmers, as well as modules written in Python that provide standardized solutions for many problems that occur in everyday programming. Some of these modules are explicitly designed to encourage and enhance the portability of Python programs by abstracting away platform-specifics into platform-neutral APIs.

Question: How to get the constant e to an arbitary precision?

The constant is only represented by a given double precision.

```
import math
print("%0.20f" % math.e)
print("%0.80f" % math.e)
```

```
2.71828182845904509080
2.71828182845904509079559829842764884233474731445312500000000000000000000000
```

Now use package `decimal` to export with an arbitary precision.

```
import decimal  # for what?

## set the required number digits to 150
decimal.getcontext().prec = 150
decimal.Decimal(1).exp().to_eng_string()
decimal.Decimal(1).exp().to_eng_string()[2:]
```

```
'718281828459045235360287471352662497757247093699959574966967627724076630353
```

4.4 Important Libraries

- NumPy
- pandas
- matplotlib
- IPython/Jupyter
- SciPy
- scikit-learn
- statsmodels

Question: how to draw a random sample from a normal distribution and evaluate the density and distributions at these points?

```
from scipy.stats import norm

mu, sigma = 2, 4
mean, var, skew, kurt = norm.stats(mu, sigma, moments='mvsk')
print(mean, var, skew, kurt)
x = norm.rvs(loc = mu, scale = sigma, size = 10)
x
```

```
2.0 16.0 0.0 0.0
```

```
array([ 2.79998665,  5.82505877, -0.03173577,  6.31901339, 10.27178137,
       -0.15176202, -5.57599415,  6.50619806,  2.69651893, -1.51032714])
```

The pdf and cdf can be evaluated:

```
norm.pdf(x, loc = mu, scale = sigma)
```

```
array([0.09776074, 0.06313664, 0.08766511, 0.0556782 , 0.01175553,
       0.08630024, 0.01659178, 0.05287684, 0.09823492, 0.06786005])
```

4.5 Writing a Function

Consider the Fibonacci Sequence $1, 1, 2, 3, 5, 8, 13, 21, 34, ...$ The next number is found by adding up the two numbers before it. We are going to use 3 ways to solve the problems.

The first is a recursive solution.

```
def fib_rs(n):
    if (n==1 or n==2):
        return 1
    else:
        return fib_rs(n - 1) + fib_rs(n - 2)

%timeit fib_rs(10)
```

```
10.4  s ± 617 ns per loop (mean ± std. dev. of 7 runs, 100,000 loops each)
```

The second uses dynamic programming memoization.

```
def fib_dm_helper(n, mem):
    if mem[n] is not None:
        return mem[n]
    elif (n == 1 or n == 2):
        result = 1
    else:
        result = fib_dm_helper(n - 1, mem) + fib_dm_helper(n - 2, mem)
    mem[n] = result
    return result

def fib_dm(n):
    mem = [None] * (n + 1)
    return fib_dm_helper(n, mem)
```

```
%timeit fib_dm(10)
```

```
2.2 s ± 37.4 ns per loop (mean ± std. dev. of 7 runs, 100,000 loops each)
```

The third is still dynamic programming but bottom-up.

```
def fib_dbu(n):
    mem = [None] * (n + 1)
    mem[1] = 1;
    mem[2] = 1;
    for i in range(3, n + 1):
        mem[i] = mem[i - 1] + mem[i - 2]
    return mem[n]


%timeit fib_dbu(500)
```

```
78.3 s ± 5.31 s per loop (mean ± std. dev. of 7 runs, 10,000 loops each)
```

Apparently, the three solutions have very different performance for larger `n`.

4.5.1 Monty Hall

Here is a function that performs the Monty Hall experiments.

```
import numpy as np

def montyhall(ndoors, ntrials):
    doors = np.arange(1, ndoors + 1) / 10
```

```
    prize = np.random.choice(doors, size=ntrials)
    player = np.random.choice(doors, size=ntrials)
    host = np.array([np.random.choice([d for d in doors
                                       if d not in [player[x], prize[x]]])
                     for x in range(ntrials)])
    player2 = np.array([np.random.choice([d for d in doors
                                          if d not in [player[x], host[x]]])
                        for x in range(ntrials)])
    return {'noswitch': np.sum(prize == player), 'switch': np.sum(prize == pl
```

Test it out:

```
montyhall(3, 1000)
montyhall(4, 1000)
```

```
{'noswitch': np.int64(252), 'switch': np.int64(361)}
```

The true value for the two strategies with n doors are, respectively, $1/n$ and $\frac{n-1}{n(n-2)}$.

