Introduction to Data Science

STAT 3255/5255 @ UConn, Spring 2025

Jun Yan and Students in Spring 2025

2025-01-21

Table of contents

Table of contents

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 Fall 2024 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-f24.

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 in exploring the lecture notes from the previous years, the 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.
- Set up a Python virtual environment in the root folder of the source.
- Install all the packages specified in `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

NYC noise complaints made to NYPD in the week of July 4, 2024. See details in the exercises.

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
- Kaggle
- DrivenData
- Top 10 Data Science Competitions in 2024

If you work on sports analytics, you are welcome to submit a poster to UConn Sports Analytics Symposium (UCSAS) 2024.

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
  - 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

- Ferris, Spencer
- Gogoj, Anatol Mateusz
- Mundiwala, Mohammad Moiz
- Vellore, Ajeeth Krishna
- Zhang, Gaofei

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
```

```
['Schittina,Thomas',
 'Mundiwala,Mohammad Moiz',
 'Jun,Joann',
 '',
 'Li,Shiyi',
 'Lagutin,Vladislav',
 'Lin,Selena',
 'Ferris,Spencer',
 'Long,Ethan Kenneth',
 'Alsadadi,Ammar Shaker',
 'Tomaino,Mario Anthony',
 'Chen,Yifei',
 'Tamhane,Shubhan',
 'Lang,Lang',
 'Symula,Sebastian',
 'Pfeifer,Nicholas Theodore',
 'Zhang,Gaofei',
 'El Zein,Amer Hani',
 'Schlessel,Jacob E',
 'Hutchins,Isabella Grace',
 'Roy,Luke William',
 'Febles,Xavier Milan',
 'Ackerman,John',
 'Horn,Alyssa Noelle',
 'Kline,Daniel Esteban',
 'Vellore,Ajeeth Krishna',
 'Nasejje,Ruth Nicole',
 'Gogoj,Anatol Mateusz',
 'Reed,Kyle Daniel',
 '']
```

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.

Course Logistics

Presentation Task Board

Here are some example tasks:

- Making presentations with Quarto
- Data science ethics
- 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
- Calling C/C++ from Python

Preliminaries

- Calling R from Python and vice versa
- Developing a Python package

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
10/02	Julia Mazzola	Data Visu-aliZation with Plotnine
10/07	Suha Akach	Naive Bayes classifier
10/09	Rahul Manna	Animation

Date	Presenter	Topic
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
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 Section 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.
- Use `quarto` source only. See 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.

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

Preliminaries

- 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.

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 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.

    a. Install Quarto onto your computer following the instructions of Get Started. Document the obstacles you encountered and how you overcame them.
    b. 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.
    c. Render the homework into a pdf file and put the file into a release in your GitHub repo. Document any obstacles you have and how you overcome them.

2. **Git basics and GitHub setup** Learn the Git basics and set up an account on GitHub if you do not already have one. Practice the tips on Git in the notes. By going through the following tasks, ensure your repo has at least 10 commits, each with an informative message. Regularly check the status of your repo using `git status`. The specific tasks are:

    a. Clone the class notes repo to an appropriate folder on your computer.
    b. Add all the files to your designated homework repo from GitHub Classroom and work on that repo for the rest of the problem.
    c. Add your name and wishes to the Wishlist; commit.
    d. Remove the `Last, First` entry from the list; commit.

e. Create a new file called `add.qmd` containing a few lines of texts; commit.
f. Remove `add.qmd` (pretending that this is by accident); commit.
g. Recover the accidentally removed file `add.qmd`; add a long line (a paragraph without a hard break); add a short line (under 80 characters); commit.
h. Change one word in the long line and one word in the short line; use `git diff` to see the difference from the last commit; commit.
i. Play with other git operations and commit.

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 as if you are explaining them to someone who want to contribute too.

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

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