About this Book

Contents

Skip to main content

Welcome to the course manual for CSC310 at URI with Professor Brown.

This class meets TTh 3:30-4:45 in Pastore 259.

This website will contain the syllabus, class notes, and other reference material for the class.

Land University of Rhode Island Land Acknowledgment

> **Note**
>
> The University of Rhode Island land acknowledgment is a statement written by members of the University community in close partnership with members of the Narragansett Tribe. The statement recognizes and pays tribute to the people who lived on and stewarded the land on which the University now resides. The statement seeks to show gratitude and respect to Indigenous people and cultures and build community with the Narragansett Nation and other Native American tribes.

The University of Rhode Island occupies the traditional stomping ground of the Narragansett Nation and the Niantic People. We honor and respect the enduring and continuing relationship between the Indigenous people and this land by teaching and learning more about their history and present-day communities, and by becoming stewards of the land we, too, inhabit.

Skip to main content

Navigating the Sections

The Syllabus section has logistical operations for the course broken down into sections. You can also read straight through by starting in the first one and navigating to the next section using the arrow navigation at the end of the page.

This site is a resource for the course. We do not follow a text book for this course, but all notes from class are posted in the notes section, accessible on the left hand side menu, visible on large screens and in the menu on mobile.

The resources section has links and short posts that provide more context and explanation. Content in this section is for the most part not strictly the material that you'll be graded on, but it is often material that will help you understand and grow as a programmer and data scientist.

Reading each page

All class notes can be downloaded in multiple formats, including as a notebook. Some pages of the syllabus and resources are also notebooks, if you want to see behind the curtain of how I manage the course information.

Try it Yourself

Notes will have exercises marked like this

Question from Class

Questions that are asked in class, but unanswered at that time will be answered in the notes and marked with a box like this. Long answers will be in the main notes

Further reading

Notes that are mostly links to background and context will be highlighted like this. These are optional, but will mostly help you understand code excerpts they relate to.

Hint

Both notes and assignment pages will have hints from time to time. Pay attention to these on the notes, they'll typically relate to things that will appear in the assignment.

Think Ahead

Think ahead boxes will guide you to start thinking about what can go into your portfolio to build on the material at hand.

Skip to main content

Click here!

Special tips will be formatted like this

Check your Comprehension

Questions to use to check your comprehension will looklike this

About

About the topic

Data science exists at the intersection of computer science, statistics, and domain expertise. That means writing programs to access and manipulate data so that it becomes available for analysis using statistical and machine learning techniques is at the core of data science. Data scientists use their data and analytical ability to find and interpret rich data sources; manage large amounts of data despite hardware, software, and bandwidth constraints; merge data sources; ensure consistency of datasets; create visualizations to aid in understanding data; build mathematical models using the data; and present and communicate the data insights/findings.

About the goals and preparation

This course provides a survey of data science. Topics include data driven programming in Python; data sets, file formats and meta-data; descriptive statistics, data visualization, and foundations of predictive data modeling and machine learning; accessing web data and databases; distributed data management. You will work on weekly programming problems such as accessing data in database and visualize it or build machine learning models of a given data set.

Basic programming skills (CSC201 or CSC211) are a prerequisite to this course. This course is a prerequisite course to machine learning, where you learn how machine learning algorithms work. In this course, we will start with a very fast review of basic programming ideas, since you've already done that before. We will learn how to *use* machine learning algorithms to do data science, but not how to *build* machine learning algorithms, we'll use packages that implement the algorithms for us.

About the course

This course is designed to make you a better programmer while learning data science. You may be stronger in one of those areas than the other at the beginning, but you should grow in both areas by the end of the semester.

About this semester

This semester, I will be trying some new ways to update the course to reflect the reality that in a job, a lot of your work may be done with an AI assistant to help. That said, *learning* the basic material still has to happen, you cannot supervise an AI if you do not know what correct looks like.

Skip to main content

Each assignment will have specific AI use guidelines that you must follow in addition to general overall course style guide and requirements.

Additionally, the grading that we will use this semester will be new, if something does not make sense, ask! I will never change a policy in a way that could hurt a student who was acting in good faith (if you were trying to game thigns, I may close loopholes in ways that do not benefit you.)

About this syllabus

This syllabus is a *living* document and accessible from BrightSpace, as a pdf for download directly, and online at rhodyprog4ds.github.io/BrownFall24/syllabus. If you choose to download a copy of it, note that it is only a copy. You can get notification of changes from GitHub by "watching" the repository. You can view the date of changes and exactly what changes were made on the Github commits page.

Creating an issue on the repository is also a good way to ask questions about anything in the course it will prompt additions and expand the FAQ section.

About your instructor

Name: Dr. Sarah Brown Office hours: TBA via zoom, link on GitHub Org Page

Dr. Brown is an Assistant Professor of Computer Science, who does research on how social context changes machine learning. Dr. Brown earned a PhD in Electrical Engineering from Northeastern University, completed a postdoctoral fellowship at University of California Berkeley, and worked as a postdoctoral research associate at Brown University before joining URI. At Brown University, Dr. Brown taught the Data and Society course for the Master's in Data Science Program.

Important

For assignment or notes specific issues, a comment on the corresponding repository is the best. I cannot help you with code issues from screenshots.

The best way to contact me for general questions is e-mail or by dropping into my office hours. Please include `[CSC310]` or `[DSP310]` in the subject line of your email along with the topic of your message. This is important, because your messages are important, but I also get a lot of e-mail. Consider these a cheat code to my inbox: I have setup a filter that will flag your e-mail if you use one of those in the subject to ensure that I see it. I rarely check e-mail between 6pm and 9am, on weekends or holidays. You might see me post or send things during these hours, but I will not reliably see emails that arrive during those hours.

Tools and Resources

We will use a variety of tools to conduct class and to facilitate your programming. You will need a computer with Linux, MacOS, or Windows. It is unlikely that a tablet will be able to do all of the things required in this course. A Chromebook may work, especially with developer tools turned on. Ask Dr. Brown if you need help getting access to an adequate computer.

Skip to main content

- paid for by URI **OR**
- freely available online.

BrightSpace

This will be the central location from which you can access links to other materials. Any links that are for private discussion among those enrolled in the course will be available only from our course Brightspace site.

Prismia chat

Our class link for Prismia chat is available on Brightspace. We will use this for chatting and in-class understanding checks.

On Prismia, all students see the instructor's messages, but only the Instructor and TA see student responses.

Course website

The course manual will have content including the class policies, scheduling, class notes, assignment information, and additional resources. This will be linked from Brightspace and available publicly online at rhodyprog4ds.github.io/BrownSpring23/. Links to the course reference text and code documentation will also be included here in the assignments and class notes.

GitHub

You will need a GitHub Account. If you do not already have one, please create one by the first day of class. If you have one, but have not used it recently, you may need to update your password and login credentials as the Authentication rules changed over the summer. In order to use the command line with https, you will need to us the GitHub CLI or create a Personal Access Token for each device you use. In order to use the command line with SSH, set up your public key.

Programming Environment

This a programming course, so you will need a programming environment. In order to complete assignments you need the items listed in the requirements list. The easiest way to meet these requirements is to follow the recommendations below. I will provide instruction assuming that you have followed the recommendations.

Requirements:

- Python with scientific computing packages (numpy, scipy, jupyter, pandas, seaborn, sklearn)
- Git
- A web browser compatible with Jupyter Notebooks

Skip to main content

Warning

Everything in this class will be tested with the up to date (or otherwise specified) version of Jupyter Notebooks. Google Colab is similar, but not the same, and some things may not work there. It is an okay backup, but should not be your primary work environment.

Recommendation:

- Install python via Anaconda
- if you use Windows, install Git with GitBash (video instructions).
- if you use MacOS, install Git with the Xcode Command Line Tools. On Mavericks (10.9) or above you can do this by trying to run git from the Terminal the very first time. `git --version`
- if you use Chrome OS, follow these instructions:

1. Find Linux (Beta) in your settings and turn that on.
2. Once the download finishes a Linux terminal will open, then enter the commands: sudo apt-get update and sudo apt-get upgrade. These commands will ensure you are up to date.
3. Install tmux with:

```
sudo apt -t stretch-backports install tmux
```

4. Next you will install nodejs, to do this, use the following commands:

```
curl -sL https://deb.nodesource.com/setup_14.x | sudo -E bash
sudo apt-get install -y nodejs
sudo apt-get install -y build-essential.
```

5. Next install Anaconda's Python from the website provided by the instructor and use the top download link under the Linux options.
6. You will then see a .sh file in your downloads, move this into your Linux files.
7. Make sure you are in your home directory (something like home/YOURUSERNAME), do this by using the `pwd` command.
8. Use the `bash` command followed by the file name of the installer you just downloaded to start the installation.
9. Next you will add Anaconda to your Linux PATH, do this by using the `vim .bashrc` command to enter the .bashrc file, then add the `export PATH=/home/YOURUSERNAME/anaconda3/bin/:$PATH` line. This can be placed at the end of the file.
10. Once that is inserted you may close and save the file, to do this hold escape and type `:x`, then press enter. After doing that you will be returned to the terminal where you will then type the source .bashrc command.
11. Next, use the `jupyter notebook –generate-config` command to generate a Jupyter Notebook.
12. Then just type `jupyter lab` and a Jupyter Notebook should open up.

Optional:

- Text Editor: you may want a text editor outside of the Jupyter environment. Jupyter can edit markdown files (that you'll need for your portfolio), in browser, but it is more common to use a text editor like Atom or Sublime for this purpose.

Skip to main content

- Windows
- Mac

On Mac, to install python via environment, this article may be helpful

- I don't have a video for linux, but it's a little more straight forward.

Textbook

The text for this class is a reference book and will not be a source of assignments. It will be a helpful reference and you may be directed there for answers to questions or alternate explanations ot topics.

Python for Data Science is available free online:

Zoom (backup and office hours only)

This is where we will meet if for any reason we cannot be in person. You will find the link to class zoom sessions on Brightspace.

URI provides all faculty, staff, and students with a paid Zoom account. It *can* run in your browser or on a mobile device, but you will be able to participate in class best if you download the Zoom client on your computer. Please log in and configure your account. Please add a photo of yourself to your account so that we can still see your likeness in some form when your camera is off. You may also wish to use a virtual background and you are welcome to do so.

Class will be interactive, so if you cannot be in a quiet place at class time, headphones with a built in microphone are strongly recommended.

For help, you can access the instructions provided by IT.

---

[1] Too long; didn't read.

Data Science Achievements

In this course there are 5 learning outcomes that I expect you to achieve by the end of the semester. To get there, you'll focus on 15 smaller achievements that will be the basis of your grade. This section will describe how the topics covered, the learning outcomes, and the achievements are covered over time. In the next section, you'll see how these achievements turn into grades.

Learning Outcomes

By the end of the semester

1. (process) Describe the process of data science, define each phase, and identify standard tools
2. (data) Access and combine data in multiple formats for analysis
3. (exploratory) Perform exploratory data analyses including descriptive statistics and visualization
4. (modeling) Select models for data by applying and evaluating mutiple models to a single dataset
5. (communicate) Communicate solutions to problems with data in common industry formats

We will build your skill in the `process` and `communicate` outcomes over the whole semester. The middle three skills will correspond roughly to the content taught for each of the first three portfolio checks.

Schedule

The course will meet in . Every class will include participatory live coding (instructor types code while explaining, students follow along) instruction and small exercises for you to progress toward level 1 achievements of the new skills introduced in class that day.

Each Assignment will have a deadline posted on the assignment page, typically the same day each week. Portfolio deadlines will be announced at least 2 weeks in advance.

week	topics	skills
1	[admin, python review]	process
2	Loading data, Python review	[access, prepare, summarize]
3	Exploratory Data Analysis	[summarize, visualize]
4	Data Cleaning	[prepare, summarize, visualize]
5	Databases, Merging DataFrames	[access, construct, summarize]
6	Modeling, classification performance metrics, cross validation	[evaluate]
7	Naive Bayes, decision trees	[classification, evaluate]
8	Regression	[regression, evaluate]
9	Clustering	[clustering, evaluate]
10	SVM, parameter tuning	[optimize, tools]
11	KNN, Model comparison	[compare, tools]
12	Text Analysis	[unstructured]
13	Images Analysis	[unstructured, tools]
14	Deep Learning	[tools, compare]

Achievement Definitions

The table below describes how your work will be assessed to earn each achievement. The keyword for each skill is a short name that will be used to refer to skills throughout the course materials; the full description of the skill is in this table.