4.6 Variables versus Objects

In Python, variables and the objects they point to actually live in two different places in the computer memory. Think of variables as pointers to the objects they're associated with, rather than being those objects. This matters when multiple variables point to the same object.

```
x = [1, 2, 3]  # create a list; x points to the list
y = x          # y also points to the same list in the memory
y.append(4)    # append to y
x              # x changed!
```

```
[1, 2, 3, 4]
```

Now check their addresses

```
print(id(x))   # address of x
print(id(y))   # address of y
```

```
4364181312
4364181312
```

Nonetheless, some data types in Python are "immutable", meaning that their values cannot be changed in place. One such example is strings.

```
x = "abc"
y = x
y = "xyz"
x
```

```
'abc'
```

Now check their addresses

```
print(id(x))   # address of x
print(id(y))   # address of y
```

```
4311943808
4395323472
```

Question: What's mutable and what's immutable?

Anything that is a collection of other objects is mutable, except `tuples`.

Not all manipulations of mutable objects change the object rather than create a new object. Sometimes when you do something to a mutable object, you get back a new object. Manipulations that change an existing object, rather than create a new one, are referred to as "in-place mutations" or just "mutations." So:

- **All** manipulations of immutable types create new objects.
- **Some** manipulations of mutable types create new objects.

Different variables may all be pointing at the same object is preserved through function calls (a behavior known as "pass by object-reference"). So if you pass a list to a function, and that function manipulates that list using an in-place mutation, that change will affect any variable that was pointing to that same object outside the function.

```
x = [1, 2, 3]
y = x

def append_42(input_list):
    input_list.append(42)
    return input_list

append_42(x)
```

```
[1, 2, 3, 42]
```

Note that both `x` and `y` have been appended by 42.

4.7 Number Representation

Numers in a computer's memory are represented by binary styles (on and off of bits).

4.7.1 Integers

If not careful, It is easy to be bitten by overflow with integers when using Numpy and Pandas in Python.

```
import numpy as np

x = np.array(2 ** 63 - 1 , dtype = 'int')
x
# This should be the largest number numpy can display, with
# the default int8 type (64 bits)
```

```
array(9223372036854775807)
```

Note: on Windows and other platforms, ***`dtype = 'int'`*** *may have to be changed to* ***`dtype = np.int64`*** *for the code to execute. Source: Stackoverflow*

What if we increment it by 1?

```
y = np.array(x + 1, dtype = 'int')
y
# Because of the overflow, it becomes negative!
```

```
array(-9223372036854775808)
```

For vanilla Python, the overflow errors are checked and more digits are allocated when needed, at the cost of being slow.

```
2 ** 63 * 1000
```

```
9223372036854775808000
```

This number is 1000 times larger than the prior number, but still displayed perfectly without any overflows

4.7.2 Floating Number

Standard double-precision floating point number uses 64 bits. Among them, 1 is for sign, 11 is for exponent, and 52 are fraction significand, See https://en.wikipedia.org/wiki/Double-precision_floating-point_format. The bottom line is that, of course, not every real number is exactly representable.

If you have played the Game 24, here is a tricky one:

```
8 / (3 - 8 / 3) == 24
```

```
False
```

Surprise?

There are more.

```
0.1 + 0.1 + 0.1 == 0.3
```

```
False
```

```
0.3 - 0.2 == 0.1
```

```
False
```

What is really going on?

```
import decimal
decimal.Decimal(0.1)
```

```
Decimal('0.1000000000000000055511151231257827021181583404541015625')
```

```
decimal.Decimal(8 / (3 - 8 / 3))
```

```
Decimal('23.999999999999989341858963598497211933135986328125')
```

Because the mantissa bits are limited, it can not represent a floating point that's both very big and very precise. Most computers can represent all integers up to 2^{53}, after that it starts skipping numbers.

```
2.1 ** 53 + 1 == 2.1 ** 53

# Find a number larger than 2 to the 53rd
```

```
True
```

```
x = 2.1 ** 53
for i in range(1000000):
    x = x + 1
x == 2.1 ** 53
```

```
True
```

We add 1 to `x` by 1000000 times, but it still equal to its initial value, `2.1 ** 53`. This is because this number is too big that computer can't handle it with precision like add 1.