Skip to main content

keyword	skill	Level 1	Level 2	Level 3
python	pythonic code writing	python code that mostly runs, occasional pep8 adherance	python code that reliably runs, frequent pep8 adherance	reliable, efficient, pythonic code that consistently adheres to pep8
process	describe data science as a process	Identify basic components of data science	Describe and define each stage of the data science process	Compare different ways that data science can facilitate decision making
access	access data in multiple formats	load data from at least one format; identify the most common data formats	Load data for processing from the most common formats; Compare and constrast most common formats	access data from both common and uncommon formats and identify best practices for formats in different contexts
construct	construct datasets from multiple sources	identify what should happen to merge datasets or when they can be merged	apply basic merges	merge data that is not automatically aligned
summarize	Summarize and describe data	Describe the shape and structure of a dataset in basic terms	compute summary statndard statistics of a whole dataset and grouped data	Compute and interpret various summary statistics of subsets of data
visualize	Visualize data	identify plot types, generate basic plots from pandas	generate multiple plot types with complete labeling with pandas and seaborn	generate complex plots with pandas and plotting libraries and customize with matplotlib or additional parameters
prepare	prepare data for analysis	identify if data is or is not ready for analysis, potential problems with data	apply data reshaping, cleaning, and filtering as directed	apply data reshaping, cleaning, and filtering manipulations reliably and correctly by assessing data as received
evaluate	Evaluate model performance	Explain and compute basic performance metrics for different data science tasks	Apply and interpret basic model evaluation metrics to a held out test set	Evaluate a model with multiple metrics and cross validation
classification	Apply classification	identify and describe what classification is, apply pre-fit classification models	fit, apply, and interpret preselected classification model to a dataset	fit and apply classification models and select appropriate classification models for different contexts
regression	Apply Regression	identify what data that can be used for regression looks like	fit and interpret linear regression models	fit and explain regrularized or nonlinear regression
clustering	Clustering	describe what clustering is	apply basic clustering	apply multiple clustering techniques, and interpret results
optimize	Optimize model parameters	Identify when model parameters need to be	Optimize basic model parameters such as	Select optimal parameters based of mutiple quanttiateve criteria and automate parameter tuning

keyword	skill	Level 1	Level 2	Level 3
compare	compare models	Qualitatively compare model classes	Compare model classes in specific terms and fit models in terms of traditional model performance metrics	Evaluate tradeoffs between different model comparison types
representation	Choose representations and transform data	Identify options for representing text and categorical data in many contexts	Apply at least one representation to transform unstructured or inappropriately data for model fitting or summarizing	apply transformations in different contexts OR compare and contrast multiple representations a single type of data in terms of model performance
workflow	use industry standard data science tools and workflows to solve data science problems	Solve well strucutred fully specified problems with a single tool pipeline	Solve well-strucutred, open-ended problems, apply common structure to learn new features of standard tools	Independently scope and solve realistic data science problems OR independently learn releated tools and describe strengths and weakensses of common tools

Assignments and Skills

Using the keywords from the table above, this table shows which assignments you will be able to demonstrate which skills and the total number of assignments that assess each skill. This is the number of opportunities you have to earn Level 2 and still preserve 2 chances to earn Level 3 for each skill.

keyword	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	# Assignments
python	1	1	0	1	1	0	0	0	0	0	0	0	0	4
process	1	0	0	0	0	1	1	1	1	1	1	0	0	7
access	0	1	1	1	1	0	0	0	0	0	0	0	0	4
construct	0	0	0	0	1	0	1	1	0	0	0	0	0	3
summarize	0	0	1	1	1	1	1	1	1	1	1	1	1	11
visualize	0	0	1	1	0	1	1	1	1	1	1	1	1	10
prepare	0	0	0	1	1	0	0	0	0	0	0	0	0	2
evaluate	0	0	0	0	0	1	1	1	0	1	1	0	0	5
classification	0	0	0	0	0	0	1	0	0	1	0	0	0	2
regression	0	0	0	0	0	0	0	1	0	0	1	0	0	2
clustering	0	0	0	0	0	0	0	0	1	0	1	0	0	2
optimize	0	0	0	0	0	0	0	0	0	1	1	0	0	2
compare	0	0	0	0	0	0	0	0	0	0	1	0	1	2
representation	0	0	0	0	0	0	0	0	0	0	0	1	1	2
workflow	0	0	0	0	0	0	0	0	0	1	1	1	1	4

Skip to main content

> ⚠ **Warning**
>
> **process** achievements are accumulated a little slower; details will follow.

Extensions

> ⚠ **Warning**
>
> this rolling deadline is new for Fall 2024 and aims to let students distribute work in a better way for yourself. After A2 feedback is posted, I will give more explanation about how to do this, in concrete terms.

There are no extensions applicable to assignment 1, but starting after assignment 2's feedback you can start working on level 3 achievements. You can add on and extend each analysis, once you have earned level 2 for a skill to earn level 3. You can also add new analyses that instead combine different sets of skills.

Extensions will all be graded by Dr. Brown (and most assignments will be graded by the TA Surbhi). You will make separate PRs for your attempts at level 3 from level 2.

While assignments have fixed grades, you can submit extensions as you complete them. I recommend planning to work on them consistently throughout the semester.

> ⚠ **Warning**
>
> In previous semesters, there were checklists, but they are removed because they distracted students from learning the important things

Grading

This section of the syllabus describes the principles and mechanics of the grading for the course. This course will be graded on a basis of a set of *skills* (described in detail the next section of the syllabus). This is in contrast to more common grading on a basis of points earned through assignments.

Principles of Grading

Learning happens through practice and feedback. My goal as a teacher is for you to learn. The grading in this course is based on your learning of the material, rather than your completion of the activities that are assigned.

This course is designed to encourage you to work steadily at learning the material and demonstrating your new knowledge. There are no single points of failure, where you lose points that cannot be recovered. Also, you cannot cram anything one time and then forget it. The material will build and you have to demonstrate that you retained things.

- Earning a C in this class means you have a general understanding of Data Science and could participate in a basic conversation about all of the topics we cover. I expect everyone to reach this level.
- Earning a B means that you could solve simple data science problems on your own and complete parts of more complex

ccessible goal, it

does not require you to get anything on the first try or to explore topics on your own. I expect most students to reach this level.
- Earning an A means that you could solve moderately complex problems independently and discus the quality of others' data science solutions. This class will be challenging, it requires you to explore topics a little deeper than we cover them in class, but unlike typical grading it does not require all of your assignments to be near perfect.

Grading this way also is more amenable to the fact that there are correct and incorrect ways to do things, but there is not always a single correct answer to a realistic data science problem. Your work will be assessed on whether or not it demonstrates your learning of the targeted skills. You will also receive feedback on how to improve.

How it works

> **Warning**
>
> This is going to change; you will get a notificaiton when it is final

There are 15 skills that you will be graded on in this course. While learning these skills, you will work through a progression of learning, first getting the basic idea, then applying it, then exploring advanced usage. You will have to demonstrate each skill area on 3 separate occasions to get level 3 credit for it. Your grade will be based on earning 45 achievements that are organized into 15 skill groups with 3 levels for each.

These map onto letter grades roughly as follows:

- If you achieve level 1 in all of the skills, you will earn at least a C in the course.
- To earn a B, you must earn all of the level 1 and level 2 achievements.
- To earn an A, you must earn all of the achievements.

You will have at least three opportunities to earn every level 2 achievement. You will have at least two opportunities to earn every level 3 achievement.

This is a
student w
up below
level 3s a
any work

Each level of achievement corresponds to a phase in your learning of the skill:

- To earn level 1 achievements, you will need to demonstrate basic awareness of the required concepts and know approximately what to do, but you may need specific instructions of which things to do or to look up examples to modify every step of the way. You can learn level 1 in any assignment (even without completing it completely correct) or in office hours.
- To earn level 2 achievements you will need to demonstrate understanding of the concepts and the ability to apply them with instruction after earning the level 1 achievement for that skill. Weekly assignments will guide you to level 2, if you complete it well.
- To earn level 3 achievements you will be required to consistently execute each skill and demonstrate deep understanding of the course material, after achieving level 2 in that skill. This will happen mostly extending previous assignments in your portfolio.

For each skill these are defined in the Achievement Definition Table

Skip to main content

Participation

While attending synchronous class sessions, there will be understanding checks and in class exercises. Completing in class exercises and correctly answering questions in class is approximately the level of understanding required for level 1.

We will not directly add these to your grade, but you can always visit office hours to earn level 1, so that your assignment can go straight to 2 in most skills.

Assignments

For your learning to progress and earn level 2 achievements, you must practice with the skills outside of class time. You wil submit all of your assignments in your portfolio repository. Sometimes you will need to sync your repo to get templates for the assignment and some will be open. All will be posted on this site.

Assignments will each evaluate certain skills. After your assignment is reviewed, you will get qualitative feedback on your work, and an assessment of your demonstration of the targeted skills. Your feedback will recap all of your earned achievements to that point, not only what was earned in that assignment.

You *can* revise assignments if you do not earn achievements by also adding reflections to them while you edit, but since each skill is available in multiple assignments you do not have to.

You can revise what you submitted and resubmit it, with reflections and explanation of what you were confused about, what you tried initially, how you eventually figured it out, and explains the correct answer.

Extensions

> **Warning**
>
> the logistics of this are changing, but tbd, will update soon

To earn level 3 achievements, you will extend your prior work. Starting with assignment 2, there will be extension ideas in the assignment.

TLDR

You *could* earn a C through oral exams in office hours alone. To earn a B, you must complete assignments. To earn an A you must complete assignments and add extensions that demonstrate deeper understanding (tips will be provided).

Detailed mechanics

The table below shows the minimum number of skills at each level to earn each letter grade.

Skip to main content

letter grade	Level 3	Level 2	Level 1
A	15	15	15
A-	10	15	15
B+	5	15	15
B	0	15	15
B-	0	10	15
C+	0	5	15
C	0	0	15
C-	0	0	10
D+	0	0	5
D	0	0	3

For example, if you achieve level 2 on all of the skills and level 3 on 7 skills, that will be a B+.

If you achieve level 3 on 14 of the skills, but only level 1 on one of the skills, that will be a B-, because the minimum number of level 2 achievements for a B is 15. In this scenario the total number of achievements is 14 at level 3, 14 at level 2 and 15 at level 3, because you have to earn achievements within a skill in sequence.

The letter grade can be computed as follows

Important

this will be revealed after assignment 1

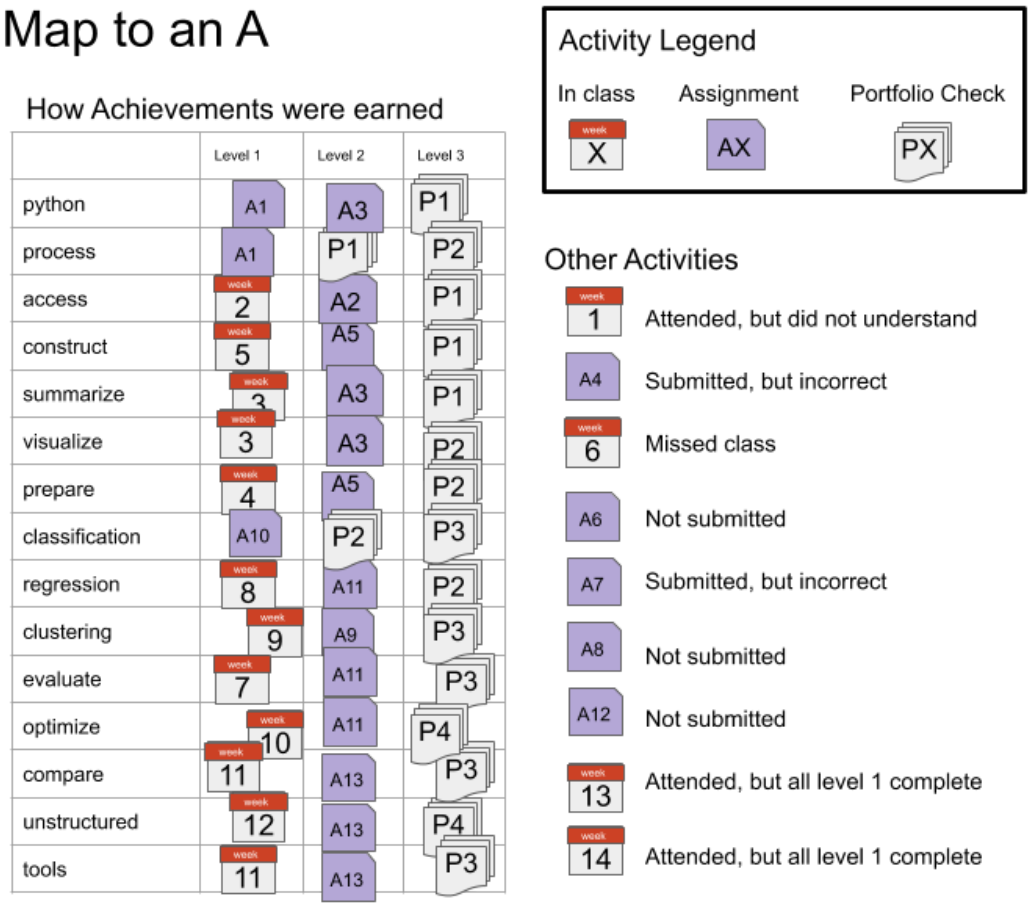
Grading Examples

Getting an A Without Perfection

Warning

acheivements are no longer awarded in class in fall 24; images to be updated and porfolio is rolling instead of discrete check points.

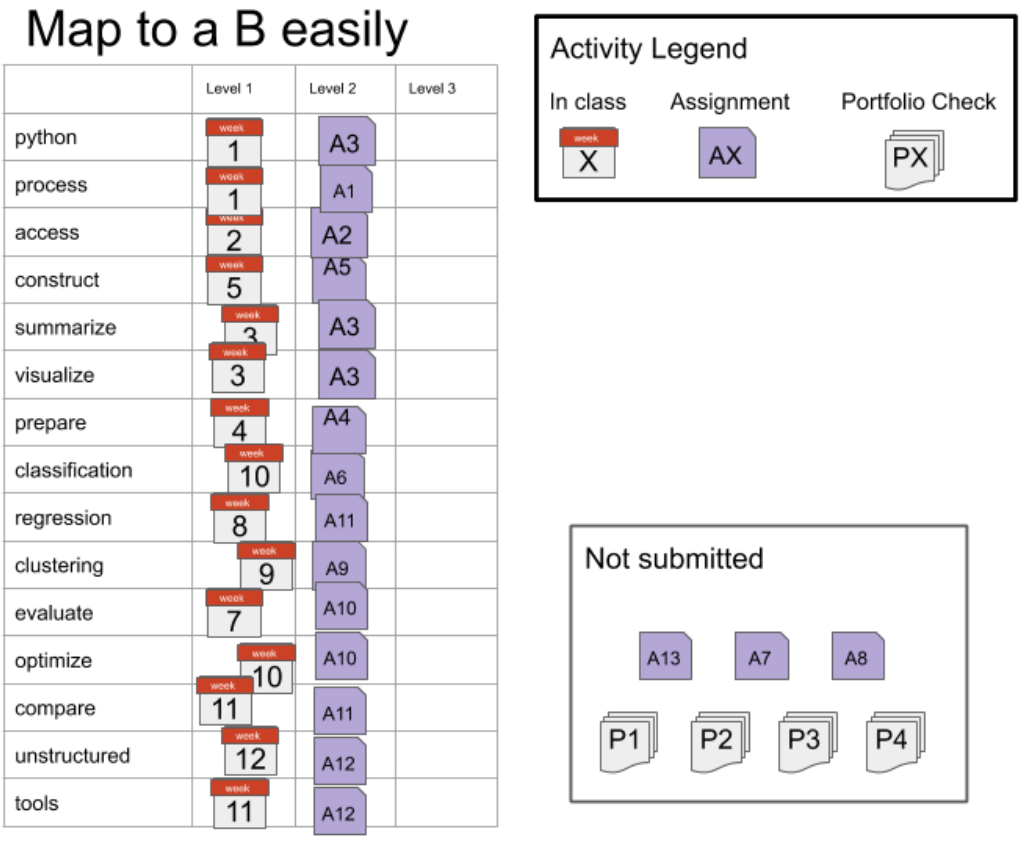
Skip to main content



In this example the student made several mistakes, but still earned an A. This is the advantage to this grading scheme. For the `python`, `process`, and `classification` skills, the level 1 achievements were earned on assignments, not in class. For the `process` and `classification` skills, the level 2 achievements were not earned on assignments, only on portfolio checks, but they were earned on the first portfolio of those skills, so the level 3 achievements were earned on the second portfolio check for that skill. This student's fourth portfolio only demonstrated two skills: `optimize` and `unstructured`. It included only 1 analysis, a text analysis with optimizing the parameters of the model. Assignments 4 and 7 were both submitted, but didn't earn any achievements, the student got feedback though, that they were able to apply in later assignments to earn the achievements. The student missed class week 6 and chose to not submit assignment 6 and use week 7 to catch up. The student had too much work in another class and chose to skip assignment 8. The student tried assignment 12, but didn't finish it on time, so it was not graded, but the student visited office hours to understand and be sure to earn the level 2 `unstructured` achievement on assignment 13.

Getting a B with minimal work

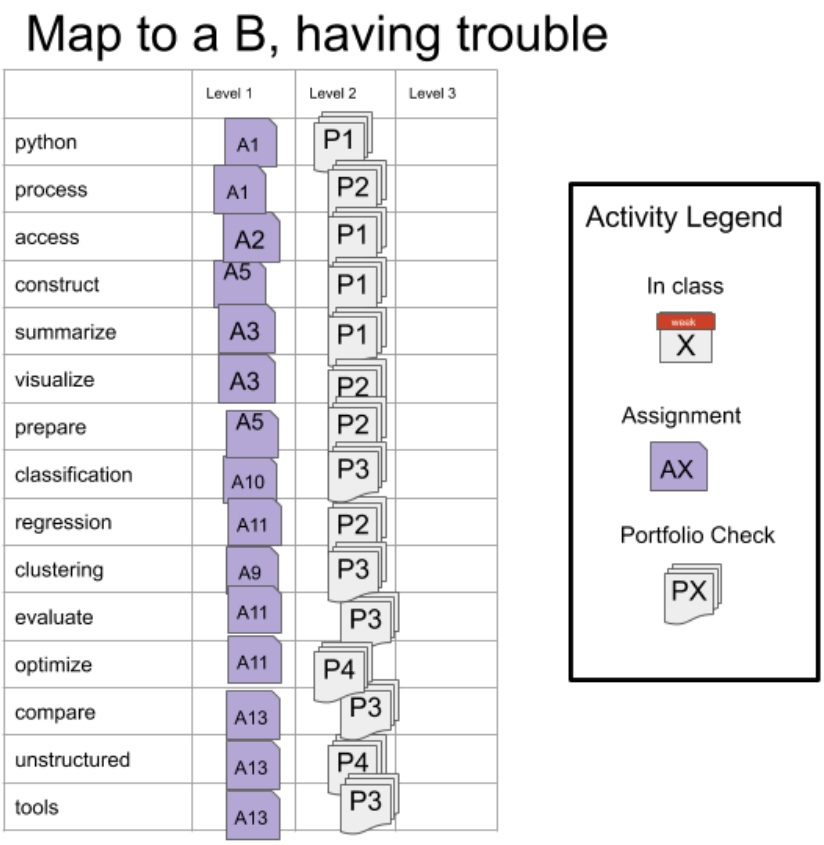
Skip to main content



In this example, the student earned all level 1 achievements in class and all level 2 on assignments. This student was content with getting a B and chose to not submit a portfolio.

Getting a B while having trouble

Skip to main content



In this example, the student struggled to understand in class and on assignments. Assignments were submitted that showed some understanding, but all had some serious mistakes, so only level 1 achievements were earned from assignments. The student wanted to get a B and worked hard to get the level 2 achievements on the portfolio checks.

Grading Policies

Attendance

Attendance and active participation is expeted. You earn level 1 achievements in class and all class sessions are active learning.

If you miss class, you can make it up by reading the posted notes and the prismia transcript. Best practice is to download them as a notebook and run them to make sure you understand each step. If you miss both class sessions in a week, the level one achievements can be made up through annotation or in your assignment.

Absences do not require notification.

Assignment Deadlines and Late Work

Late assignments will not be graded. Extensions will not be granted for assignments. Every skill will be assessed through more than one assignment, so missing assignments occasionally will not necessarily impact your grade. If you do not submit **any** assignments that cover a given skill, you may earn the level 2 achievement in that skill through a portfolio check, but you

Skip to main content

If you submit work that is **not complete**, it will be assessed and receive feedback. Submitting pseudocode or code with errors and comments about what you have tried could even be enought earn a level 1 achievement. Assignments cover multiple skills, so partially completing the assignment may earn level 2 for one, but not all. Submitting *something* even if it is not perfect is important to keeping conversation open and getting feedback and help continuously.

> **Important**
>
> If you have a serious issue during the semester, that prevents you from submitting an assignment, email Dr. Brown to make a plan. Extensions will still not be granted because they do not help you in the long run, instead an alternate plan of how to earn the target grade.

Portfolio Deadlines and Extensions

Building your Data Science Portfolio should be an ongoing process, where you commit work to your portfolio frequently. If something comes up and you cannot finish all that you would like assessed by the deadline, open an `Extension Request` issue on your repository at least **24 hours** before the deadline.

In this issue, include:

1. A proposed new deadline
2. What additional work you plan to add
3. Why the extension is important to your learning
4. Why the extension will not hinder your ability to complete the next assignments and portfolio check on time.
5. (if less than 24 hours before the deadline) why you need an emergency request

> **Important**
>
> Your request should not include a reason why you are asking, unless you are asking for an emergency extension. Emergency requests can be submitted at any time, even after the deadline.

This request should be no more than 7 sentences.

Portfolio due dates will be announced well in advance and prompts for it will be released weekly. You should spend some time working on it each week, applying what you've learned so far, from the feedback on previous assignments.

Academic Dishonesty

All work must represent your own understanding of both the data science practices and the related programming concepts. Submitting code or prose that was generated by a generative model or another person is not allowed.

If you are found to have submitted work that does not constitute your own work, the following penalties apply:

- in a portfolio, all achievements attempted in the dishonest component are permanently ineligible.
- in an assignment the level three achievements for the skills of focus in the assignment are ineligible, and the relevent level

Skip to main content

For example, if you are violate the academic honesty policy in asignment 4, Prepare level 3 becomes ineligible and you must meet the requirements for prepare level 3 in a portfolio in order to earn prepare level 2.

If you violate acadmic honesty policy in portfolio 1 while attempting level 3 at Python, access, prepare, summarize and visualize and process level 2, then your maximum grade becomes a B+, because level 3 in all five of those skills becomes inelgible.

Regrading

1. Add comments:
    - For general questions, post on the conversation tab of your Feedback PR with your request.
    - For specific questions, reply to a specifc comment.
2. Re-request a review from Dr. Brown on your Feedback Pull request.

If you think we missed *where* you did something, add a comment on that line to help us find it (on the code tab of the PR, click the plus (+) next to the line) and then post on the conversation tab with an overview of what you're requesting and tag @brownsarahm

Course Style Guide

Following a style guide is a common requirement in companies to make it so that code written by different people stays easy to read for everyone. Consistent style also makes it easier to onboard new developers join a project and contribute faster.

The following style guide serves as practice for you following a style guide, makes your work easier to read for grading purposes, and holds you accountable to learning deeply and demonstrating that you have learned well.

Hard Requirements

> ⚠ **Warning**
>
> All work must adhere to these requirements or it may receive no feedback or credit. Minor misses may receive warnings, but if submitted work does not appear to represent a good faith effort at adhering to this style guide, the only comment will be, "Follow the style guide on the next assignment"

1. All code must be submitted in a notebook file (.ipynb or myst)
2. Code must run or have explicit questions and comments about what was done about the errors
3. Python comments (`# comment text`) inside code cells should **only** be used to explain complex code that is not explained in the course notes. Using such code requires a citation for the source.
4. Each code cell should exactly one conceptually complete step in terms of the analysis.
5. Nearly all code cells should have output in some form
6. Every code cell must be motivated by text in markdown before it
7. Every code cell's output must be interpretted in a markdown cell (may be combined in one cell that explains the above

Skip to main content

8. the `print` function can only be used when it improves the readability over using jupyter's display, must be justified
9. No deprecated or dangerous code constructs without justification
10. All assignment questions must be answered in markdown cells
11. Notebook files may not have extraneous metadata in them
12. Alternative libraries not taught in class can only be used when attempting level 3.

Additional Style

> **Important**
>
> Mistakes on these will get detailed feedback once and a "see previous feedback" a second time before the whole assignment receives no feedback.

1. Code should ahdere to PEP8
2. Markdown syntax should be used to enhance the readability of the text (eg not all headings, bullets where they make sense)
3. Best practices that are highlighted in class should be followed (this list will expand over the semester)

Support

> **Warning**
>
> URI changed some links and this page is not yet up to date

Academic Enhancement Center

Academic Enhancement Center (for undergraduate courses): Located in Roosevelt Hall, the AEC offers free face-to-face and web-based services to undergraduate students seeking academic support. Peer tutoring is available for STEM-related courses by appointment online and in-person. The Writing Center offers peer tutoring focused on supporting undergraduate writers at any stage of a writing assignment. The UCS160 course and academic skills consultations offer students strategies and activities aimed at improving their studying and test-taking skills. Complete details about each of these programs, up-to-date schedules, contact information and self-service study resources are all available on the AEC website.