Machine epsilon is the smallest positive floating-point number `x` such that `1 + x != 1`.

```
print(np.finfo(float).eps)
print(np.finfo(np.float32).eps)
```

```
2.220446049250313e-16
1.1920929e-07
```

4.8 Virtual Environment

Virtual environments in Python are essential tools for managing dependencies and ensuring consistency across projects. They allow you to create isolated environments for each project, with its own set of installed packages, separate from the global Python installation. This isolation prevents conflicts between project dependencies and versions, making your projects more reliable and easier to manage. It's particularly useful when working on multiple projects with differing requirements, or when collaborating with others who may have different setups.

To set up a virtual environment, you first need to ensure that Python is installed on your system. Most modern Python installations come with the venv module, which is used to create virtual environments. Here's how to set one up:

- Open your command line interface.
- Navigate to your project directory.

- Run `python3 -m venv myenv`, where `myenv` is the name of the virtual environment to be created. Choose an informative name.

This command creates a new directory named `myenv` (or your chosen name) in your project directory, containing the virtual environment.

To start using this environment, you need to activate it. The activation command varies depending on your operating system:

- On Windows, run `myenv\Scripts\activate`.
- On Linux or MacOS, use `source myenv/bin/activate` or `. myenv/bin/activate`.

Once activated, your command line will typically show the name of the virtual environment, and you can then install and use packages within this isolated environment without affecting your global Python setup.

To exit the virtual environment, simply type `deactivate` in your command line. This will return you to your system's global Python environment.

As an example, let's install a package, like `numpy`, in this newly created virtual environment:

- Ensure your virtual environment is activated.
- Run `pip install numpy`.

This command installs the requests library in your virtual environment. You can verify the installation by running `pip list`, which should show requests along with its version.

5 Exercises

1. **Quarto and Git setup** Quarto and Git are two important tools for data science. Get familiar with them through the following tasks. Please use the `templates/hw.qmd` template to document, for each step, what you did, the obstacles you encountered, and how you overcame them. Think of this as a user manual for students who are new to this. Use the command line interface.

    a. Set up SSH authentication between your computer and your GitHub account.
    b. Install Quarto onto your computer following the instructions of Get Started.
    c. Pick a tool of your choice (e.g., VS Code, Jupyter Notebook, Emacs, etc.), follow the instructions to reproduce the example of line plot on polar axis.
    d. Render the homework into a pdf file and put the file into a release in your GitHub repo.

2. **Working on Homework Problems**

    a. What are the differences between binary and source files?
    b. Why do we not want to track binary files in a repo?
    c. Why do I require pdf output via release?
    d. Why do I not want your files added via 'upload'?
    e. Why do I require line width under 80?
    f. Why is it not a good idea to have spaces in file/folder names?

3. **Contributing to the Class Notes** To contribute to the classnotes, you need to have a working copy of the sources on your computer. Document the following steps in a `qmd` file in the form of a step-by-step manual, as if you are explaining them to someone who wants to contribute too. Make at least 10 commits for this task, each with an informativ emessage.

    a. Create a fork of the notes repo into your own GitHub account.
    b. Clone it to an appropriate folder on your computer.
    c. Render the classnotes on your computer; document the obstacles and solutions.
    d. Make a new branch (and name it appropriately) to experiment with your changes.
    e. Checkout your branch and add your wishes to the wish list; commit with an informative message; and push the changes to your GitHub account.
    f. Make a pull request to class notes repo from your fork at GitHub. Make sure you have clear messages to document the changes.

4. **Monty Hall** Consider a generalized Monty Hall experiment. Suppose that the game start with n doors; after you pick one, the host opens $m \leq n-2$ doors, that show no award. Include sufficient text around the code chunks to explain them.

    a. Write a function to simulate the experiment once. The function takes two arguments `ndoors` and `nempty`, which represent the number of doors and the number of empty doors showed by the host, respectively, It returns the result of two strategies, switch and no-switch, from playing this game.
    b. Play this game with 3 doors and 1 empty a few times.
    c. Play this game with 10 doors and 8 empty a few times.
    d. Write a function to play this game `ntrial` times and return the proportion of wins for both strategies.

e. Apply your function to play this game 1000 times, with 3 doors and 10 doors, and summarize your results.
f. Write a function to demonstrate the Monty Hall problem through simulation. The function takes two arguments `ndoors` and `ntrials`, representing the number of doors in the experiment and the number of trials in a simulation, respectively. The function should return the proportion of wins for both the switch and no-switch strategy.
g. Apply your function with 3 doors and 5 doors, both with 1000 trials. Summarize your results.