- **STEM Tutoring** helps students navigate 100 and 200 level math, chemistry, physics, biology, and other select STEM courses. The STEM Tutoring program offers free online and limited in-person peer-tutoring this fall. Undergraduates in introductory STEM courses have a variety of small group times to choose from and can select occasional or weekly appointments. The TutorTrac application is available through URI Microsoft 365 single sign-on and by visiting aec.uri.edu. More detailed information and instructions can be found on the AEC tutoring page.
- **Academic Skills Development** resources helps students plan work, manage time, and study more effectively. In Fall 2020, all Academic Skills and Strategies programming are offered both online and in-person. UCS160: Success in Higher Education is a one-credit course on developing a more effective approach to studying. Academic Consultations are 30-

issues. Study Your Way to Success is a self-guided web portal connecting students to tips and strategies on studying and time management related topics. For more information on these programs, visit the Academic Skills Page or contact Dr. Hayes directly at davidhayes@uri.edu.

- The **Undergraduate Writing Center** provides free writing support to students in any class, at any stage of the writing process: from understanding an assignment and brainstorming ideas, to developing, organizing, and revising a draft. Fall 2020 services are offered through two online options: 1) real-time synchronous appointments with a peer consultant (25- and 50-minute slots, available Sunday - Friday), and 2) written asynchronous consultations with a 24-hour turn-around response time (available Monday - Friday). Synchronous appointments are video-based, with audio, chat, document-sharing, and live captioning capabilities, to meet a range of accessibility needs. View the synchronous and asynchronous schedules and book online, visit uri.mywconline.com.

General URI Policies

> **Warning**
>
> URI changed some links and this page is not yet up to date

Anti-Bias Statement:

We respect the rights and dignity of each individual and group. We reject prejudice and intolerance, and we work to understand differences. We believe that equity and inclusion are critical components for campus community members to thrive. If you are a target or a witness of a bias incident, you are encouraged to submit a report to the URI Bias Response Team at www.uri.edu/brt. There you will also find people and resources to help.

Mental Health and Wellness

We understand that college comes with challenges and stress associated with your courses, job/family responsibilities and personal life. URI offers students a range of services to support your mental health and wellbeing, including the URI Counseling Center, MySSP (Student Support Program) App, the Wellness Resource Center, and Well-being Coaching.

Disability Services for Students Statement:

Your access in this course is important. Please send me your Disability Services for Students (DSS) accommodation letter early in the semester so that we have adequate time to discuss and arrange your approved academic accommodations. If you have not yet established services through DSS, please contact them to engage in a confidential conversation about the process for requesting reasonable accommodations in the classroom. DSS can be reached by calling: 401-874-2098, visiting: web.uri.edu/disability, or emailing: dss@etal.uri.edu. We are available to meet with students enrolled in Kingston as well as Providence courses.

Academic Honesty

Students are expected to be honest in all academic work. A student's name on any written work, quiz or exam shall be

Skip to main content

k should be stated

in the student's own words, properly attributed to its source. Students have an obligation to know how to quote, paraphrase, summarize, cite and reference the work of others with integrity. The following are examples of academic dishonesty.

- Using material, directly or paraphrasing, from published sources (print or electronic) without appropriate citation
- Claiming disproportionate credit for work not done independently
- Unauthorized possession or access to exams
- Unauthorized communication during exams
- Unauthorized use of another's work or preparing work for another student
- Taking an exam for another student
- Altering or attempting to alter grades
- The use of notes or electronic devices to gain an unauthorized advantage during exams
- Fabricating or falsifying facts, data or references directly or indirectly through the use of generative AI
- Facilitating or aiding another's academic dishonesty
- Submitting the same paper for more than one course without prior approval from the instructors

Communications & Office Hours

> **Warning**
>
> Due to Dept Seminar Office hours on 11/10 will be at 5pm instead of 4pm.

Announcements

Announcements will be made via GitHub Release. You can view them online in the releases page or you can get notifications by watching the repository, choosing "Releases" under custom see GitHub docs for instructions with screenshots. You can choose GitHub only or e-mail notificaiton from the notification settings page

Help Hours

Day	Time	Location	Host
Friday	4-6pm	Zoom	Dr. Brown
TBA	TBA	134 Tyler	Dr. Brown
Tuesday	11am-1pm	Zoom	Surbhi
Wednesday	1-3pm	Zoom	Surbhi

Zoom links are on the course organization page of GitHub

To reach out, By usage

We have several different ways to communicate in this course. This section summarizes them

Skip to main content

usage	platform	area	note
in class	prismia	chat	outside of class time this is not monitored closely
any time	prismia	download transcript	use after class to get preliminary notes eg if you miss a class
private questions to your assignment	github	issue on assignment repo	eg bugs in your code"
for general questions that can help others	github	issue on course website	eg what the instructions of an assignment mean or questions about the syllabus
to share resources or ask general questions in a semi-private forum	github	discussion on community repo	include links in your portfolio
matters that don't fit into another category	e-mail	to brownsarahm@uri.edu	remember to include `[CSC310]` or `[DSP310]` (note `verbatim` no space)

Note

e-mail is last because it's not collaborative; other platforms allow us (Proessor + TA) to collaborate on who responds to things more easily.

Tips

For assignment help

- **send in advance, leave time for a response** I check e-mail/github a small number of times per day, during work hours, almost exclusively. You might see me post to this site, post to BrightSpace, or comment on your assignments outside of my normal working hours, but I will not reliably see emails that arrive during those hours. This means that it is important to start assignments early.

Using issues

- use issues for content directly related to assignments. If you push your code to the repository and then open an issue, I can see your code and your question at the same time and download it to run it if I need to debug it
- use issues for questions about this syllabus or class notes. At the top right there's a GitHub logo that allows you to open a issue (for a question) or suggest an edit (eg if you think there's a typo or you find an additional helpful resource related to something)

For E-email

- use e-mail for general inquiries or notifications
- Please include `[CSC310]` or `[DSP310]` in the subject line of your email along with the topic of your message. This is important, because your messages are important, but I also get a lot of e-mail. Consider these a cheat code to my inbox: I have setup a filter that will flag your e-mail if you use one of those in the subject to ensure that I see it.

Skip to main content

1. Welcome & What is Data Science

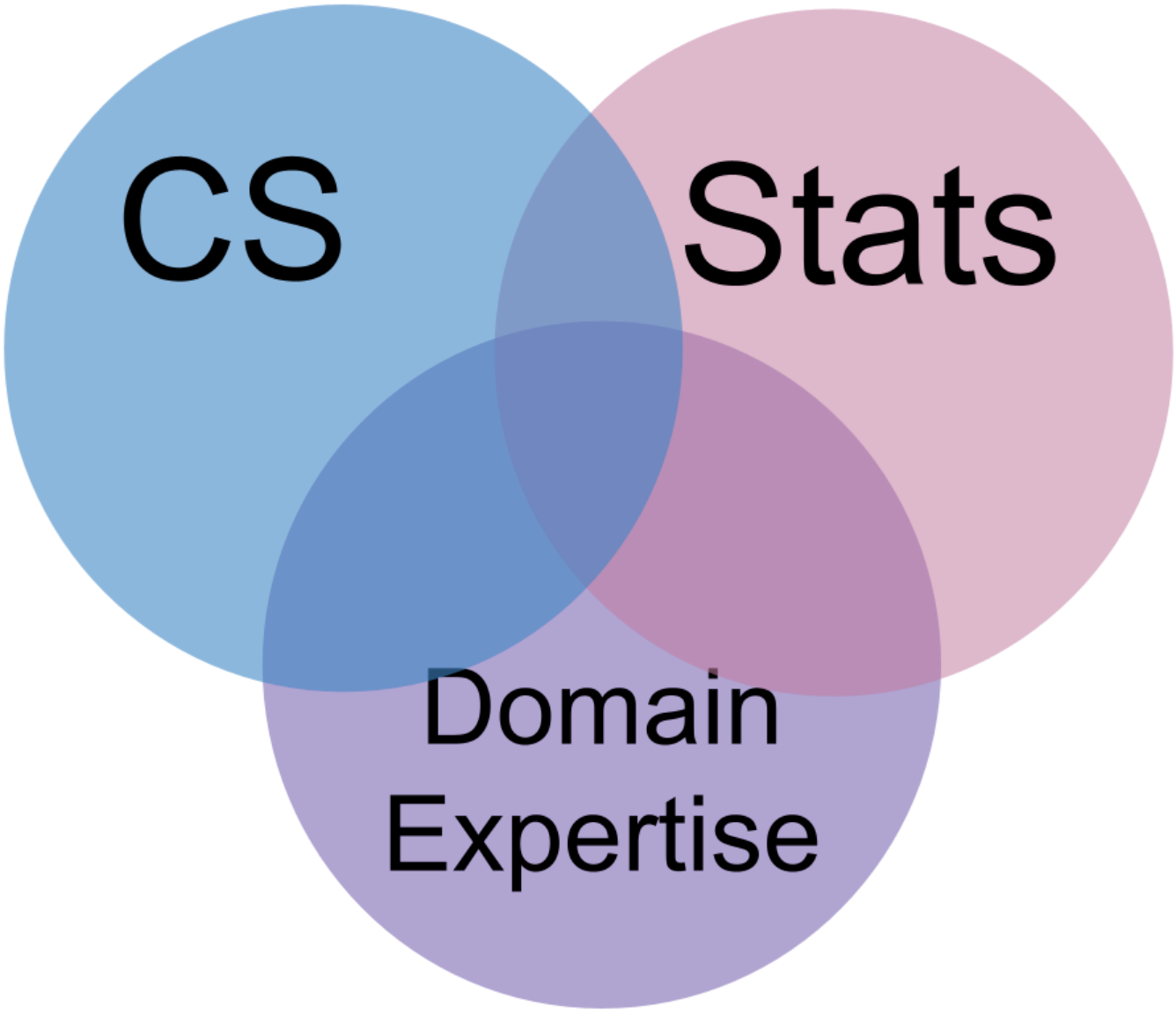
1.1. Prismia Chat

We will use these to monitor your participation in class and to gather information. Features:

- instructor only
- reply to you directly
- share responses for all

1.2. What is Data Science?

Data Science is the combination of



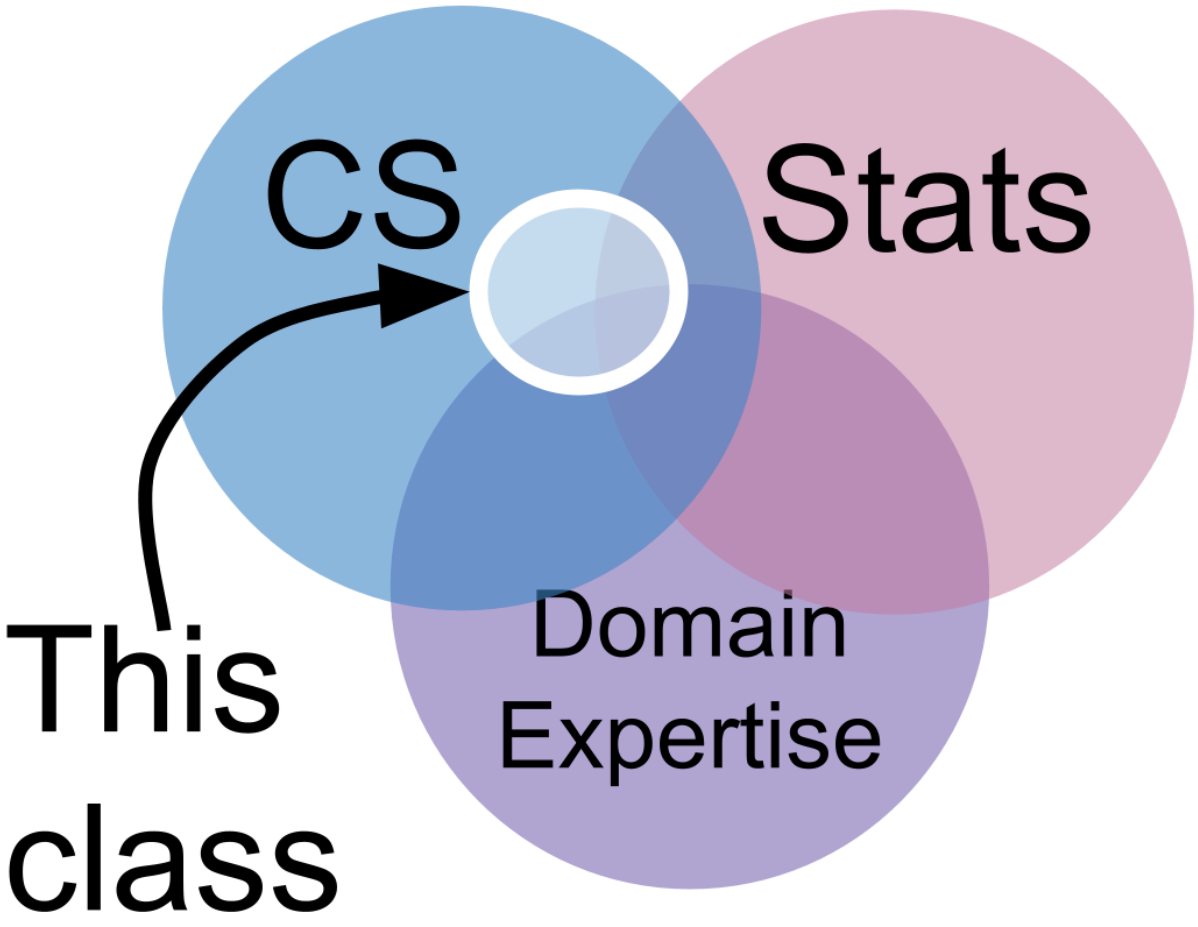
Skip to main content

statistics is the type of math we use to make sense of data. Formally, a statistic is just a function of data.

computer science is so that we can manipulate visualize and automate the inferences we make.

domain expertise helps us have the intuition to know if what we did worked right. A statistic must be interpreted in context; the relevant context determines what they mean and which are valid. The context will say whether automating something is safe or not, it can help us tell whether our code actually worked right or not.

1.2.1. In this class,

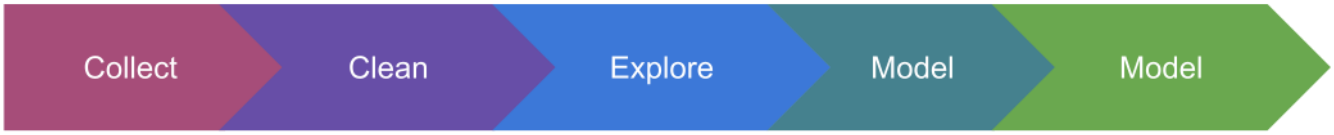


We'll focus on the programming as our main means of studying data science, but we will use bits of the other parts. In particular, you're encouraged to choose datasets that you have domain expertise about, or that you want to learn about.

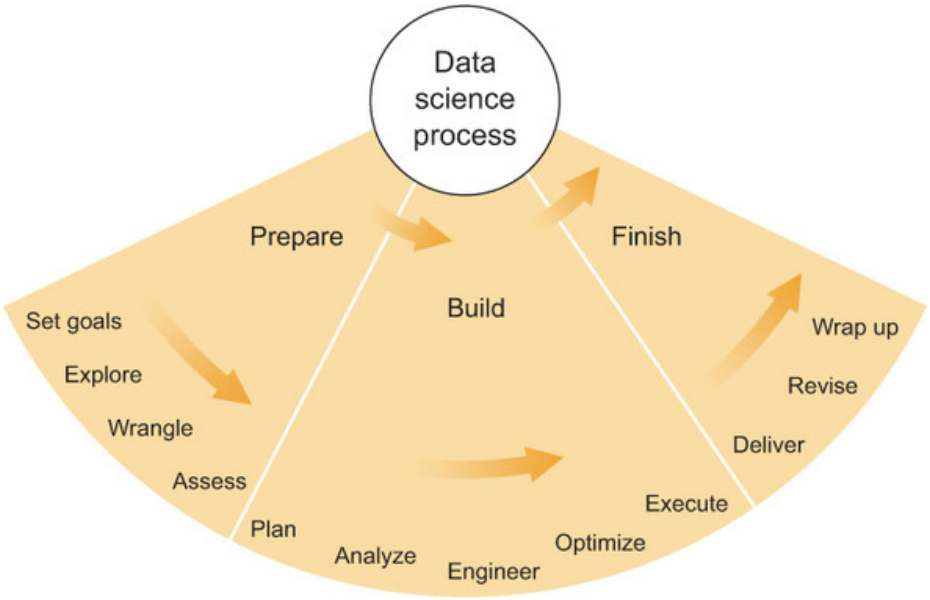
But there are many definitions. We'll use this one, but you may come across others.

1.2.2. How does data science happen?

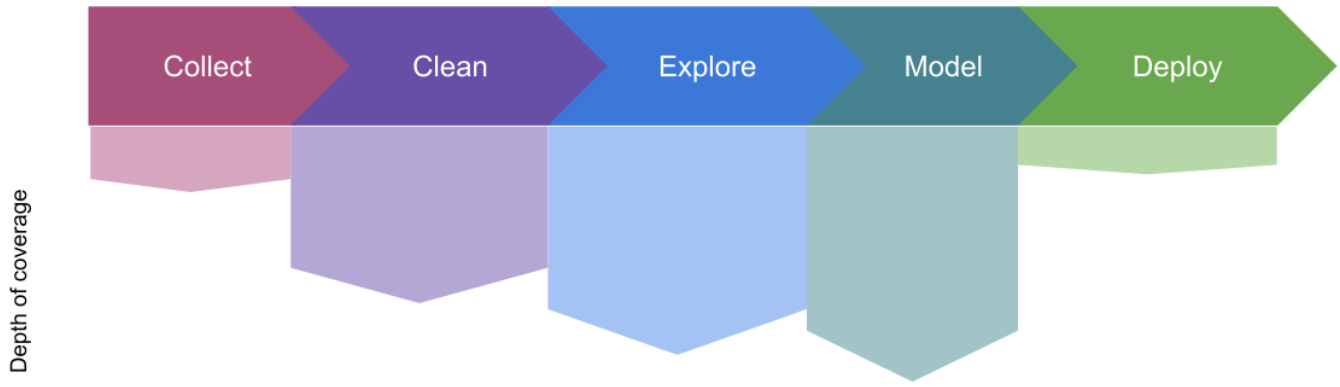
The most common way to think about what doing data science means is to think of this pipeline. It is in the perspective of the data, these are all of the things that happen to the data.



Another way to think about it



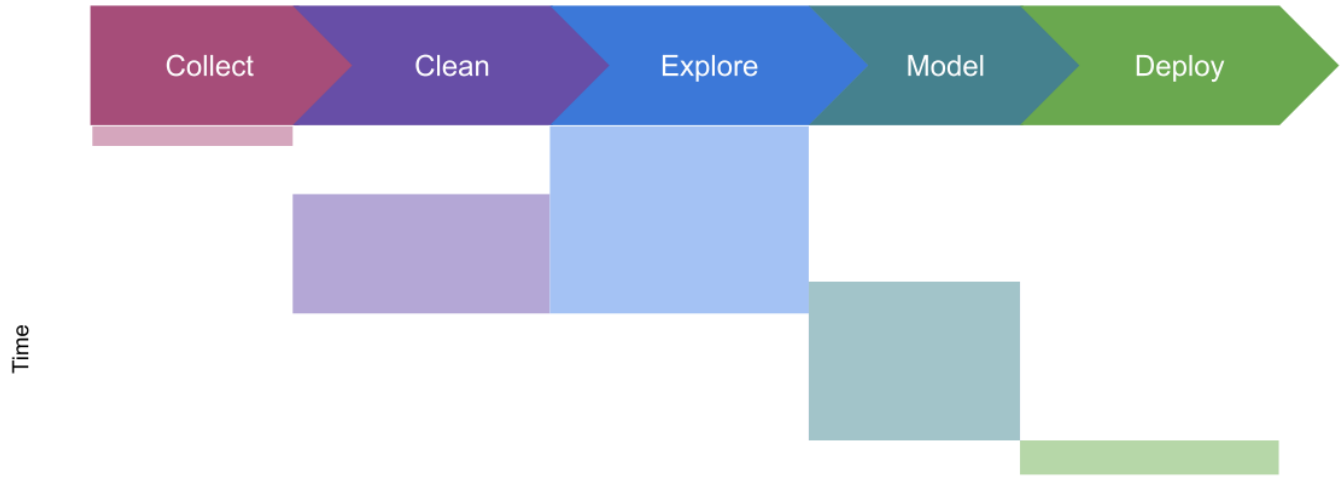
1.2.3. how we'll cover Data Science, in depth



- *collect*: Discuss only a little; Minimal programming involved
- *clean*: Cover the main programming techniques; Some requires domain knowledge beyond scope of course
- *explore*: Cover the main programming techniques; Some requires domain knowledge beyond scope of course
- *model*:Cover the main programming, basic idea of models; How to use models, not how learning algorithms work

Skip to main content

1.2.4. how we'll cover it in, time



We'll cover exploratory data analysis before cleaning because those tools will help us check how we've cleaned the data.

1.3. How this class will work

Participatory Live Coding

What is a topic you want to use data to learn about?

> **Note**
>
> Here I changed this title from the title of the notebook, to a subsection but the rest of this is what was actually done in class, with explanation.

1.4. Intro to Jupyter Notebooks

1.5. Programming for Data Science vs other Programming

The audience is different, so the form is different.

In Data Science our product is more often a report than a program.

Sometimes there will be points in the notes that were not made in class due to time or in response questions that came at the end of class.

Also, in data science we are *using code* to interact with data, instead of having a plan in advance

So programming for data science is more like *writing* it has a narrative flow and is made to be seen more than some other

Skip to main content

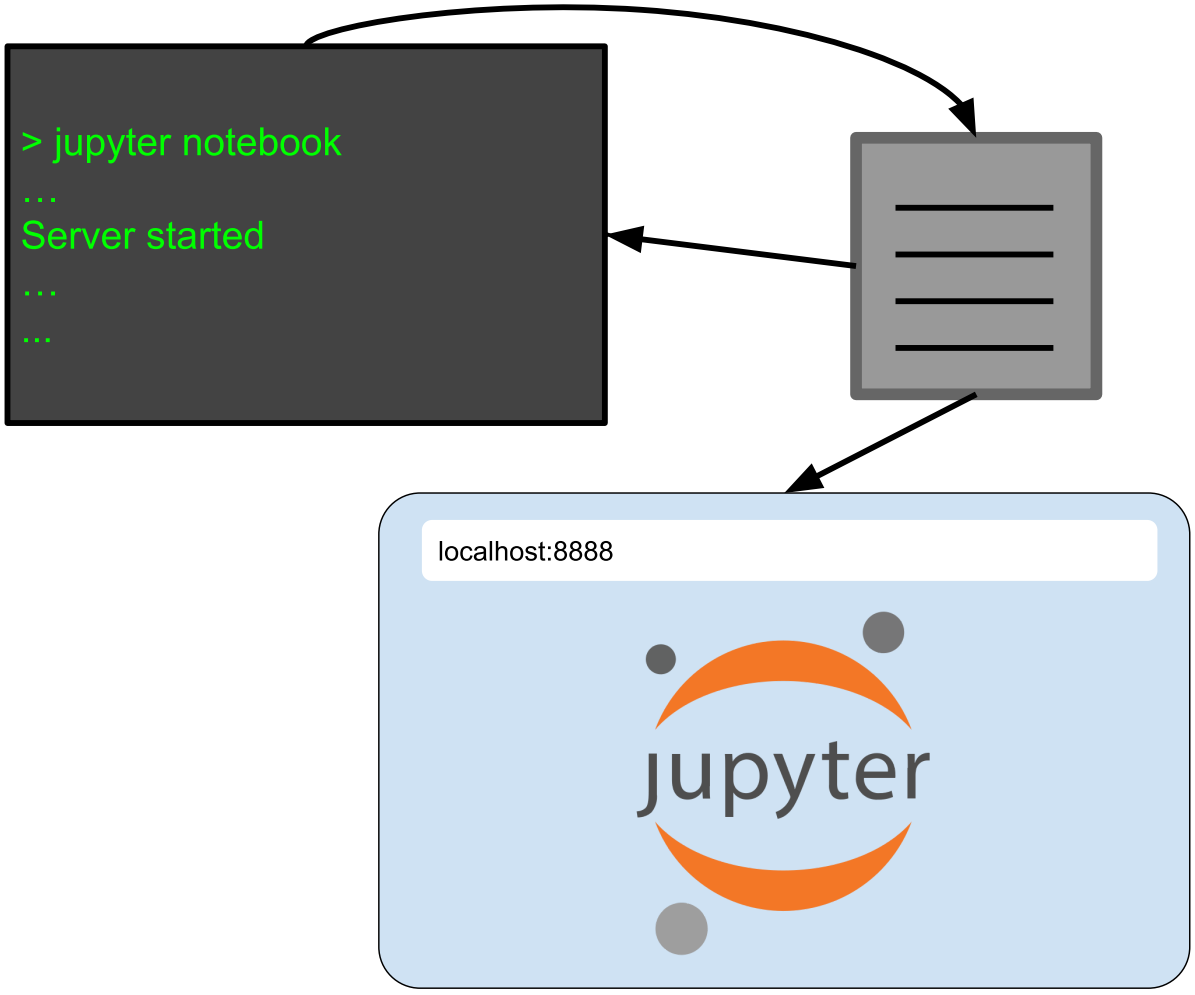
1.6. Jupyter Lab and Jupyter notebooks

Launch a `jupyter lab` server:

- on Windows, use anaconda terminal
- on Mac/Linux, use terminal
- `cd path/to/where/you/save/notes`
- enter `jupyter lab`

1.6.1. What just happened?

- launched a local web server
- opened a new browser tab pointed to it



1.6.2. A jupyter notebook tour

A Jupyter notebook has two modes. When you first open, it is in command mode. It says the mode in the bottom right of the screen. Each bos is a cell, the highlighted cell is gray when in command mode.