5. **Approximating** π Write a function to do a Monte Carlo approximation of π. The function takes a Monte Carlo sample size `n` as input, and returns a point estimate of π and a 95% confidence interval. Apply your function with sample size 1000, 2000, 4000, and 8000. Repeat the experiment 1000 times for each sample size and check the empirical probability that the confidence intervals cover the true value of π. Comment on the results.

6. **Google Billboard Ad** Find the first 10-digit prime number occurring in consecutive digits of e. This was a Google recruiting ad.

7. **Game 24** The math game 24 is one of the addictive games among number lovers. With four randomly selected cards form a deck of poker cards, use all four values and elementary arithmetic operations $(+ - \times /)$ to come up with 24. Let $\square$ be one of the four numbers. Let $\bigcirc$ represent one of the four operators. For example,

$$(\square \bigcirc \square) \bigcirc (\square \bigcirc \square)$$

is one way to group the the operations.

a. List all the possible ways to group the four numbers.
b. How many possible ways are there to check for a solution?

c. Write a function to solve the problem in a brutal force way. The inputs of the function are four numbers. The function returns a list of solutions. Some of the solutions will be equivalent, but let us not worry about that for now.

8. **NYC Crash Data Cleaning** The NYC motor vehicle collisions data with documentation is available from NYC Open Data. The raw data needs some cleaning.

   a. Use the filter from the website to download the crash data of the week of June 30, 2024 in CSV format; save it under a directory `data` with an informative name (e.g., `nyccrashes_2024w0630_by20240916.csv`); read the data into a Panda data frame with careful handling of the date time variables.
   b. Clean up the variable names. Use lower cases and replace spaces with underscores.
   c. Get the basic summaries of each variables: missing percentage; descriptive statistics for continuous variables; frequency tables for discrete variables.
   d. Are their invalid `longitude` and `latitude` in the data? If so, replace them with `NA`.
   e. Are there `zip_code` values that are not legit NYC zip codes? If so, replace them with `NA`.
   f. Are there missing in `zip_code` and `borough`? Do they always co-occur?
   g. Are there cases where `zip_code` and `borough` are missing but the geo codes are not missing? If so, fill in `zip_code` and `borough` using the geo codes.
   h. Is it redundant to keep both `location` and the `longitude`/`latitude` at the NYC Open Data server?
   i. Check the frequency of `crash_time` by hour. Is there a matter of bad luck at exactly midnight? How would you interpret this?
   j. Are the number of persons killed/injured the summation of the

numbers of pedestrians, cyclist, and motorists killed/injured? If so, is it redundant to keep these two columns at the NYC Open Data server?

k. Print the whole frequency table of `contributing_factor_vehicle_1`. Convert lower cases to uppercases and check the frequencies again.

l. Provided an opportunity to meet the data provider, what suggestions would you make based on your data exploration experience?

9. **NYC Crash Data Exploration** Except for the first question, use the cleaned crash data in feather format.

   a. Construct a contigency table for missing in geocode (latitude and longitude) by borough. Is the missing pattern the same across boroughs? Formulate a hypothesis and test it.

   b. Construct a `hour` variable with integer values from 0 to 23. Plot the histogram of the number of crashes by `hour`. Plot it by borough.

   c. Overlay the locations of the crashes on a map of NYC. The map could be a static map or Google map.

   d. Create a new variable `severe` which is one if the number of persons injured or deaths is 1 or more; and zero otherwise. Construct a cross table for `severe` versus borough. Is the severity of the crashes the same across boroughs? Test the null hypothesis that the two variables are not associated with an appropriate test.

   e. Merge the crash data with the zip code database.

   f. Fit a logistic model with `severe` as the outcome variable and covariates that are available in the data or can be engineered from the data. For example, zip code level covariates can be obtained by merging with the zip code database; crash hour; number of vehicles involved.

10. **NYC Crash severity modeling** Using the cleaned NYC crash

data, merged with zipcode level information, predict `severe` of a crash.

a. Set random seed to `1234`. Randomly select 20% of the crashes as testing data and leave the rest 80% as training data.
b. Fit a logistic model on the training data and validate the performance on the testing data. Explain the confusion matrix result from the testing data. Compute the F1 score.
c. Fit a logistic model on the training data with L_1 regularization. Select the tuning parameter with 5-fold cross-validation in F1 score
d. Apply the regularized logistic regression to predict the severity of the crashes in the testing data. Compare the performance of the two logistic models in terms of accuracy, precision, recall, F1-score, and AUC.

11. **Midterm project: Noise complaints in NYC** The NYC Open Data of 311 Service Requests contains all requests from 2010 to present. We consider a subset of it with requests to NYPD on noise complaints that are created between 00:00:00 06/30/2024 and 24:00:00 07/06/2024. The subset is available in CSV format as `data/nypd311w063024noise_by100724.csv`. Read the data dictionary online to understand the meaning of the variables.

    a. Data cleaning.
        i. Import the data, rename the columns with our preferred styles.
        ii. Summarize the missing information. Are there variables that are close to completely missing?
        iii. Are there redundant information in the data? Try storing the data using the Arrow format and comment on the efficiency gain.
        iv. Are there invalid NYC zipcode or borough? Justify and clean them if yes.

v. Are there date errors? Examples are earlier `closed_date` than `created_date`; `closed_date` and `created_date` matching to the second; dates exactly at midnight or noon to the second; `action_update_date` after `closed_date`.
vi. Summarize your suggestions to the data curator in several bullet points.

b. Data exploration.

i. If we suspect that response time may depend on the time of day when a complaint is made, we can compare the response times for complaints submitted during nighttime and daytime. To do this, we can visualize the comparison by complaint type, borough, and weekday (vs weekend/holiday).
ii. Perform a formal hypothesis test to confirm the observations from your visualization. Formally state your hypotheses and summarize your conclusions in plain English.
iii. Create a binary variable `over2h` to indicate that a service request took two hours or longer to close.
iv. Does `over2h` depend on the complaint type, borough, or weekday (vs weekend/holiday)? State your hypotheses and summarize your conclusions in plain English.

c. Data analysis.

i. The addresses of NYC police precincts are stored in `data/nypd_precincts.csv`. Use geocoding tools to find their geocode (longitude and latitude) from the addresses.
ii. Create a variable `dist2pp` which represent the distance from each request incidence to the nearest police precinct.
iii. Create zip code level variables by merging with data from package `uszipcode`.
iv. Randomly select 20% of the complaints as testing data with seeds `1234`. Build a logistic model to predict `over2h` for the noise complaints with the training data, using all the variables you can engineer from the available data. If you

have tuning parameters, justify how they were selected.

v. Assess the performance of your model in terms of commonly used metrics. Summarize your results to a New Yorker who is not data science savvy.

References

Agonafir, C., Lakhankar, T., Khanbilvardi, R., Krakauer, N., Radell, D., & Devineni, N. (2022). A machine learning approach to evaluate the spatial variability of New York City's 311 street flooding complaints. *Computers, Environment and Urban Systems*, *97*, 101854.

Agonafir, C., Pabon, A. R., Lakhankar, T., Khanbilvardi, R., & Devineni, N. (2022). Understanding New York City street flooding through 311 complaints. *Journal of Hydrology*, *605*, 127300.

(ASA), A. S. A. (2018). *Ethical guidelines for statistical practice*.

Computing Machinery (ACM), A. for. (2018). *Code of ethics and professional conduct*.

Congress, U. S. (1990). *Americans with disabilities act of 1990 (ADA)*.

Health, U. S. D. of, & Services, H. (1996). *Health insurance portability and accountability act of 1996 (HIPAA)*.

Protection of Human Subjects of Biomedical, N. C. for the, & Research, B. (1979). *The belmont report: Ethical principles and guidelines for the protection of human subjects of research*.

Team, F. D. S. D. (2019). *Federal data strategy 2020 action plan*.

VanderPlas, J. (2016). *Python data science handbook: Essential tools for working with data*. O'Reilly Media, Inc.