When you press a key in command mode it works like a shortcut. For example `p` shows the command search menu.

If you press `enter` (or `return`) or click on the highlighted cell, which is the boxes we can type in, it changes to edit mode.

There are two type of cells that we will used: code and markdown. You can change that in command mode with `y` for code and `m` for markdown or on the cell type menu at the top of the notebook.

This is a markdown cell

- we can make
- itemized lists of
- bullet points

1. and we can make nubmered
2. lists, and not have to worry
3. about renumbering them
4. if we add a step in the middle later

```
3+4
```

```
7
```

the output here is the value returned by the python interpretter for the last line of the cell

We can set variables

```
name = 'Sarah'
```

The notebook displays nothing when we do an assignment, bcause it returns nothing

we can put a variable there to see it

```
name
```

```
'Sarah'
```

```
name_list = ['Sarah', 'Adam', 'Alex']
```

Skip to main content

```
name_list
```

```
['Sarah', 'Adam', 'Alex']
```

Note

built in functions turn green in jupyter

```
print(name_list)
```

```
['Sarah', 'Adam', 'Alex']
```

Common command mode actions:

- m: switch cell to markdown
- y: switch cell to code
- a: add a cell above
- b: add a cell below
- c: copy cell
- v: paste the cell
- 0 + 0: restart kernel
- p: command menu

use enter/return to get to edit mode

1.7. Getting Help in Jupyter

Getting help is important in programming

When your cursor is inside the `()` of a function if you hold the shift key and press tab it will open a popup with information. If you press tab twice, it gets bigger and three times will make a popup window.

Python has a `print` function and we can use the help in jupyter to learn about how to use it in different ways.

```
year = '2020'
```

```
print(name, year)
```

```
Sarah 2020
```

Skip to main content

> **Tip**
>
> This is an added tip for the notes that I did not show in class, it's less useful while working, but is helpful for the notes

We can also use the `help` function

```
help(print)
```

```
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

The first line says that it can take multiple values, because it says `args*`. The `*` means multiple.

It also has a keyword argument (must be used like `argument=value` and has a default) described as `sep=' '`. This means that by default it adds a space as above.

The help also tells us about other parameters, like the `sep` one

```
print(name, year,'sdkjfdsk', 'ksdjfosidj',sep='\n')
```

```
Sarah
2020
sdkjfdsk
ksdjfosidj
```

Basic programming is a prereq and we will go faster soon, but the goal of this review was to understand notebooks, getting help, and reading docstrings

> **Important**
>
> More extra tips to help you review/refresh your Python that we did not cover in class. In general, I will not require things that we do not at least get very close to in assignments, but since this is supposed to be in the prerequisite, I am providing resources.

1.8. Python Review

Official source on python:

- pep8 official style

Skip to main content

We will go quickly through these focusing on pythonic style, because the prerequisite is a programming course.

1.9. Functions

```
def greeting(name):
    '''
    say hi to a person

    Parameters
    ----------
    name : string
        the name of the person to greet
    '''
    return 'hi ' + name
```

A few things to note:

- the `def` keyword starts a function
- then the name of the function
- parameters in `()` then `:`
- the body is indented
- the first thing in the body should be a docstring, denoted in `'''` which is a multiline comment
- returning is more reliable than printing in a function

In python, PEP 257 says how to write a docstring, but it is very broad.

In Data Science, numpydoc style docstrings are popular.

- Pandas follows numpydoc
- [Numpy uses it]
- Scipy follows numpydoc

Once the cell with the function definition is run, we can use the function

```
greeting(name)
```

```
'hi Sarah'
```

```
print(greeting('surbhi'))
```

```
hi surbhi
```

```
assert greeting('sarah') == 'hi sarah'
```

With a return this works to check that it does the right thing.

Skip to main content

when assert is true, it returns nothing, it throws an error on failure

1.10. Conditionals

```
def greeting2(name,formal=False):
    '''
    say hi to a person

    Parameters
    +++++++++-
    name : string
        the name of the person to greet
    formal: bool
        if the greeting should formal (hello) or not (hi)
    '''
    if formal:
        message = 'hello  ' + name
    else:
        message = 'hi ' + name
    return message
```

key points in this function:

- an `if` also has the conditional part indented
- for a `bool` variable we can just use the variable
- we can set a default value

because of the default value we do not have to pass the second variable:

```
greeting2(name)
```

```
'hi Sarah'
```

```
greeting2(name,True)
```

```
'hello  Sarah'
```

1.11. More Reading

Reading chapter 1 of think like a data scientist will help you with the data science definition part of the assignment.

Think like a data scientist is written for practitioners; not as a text book for a class. It does not have a lot of prerequisite background, but the sections of it that I assign will help you build a better mental picture of what doing Data Science about.

Only the first assignment will be due this fast, it's a short review and setup assignment. It's due quickly so that we know that you have everything set up and the prerequisite material before we start new material next week.

Skip to main content

1.12. Questions

1.12.1. Do I need anaconda, or can I just install each packagae? Ie. I just installed jupyter labs using an external terminal command, can I do that for each package we use?”

Yes, and even better, I made a requirements.txt file with all of what you should need.

2. Iterables and Pandas Data Frames

2.1. House Keeping

2.2. Assignment 1

You can revise it to fix anything you learned today before I give feedback.

2.3. Closing Jupyter server.

In the terminal use Ctrl+C (actually control, not command on mac).

It will ask you a question and give options, read and follow

or

do ctrl+C a second time.

A jupyter server typically runs at `localhost:8888`, but if you have multiple servers running the count increases.

Once I saw a student in office hours working on `localhost:8894` asking why their code kept crashing.

> **Important**
>
> Remember to close your jupyter server

2.4. Using Pandas

We will use data with a library called pandas. By convention, we import it like:

```
import pandas as pd
```

Skip to main content

- `pandas` is the name of the package that is installed
- `as` keyword allows us to assign an alias (nickname)
- `pd` is the typical alias for pandas

2.5. Everything is Data

Data we will see:

- tabular data
- websites as data
- activity logs on websites
- images
- text

2.6. Why inspection in code?

Some IDEs give you GUI based tools to inspect objects. We are going to do it programmatically inline with our analyses for two reasons.

- (minor, logistical) it helps make for good notes
- (most importantly) it helps build habits of data science

In data science, our code will be aiming to tell a story.

If you're curious about something, try it out, see what happens. We're going to use a lot of code inspection tools during class. These are helpful both for understanding what's going on, but the advantage to knowing how to get this information programmatically even though a different IDE would give you inspection tools is that it helps you treat your code as data.

2.7. everything is an object

let's examine the `type` of some variables:

```
a=4
b = 'monday'
c =5.3
d = print
```

```
type(a)
```

```
int
```

` ints are a base python type, like they appear in other languages

strings are iterable type, meaning that theycan be indexed into, or their elements iterated over. For a more technical definition,

```
type(b)
```

```
str
```

we can select one element

```
b[0]
```

```
'm'
```

or multiple, this is called slicing.

```
b[0:3]
```

```
'mon'
```

negative numbers count from the right.

```
b[-1]
```

```
'y'
```

```
type (c)
```

```
float
```

a variable can hold a whole function.

```
type(d)
```

```
builtin_function_or_method
```

functions are also objects like any other type in python

we can use the variable just like the function itself

```
d('hello')
```

```
hello
```

Skip to main content

```
print(b)
```

```
monday
```

2.8. Tabular Data

Structured data is easier to work with than other data.

We're going to focus on tabular data for now. At the end of the course, we'll examine images, which are structured, but more complex and text, which is much less structured.

2.9. Getting familiar with the datset

We're going to use a dataset about coffee quality today.

How was this dataset collected?

- reviews added to DB
- then scraped

Where did it come from?

- coffee Quality Institute's trained reviewers.

what format is it provided in?

- csv (Comma Separated Values)

what other information is in this repository?

- the code to scrape and clean the data
- the data before cleaning

It's important to always know where data came from and how it was collected.

This helps you know what is is useful for and what its limitations are.

Further Reading

An important research article on documenting datasets for machine learning is called Datasheets for Datasets these researchers also did a follow up study to better understand how practitioner use datasheets and decide how to use data.

If topics like this are interesting to you, let me know! my research is related to this and I have a lot of students who complete 310 do research in my lab.

2.10. Loading the Coffee Data

Get raw url for the dataset click on the raw button on the csv page, then copy the url.

```
coffee_data_url = 'https://raw.githubusercontent.com/jldbc/coffee-quality-database/master/data/robusta_da
```

Warning

This did not work in class, so I downloaded the data and dragged it to the same folder as my notebook

```
pd.read_csv(coffee_data_url)
```

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Numbe
0	1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	ankole coffee producers	
1	2	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethuraman estate	14/1148/2017/2
2	3	Robusta	andrew hetzel	India	sethuraman estate	NaN	NaN	000
3	4	Robusta	ugacof	Uganda	ugacof project area	NaN	ugacof	
4	5	Robusta	katuka development trust ltd	Uganda	katikamu capca farmers association	NaN	katuka development trust	
5	6	Robusta	andrew hetzel	India	NaN	NaN	(self)	Nal
6	7	Robusta	andrew hetzel	India	sethuraman estates	NaN	NaN	Nal
7	8	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	7	sethuraman estate	14/1148/2017/1
8	9	Robusta	nishant gurjer	India	sethuraman estate	RKR	sethuraman estate	14/1148/2016/1
9	10	Robusta	ugacof	Uganda	ishaka	NaN	nsubuga umar	
10	11	Robusta	ugacof	Uganda	ugacof project area	NaN	ugacof	
11	12	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	RC AB	sethuraman estate	14/1148/2016/1
12	13	Robusta	andrew hetzel	India	sethuraman estates	NaN	NaN	Nal
13	14	Robusta	kasozi coffee farmers association	Uganda	kasozi coffee farmers	NaN	NaN	
14	15	Robusta	ankole coffee producers coop	Uganda	kyangundu coop society	NaN	ankole coffee producers coop union ltd	
15	16	Robusta	andrew hetzel	India	sethuraman estate	NaN	NaN	000
16	17	Robusta	andrew	India	sethuraman	NaN	sethuraman es	Nal

Skip to main content

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Numbe
17	18	Robusta	kawacom uganda ltd	Uganda	bushenyi	NaN	kawacom	
18	19	Robusta	nitubaasa ltd	Uganda	kigezi coffee farmers association	NaN	nitubaasa	
19	20	Robusta	mannya coffee project	Uganda	mannya coffee project	NaN	mannya coffee project	
20	21	Robusta	andrew hetzel	India	sethuraman estates	NaN	NaN	Nal
21	22	Robusta	andrew hetzel	India	sethuraman estates	NaN	sethuraman estates	Nal
22	23	Robusta	andrew hetzel	United States	sethuraman estates	NaN	sethuraman estates	Nal
23	24	Robusta	luis robles	Ecuador	robustasa	Lavado 1	our own lab	Nal
24	25	Robusta	luis robles	Ecuador	robustasa	Lavado 3	own laboratory	Nal
25	26	Robusta	james moore	United States	fazenda cazengo	NaN	cafe cazengo	Nal
26	27	Robusta	cafe politico	India	NaN	NaN	NaN	14-1118-2014 008
27	28	Robusta	cafe politico	Vietnam	NaN	NaN	NaN	Nal

28 rows × 44 columns

If the file is local, you can load it this way. The parameter of the function is the **path** to the dataset, that can be relative, like below, absolute (a full address on your computer) or a URL like above.

```
pd.read_csv('robusta_data_cleaned.csv')
```

This read in the data and printed it out because it is the last line on the cell. If we do something else after, it will read it in, but not print it out.

In order to use it, we save the output to a variable.

```
coffee_df = pd.read_csv(coffee_data_url)
```

we choose this name so that related variables will all use `coffee` and then have other parts after `_` to describe them in terms of type and content. In Python, for variables, the typical convention is to use `_` to join words, not CamelCase, which is used for classes, like `DataFrame`

Skip to main content

```
coffee_df
```

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Numbe
0	1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	ankole coffee producers	
1	2	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethuraman estate	14/1148/2017/2
2	3	Robusta	andrew hetzel	India	sethuraman estate	NaN	NaN	000
3	4	Robusta	ugacof	Uganda	ugacof project area	NaN	ugacof	
4	5	Robusta	katuka development trust ltd	Uganda	katikamu capca farmers association	NaN	katuka development trust	
5	6	Robusta	andrew hetzel	India	NaN	NaN	(self)	Nal
6	7	Robusta	andrew hetzel	India	sethuraman estates	NaN	NaN	Nal
7	8	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	7	sethuraman estate	14/1148/2017/1
8	9	Robusta	nishant gurjer	India	sethuraman estate	RKR	sethuraman estate	14/1148/2016/1
9	10	Robusta	ugacof	Uganda	ishaka	NaN	nsubuga umar	
10	11	Robusta	ugacof	Uganda	ugacof project area	NaN	ugacof	
11	12	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	RC AB	sethuraman estate	14/1148/2016/1
12	13	Robusta	andrew hetzel	India	sethuraman estates	NaN	NaN	Nal
13	14	Robusta	kasozi coffee farmers association	Uganda	kasozi coffee farmers	NaN	NaN	
14	15	Robusta	ankole coffee producers coop	Uganda	kyangundu coop society	NaN	ankole coffee producers coop union ltd	
15	16	Robusta	andrew hetzel	India	sethuraman estate	NaN	NaN	000
16	17	Robusta	andrew	India	sethuraman	NaN	sethuraman es	Nal

Skip to main content

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Numbe
17	18	Robusta	kawacom uganda ltd	Uganda	bushenyi	NaN	kawacom	
18	19	Robusta	nitubaasa ltd	Uganda	kigezi coffee farmers association	NaN	nitubaasa	
19	20	Robusta	mannya coffee project	Uganda	mannya coffee project	NaN	mannya coffee project	
20	21	Robusta	andrew hetzel	India	sethuraman estates	NaN	NaN	Nal
21	22	Robusta	andrew hetzel	India	sethuraman estates	NaN	sethuraman estates	Nal
22	23	Robusta	andrew hetzel	United States	sethuraman estates	NaN	sethuraman estates	Nal
23	24	Robusta	luis robles	Ecuador	robustasa	Lavado 1	our own lab	Nal
24	25	Robusta	luis robles	Ecuador	robustasa	Lavado 3	own laboratory	Nal
25	26	Robusta	james moore	United States	fazenda cazengo	NaN	cafe cazengo	Nal
26	27	Robusta	cafe politico	India	NaN	NaN	NaN	14-1118-2014 008
27	28	Robusta	cafe politico	Vietnam	NaN	NaN	NaN	Nal

28 rows × 44 columns

Next we examine the type

```
type(coffee_df)
```

```
pandas.core.frame.DataFrame
```

This is a new type provided by the `pandas` library, called a dataframe

We can also exmaine its parts. It consists of several; first the column headings

```
coffee_df.columns
```

Skip to main content

```
Index(['Unnamed: 0', 'Species', 'Owner', 'Country.of.Origin', 'Farm.Name',
       'Lot.Number', 'Mill', 'ICO.Number', 'Company', 'Altitude', 'Region',
       'Producer', 'Number.of.Bags', 'Bag.Weight', 'In.Country.Partner',
       'Harvest.Year', 'Grading.Date', 'Owner.1', 'Variety',
       'Processing.Method', 'Fragrance...Aroma', 'Flavor', 'Aftertaste',
       'Salt...Acid', 'Bitter...Sweet', 'Mouthfeel', 'Uniform.Cup',
       'Clean.Cup', 'Balance', 'Cupper.Points', 'Total.Cup.Points', 'Moisture',
       'Category.One.Defects', 'Quakers', 'Color', 'Category.Two.Defects',
       'Expiration', 'Certification.Body', 'Certification.Address',
       'Certification.Contact', 'unit_of_measurement', 'altitude_low_meters',
       'altitude_high_meters', 'altitude_mean_meters'],
      dtype='object')
```

These are a special type called Index that is also provided by pandas.

It also tells us that the actual headings are of `dtype` `object`. `object` is used for strings or columns with mixed types

the `dtype` is slightly different from base Python types and is how pandas classifies but roughly is the same idea as a type.

```
type(coffee_df.columns)
```

```
pandas.core.indexes.base.Index
```

We can look at the first 5 rows with `head`

```
coffee_df.head()
```

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number
0	1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	ankole coffee producers	0
1	2	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethuraman estate	14/1148/2017/21
2	3	Robusta	andrew hetzel	India	sethuraman estate	NaN	NaN	0000
3	4	Robusta	ugacof	Uganda	ugacof project area	NaN	ugacof	0
4	5	Robusta	katuka development trust ltd	Uganda	katikamu capca farmers association	NaN	katuka development trust	0

5 rows × 44 columns

Some notes:

Skip to main content

- `head` is a method, it *does* something to the content and can rely on parameters (here `n=5` can be changed to show different numbers of rows)

If we forget the `()` on a method, it looks weird in the output

```
coffee_df.head
```

```
<bound method NDFrame.head of     Unnamed: 0  Species                                Owner Country.of.Origi
0            1  Robusta       ankole coffee producers coop           Uganda
1            2  Robusta                     nishant gurjer            India
2            3  Robusta                      andrew hetzel            India
3            4  Robusta                             ugacof           Uganda
4            5  Robusta       katuka development trust ltd           Uganda
5            6  Robusta                      andrew hetzel            India
6            7  Robusta                      andrew hetzel            India
7            8  Robusta                     nishant gurjer            India
8            9  Robusta                     nishant gurjer            India
9           10  Robusta                             ugacof           Uganda
10          11  Robusta                             ugacof           Uganda
11          12  Robusta                     nishant gurjer            India
12          13  Robusta                      andrew hetzel            India
13          14  Robusta  kasozi coffee farmers association           Uganda
14          15  Robusta       ankole coffee producers coop           Uganda
15          16  Robusta                      andrew hetzel            India
16          17  Robusta                      andrew hetzel            India
17          18  Robusta                 kawacom uganda ltd           Uganda
18          19  Robusta                      nitubaasa ltd           Uganda
```

We can see more about why this happens with `type`.

```
type(coffee_df.head)
```

```
method
```

Without the parenthesis, it is the literal function object.

```
type(coffee_df.head())
```

```
pandas.core.frame.DataFrame
```

With the parenthesis, it runs the function and `type` examines what it returns, the `DataFrame` object.

2.11. Assignment 1 tips

2.11.1. Pythonic Loops

In Python, we call good style ‘pythonic’, for loops that means making a sensible loop variable. Let’s firs tmake a list object we can iterate over

Skip to main content

```
coffee_cols_list = list(coffee_df.columns)
coffee_cols_list
```

```
['Unnamed: 0',
 'Species',
 'Owner',
 'Country.of.Origin',
 'Farm.Name',
 'Lot.Number',
 'Mill',
 'ICO.Number',
 'Company',
 'Altitude',
 'Region',
 'Producer',
 'Number.of.Bags',
 'Bag.Weight',
 'In.Country.Partner',
 'Harvest.Year',
 'Grading.Date',
 'Owner.1',
 'Variety',
 'Processing.Method',
 'Fragrance...Aroma',
 'Flavor',
 'Aftertaste',
 'Salt...Acid',
 'Bitter...Sweet',
 'Mouthfeel',
 'Uniform.Cup',
 'Clean.Cup',
 'Balance',
 'Cupper.Points',
 'Total.Cup.Points',
 'Moisture',
 'Category.One.Defects',
 'Quakers',
 'Color',
 'Category.Two.Defects',
 'Expiration',
 'Certification.Body',
 'Certification.Address',
 'Certification.Contact',
 'unit_of_measurement',
 'altitude_low_meters',
 'altitude_high_meters',
 'altitude_mean_meters']
```

Now we will write a loop to clean up the `.`

```
clean_cols = []
for col in coffee_cols_list:
    clean_cols.append(col.replace('.','_'))

clean_cols
```

Skip to main content

```
['Unnamed: 0',
 'Species',
 'Owner',
 'Country_of_Origin',
 'Farm_Name',
 'Lot_Number',
 'Mill',
 'ICO_Number',
 'Company',
 'Altitude',
 'Region',
 'Producer',
 'Number_of_Bags',
 'Bag_Weight',
 'In_Country_Partner',
 'Harvest_Year',
 'Grading_Date',
 'Owner_1',
 'Variety',
 'Processing_Method',
 'Fragrance___Aroma',
 'Flavor',
 'Aftertaste',
 'Salt___Acid',
 'Bitter___Sweet',
 'Mouthfeel',
 'Uniform_Cup',
 'Clean_Cup',
 'Balance',
 'Cupper_Points',
 'Total_Cup_Points',
 'Moisture',
 'Category_One_Defects',
 'Quakers',
 'Color',
 'Category_Two_Defects',
 'Expiration',
 'Certification_Body',
 'Certification_Address',
 'Certification_Contact',
 'unit_of_measurement',
 'altitude_low_meters',
 'altitude_high_meters',
 'altitude_mean_meters']
```

This is equivalent, but easier to read than:

```
clean_cols = []
for i in range(len(coffee_cols_list)):
    clean_cols.append(coffee_cols_list[i].replace('.','_'))

clean_cols
```

Skip to main content

```
['Unnamed: 0',
 'Species',
 'Owner',
 'Country_of_Origin',
 'Farm_Name',
 'Lot_Number',
 'Mill',
 'ICO_Number',
 'Company',
 'Altitude',
 'Region',
 'Producer',
 'Number_of_Bags',
 'Bag_Weight',
 'In_Country_Partner',
 'Harvest_Year',
 'Grading_Date',
 'Owner_1',
 'Variety',
 'Processing_Method',
 'Fragrance___Aroma',
 'Flavor',
 'Aftertaste',
 'Salt___Acid',
 'Bitter___Sweet',
 'Mouthfeel',
 'Uniform_Cup',
 'Clean_Cup',
 'Balance',
 'Cupper_Points',
 'Total_Cup_Points',
 'Moisture',
 'Category_One_Defects',
 'Quakers',
 'Color',
 'Category_Two_Defects',
 'Expiration',
 'Certification_Body',
 'Certification_Address',
 'Certification_Contact',
 'unit_of_measurement',
 'altitude_low_meters',
 'altitude_high_meters',
 'altitude_mean_meters']
```

In this version the loop variable `i` is a number we have to use to access what we want, where in the first one the `col` loop variable is the thing we want. Simpler and easier to read, which is better by definition in Python.

To make it better than in class, without a lot of extra logic we can do the `...` first then the single ones:

```
clean_cols = []
for col in coffee_cols_list:
    clean_cols.append(col.replace('...','_').replace('.','_'))

clean_cols
```

Skip to main content

```
['Unnamed: 0',
 'Species',
 'Owner',
 'Country_of_Origin',
 'Farm_Name',
 'Lot_Number',
 'Mill',
 'ICO_Number',
 'Company',
 'Altitude',
 'Region',
 'Producer',
 'Number_of_Bags',
 'Bag_Weight',
 'In_Country_Partner',
 'Harvest_Year',
 'Grading_Date',
 'Owner_1',
 'Variety',
 'Processing_Method',
 'Fragrance_Aroma',
 'Flavor',
 'Aftertaste',
 'Salt_Acid',
 'Bitter_Sweet',
 'Mouthfeel',
 'Uniform_Cup',
 'Clean_Cup',
 'Balance',
 'Cupper_Points',
 'Total_Cup_Points',
 'Moisture',
 'Category_One_Defects',
 'Quakers',
 'Color',
 'Category_Two_Defects',
 'Expiration',
 'Certification_Body',
 'Certification_Address',
 'Certification_Contact',
 'unit_of_measurement',
 'altitude_low_meters',
 'altitude_high_meters',
 'altitude_mean_meters']
```

This shows that we can chain string operations (this will coem in handy at other times).

The above is a good form for all `for` loops in Python, but since it was specifically making a list with append, we could make it more concise with a list comprehension.

```
clean_cols_alt = [clean_cols.append(col.replace('...','_').replace('.','_')) for col in coffee_cols_list
```

these two ways are the same

```
clean_cols_alt == clean_cols
```

```
False
```

Skip to main content

2.12. Conditionals Evaluate in order

recall we set this variable

```
a
```

```
4
```

If we write conditions where they can be both true, but we want the larger one to act, if we put them in this order it never sees the second, because the first is true.

```
if a >1:
    print('greater 1')
elif a >2:
    print('greater 2')
```

```
greater 1
```

This one works.

```
if a >2:
    print('greater 2')
elif a >1:
    print('greater 1')
```

```
greater 2
```

2.13. Questions After Class

2.13.1. What is the name of the inline iteration/loop again in Python?

list comprehension

2.13.2. I just want to know more about github In general as to me, although it's new so it will take some time to get used to, it's still pretty confusing to me

I will hold an optional session for a bit more GitHub. You can also take CSC311 for a lot more detail

2.13.3. when will we learn about the portfolio?

After A2 feedback, which will be the first time it makes sense for you to work on it

2.13.4. How would you attain a level 3 on any given skill?

There are example ideas in the Portfolio section of the website, but it will make more sense after Assignment 2 and then I'll spend more time on it in class again.

2.13.5. Im confused on what this "pandas.core.frame.DataFrame" is

It is the main data type provided by pandas, that represents a table of data. We will keep working with and inspecting them. For a technical description see the api docs, for a high level description, see the getting started tutorial

2.13.6. would like to get feedback on my homework so I can fix any errors I have

You will get feedback and have a chance to revise later.

2.13.7. Can you use any dataset from github using the raw URL and importing it? Can you use any dataset URL or only github?

You can use any URL that has a compatible type of data.

2.13.8. if level 1 is determined by attendence and participation, how can I assure I am getting my lesson 1s fulfilled every class

I am removing the prismia grading this semester, but it seems I missed one reference of that in the syllabus.

2.13.9. If we get an achievement are we gonna see those on github or do we have to keep track of all the achievements we have to see our grade?

in your feedback you will get a table with your current standing each time work is assessed

2.13.10. Can you slow down a little, some times it gets hard to follow along

I will try a little, but also please either message on prismia or raise your hand if you ever fall behind.

3. Pandas Data Frames and More Iterable Types

> **Note**
>
> I was able to fix my certificate problem by running this script from the python install.

We will import pandas

```
import pandas as pd
```

We want to import the coffee data and we will keep working with it over several classes. Sometimes we want to put functions we have written or constants like this in a module of our own that we can import. For today we will put the following in a file called `coffee.py`

```
robusta_data_url = 'https://raw.githubusercontent.com/jldbc/coffee-quality-database/master/data/robusta_d
robusta_data_file = 'robusta_data_cleaned.csv'
```

Now we can import it

```
import coffee
```

and use the variable with a `.`, as an attribute of the module.

```
coffee.robusta_data_url
```

```
'https://raw.githubusercontent.com/jldbc/coffee-quality-database/master/data/robusta_data_cleaned.csv'
```

We could also import the variables directly so that we can use them without the `.`

```
from coffee import robusta_data_url, robusta_data_file
```

like follows

```
ro
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[5], line 1
----> 1 ro

NameError: name 'ro' is not defined
```

This is a `NameError` because the variable does not exist.

When we spell it correctly, it works as desired.

```
robusta_data_url
```

```
'https://raw.githubusercontent.com/jldbc/coffee-quality-database/master/data/robusta_data_cleaned.csv'
```

Then we can use the variable.

Skip to main content

```
coffee_df = pd.read_csv(robusta_data_url)
```

3.1. DataFrame parts

first we will check the type

```
type(coffee_df)
```

```
pandas.core.frame.DataFrame
```

we saw that we can check the first 5 rows with `head`

```
coffee_df.head()
```

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number
0	1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	ankole coffee producers	0
1	2	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethuraman estate	14/1148/2017/21
2	3	Robusta	andrew hetzel	India	sethuraman estate	NaN	NaN	0000
3	4	Robusta	ugacof	Uganda	ugacof project area	NaN	ugacof	0
4	5	Robusta	katuka development trust ltd	Uganda	katikamu capca farmers association	NaN	katuka development trust	0

5 rows × 44 columns

and `tail` to see the last ones.

```
coffee_df.tail(3)
```

Skip to main content

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company
25	26	Robusta	james moore	United States	fazenda cazengo	NaN	cafe cazengo	NaN	globa opportunity func
26	27	Robusta	cafe politico	India	NaN	NaN	NaN	14-1118-2014-0087	cafe politicc
27	28	Robusta	cafe politico	Vietnam	NaN	NaN	NaN	NaN	cafe politicc

3 rows × 44 columns

We note that one of the columns is named `Unnamed 0:` because it does not have a name in the file, for that column.

We can use `shift` + `tab` to see help. We see that the `index_col` parameter to treat that column as the index.

```
coffee_df = pd.read_csv(robusta_data_url,index_col=0)
coffee_df.head(1)
```

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude
1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	ankole coffee producers	0	ankole coffee producers coop	1488

1 rows × 43 columns

3.1.1. Data Frame parts

Now it looks neater and tht columns is the index.

We saw before the columns

```
coffee_df.columns
```

```
Index(['Species', 'Owner', 'Country.of.Origin', 'Farm.Name', 'Lot.Number',
       'Mill', 'ICO.Number', 'Company', 'Altitude', 'Region', 'Producer',
       'Number.of.Bags', 'Bag.Weight', 'In.Country.Partner', 'Harvest.Year',
       'Grading.Date', 'Owner.1', 'Variety', 'Processing.Method',
       'Fragrance...Aroma', 'Flavor', 'Aftertaste', 'Salt...Acid',
       'Bitter...Sweet', 'Mouthfeel', 'Uniform.Cup', 'Clean.Cup', 'Balance',
       'Cupper.Points', 'Total.Cup.Points', 'Moisture', 'Category.One.Defects',
       'Quakers', 'Color', 'Category.Two.Defects', 'Expiration',
       'Certification.Body', 'Certification.Address', 'Certification.Contact',
       'unit_of_measurement', 'altitude_low_meters', 'altitude_high_meters',
       'altitude_mean_meters'],
      dtype='object')
```

The rows are named by the `index` attribute

Skip to main content

```
coffee_df.index
```

```
Index([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
       19, 20, 21, 22, 23, 24, 25, 26, 27, 28],
      dtype='int64')
```

Both of these are pandas Index objects

We can also take out only the values without the index or the column names.

```
coffee_df.values
```

```
array([['Robusta', 'ankole coffee producers coop', 'Uganda', ..., 1488.0,
        1488.0, 1488.0],
       ['Robusta', 'nishant gurjer', 'India', ..., 3170.0, 3170.0,
        3170.0],
       ['Robusta', 'andrew hetzel', 'India', ..., 1000.0, 1000.0, 1000.0],
       ...,
       ['Robusta', 'james moore', 'United States', ..., 795.0, 795.0,
        795.0],
       ['Robusta', 'cafe politico', 'India', ..., nan, nan, nan],
       ['Robusta', 'cafe politico', 'Vietnam', ..., nan, nan, nan]],
      dtype=object)
```

this is a `numpy.array` type

```
type(coffee_df.values)
```

```
numpy.ndarray
```

3.1.2. Segmenting rows or columns

We can pick out a column using it's name and `[]`

```
coffee_df['Flavor']
```

```
1     8.08
2     7.75
3     7.83
4     7.92
5     7.83
6     7.92
7     7.75
8     7.75
9     7.75
10    7.83
11    7.92
12    7.83
13    7.58
14    7.75
15    7.67
16    7.75
17    7.50
18    7.58
19    7.58
20    7.42
21    7.67
22    7.42
23    7.50
24    7.58
25    7.67
26    7.33
27    6.83
28    6.67
Name: Flavor, dtype: float64
```

this is a pandas Series

```
type(coffee_df['Flavor'])
```

```
pandas.core.series.Series
```

We can pick a row out with `loc`, the property that grabs a row by its name, a value from the index.

```
coffee_df.loc[7]
```

```
Species                                                Robusta
Owner                                           andrew hetzel
Country.of.Origin                                       India
Farm.Name                                  sethuraman estates
Lot.Number                                                NaN
Mill                                                      NaN
ICO.Number                                                NaN
Company                                           cafemakers
Altitude                                                750m
Region                                             chikmagalur
Producer                                        Nishant Gurjer
Number.of.Bags                                            320
Bag.Weight                                              2 kg
In.Country.Partner                 Specialty Coffee Association
Harvest.Year                                            2014
Grading.Date                                    May 15th, 2014
Owner.1                                          Andrew Hetzel
Variety                                                   NaN
Processing.Method                                         NaN
Fragrance...Aroma                                        7.67
Flavor                                                   7.75
Aftertaste                                               7.83
Salt...Acid                                              7.83
Bitter...Sweet                                            8.0
Mouthfeel                                                7.92
Uniform.Cup                                              10.0
Clean.Cup                                                10.0
Balance                                                  7.75
Cupper.Points                                            7.83
Total.Cup.Points                                        82.58
Moisture                                                  0.0
Category.One.Defects                                        0
Quakers                                                     0
Color                                                   Green
Category.Two.Defects                                        0
Expiration                                      May 15th, 2015
Certification.Body                 Specialty Coffee Association
Certification.Address    ff7c18ad303d4b603ac3f8cff7e611ffc735e720
Certification.Contact    352d0cf7f3e9be14dad7df644ad65efc27605ae2
unit_of_measurement                                         m
altitude_low_meters                                     750.0
altitude_high_meters                                    750.0
altitude_mean_meters                                    750.0
Name: 7, dtype: object
```

this is also a series.

3.1.3. Masking to Select Subsets

We can use a boolean mask or Series with boolean values, to pick out a subset of rows.

To build up to that, we will first evaluate a boolean expression on a column.

```
coffee_df['Flavor']>7.8
```

```
1      True
2     False
3      True
4      True
5      True
6      True
7     False
8     False
9     False
10     True
11     True
12     True
13    False
14    False
15    False
16    False
17    False
18    False
19    False
20    False
21    False
22    False
23    False
24    False
25    False
26    False
27    False
28    False
Name: Flavor, dtype: bool
```

Now, we can save it to a variable, and then use it to mask, by treating it as an index.

```
high_flavor = coffee_df['Flavor']>7.8
coffee_df[high_flavor]
```

Skip to main content

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Compa
1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	ankole coffee producers	0	ank cof produc cc
3	Robusta	andrew hetzel	India	sethuraman estate	NaN	NaN	0000	sethuram est
4	Robusta	ugacof	Uganda	ugacof project area	NaN	ugacof	0	ugacof
5	Robusta	katuka development trust ltd	Uganda	katikamu capca farmers association	NaN	katuka development trust	0	katu developm trust
6	Robusta	andrew hetzel	India	NaN	NaN	(self)	NaN	cafemake
10	Robusta	ugacof	Uganda	ishaka	NaN	nsubuga umar	0	ugacof
11	Robusta	ugacof	Uganda	ugacof project area	NaN	ugacof	0	ugacof
12	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	RC AB	sethuraman estate	14/1148/2016/12	kaapi roy

8 rows × 43 columns

this segments out only the selected rows.

The values are a numpy array which also has the same shape as the dataframe, but it does not incude the index or the column headers (which do not count as rows or columns anyway)

```
coffee_vals = coffee_df.values
```

We can see this with the `shape` attribute

```
coffee_vals.shape
```

```
(28, 43)
```

```
coffee_df.shape
```

```
(28, 43)
```

Skip to main content

3.2. Reading from a website

We saw in the help that pandas has a lot of `read_` methods. You can also see a list on the IO page of the docs

```
achievements_url = 'https://rhodyprog4ds.github.io/BrownFall24/syllabus/achievements.html'
```

We can read from html files (websites)

```
pd.read_html(achievements_url,)
```

```
[   Unnamed: 0_level_0                                          topics  \
                  week                                Unnamed: 1_level_1
 0                   1                            [admin, python review]
 1                   2                      Loading data, Python review
 2                   3                        Exploratory Data Analysis
 3                   4                                    Data Cleaning
 4                   5                  Databases, Merging DataFrames
 5                   6  Modeling, classification performance metrics, ...
 6                   7                      Naive Bayes, decision trees
 7                   8                                       Regression
 8                   9                                       Clustering
 9                  10                           SVM, parameter tuning
 10                  11                           KNN, Model comparison
 11                  12                                   Text Analysis
 12                  13                                 Images Analysis
 13                  14                                   Deep Learning

                            skills
                  Unnamed: 2_level_1
 0                         process
 1      [access, prepare, summarize]
 2            [summarize, visualize]
 3   [prepare, summarize, visualize]
 4    [access, construct, summarize]
 5                       [evaluate]
 6         [classification, evaluate]
 7             [regression, evaluate]
 8             [clustering, evaluate]
 9                  [optimize, tools]
 10                  [compare, tools]
 11                    [unstructured]
 12             [unstructured, tools]
 13                  [tools, compare]  ,
    Unnamed: 0_level_0                                           skill  \
                keyword                                Unnamed: 1_level_1
 0               python                             pythonic code writing
 1              process                 describe data science as a process
 2               access                    access data in multiple formats
 3            construct           construct datasets from multiple sources
 4            summarize                       Summarize and describe data
 5            visualize                                   Visualize data
 6              prepare                         prepare data for analysis
 7             evaluate                        Evaluate model performance
 8       classification                              Apply classification
 9           regression                                  Apply Regression
 10          clustering                                       Clustering
 11            optimize                         Optimize model parameters
 12             compare                                   compare models
 13      representation          Choose representations and transform data
 14            workflow  use industry standard data science tools and w...

                                          Level 1  \
                                Unnamed: 2_level_1
 0   python code that mostly runs, occasional pep8 ...
 1            Identify basic components of data science
 2   load data from at least one format; identify t...
 3   identify what should happen to merge datasets ...
 4   Describe the shape and structure of a dataset ...
 5   identify plot types, generate basic plots from...
 6   identify if data is or is not ready for analys...
 7   Explain and compute basic performance metrics ...
 8   identify and describe what classification is, ...
 9   identify what data that can be used for regres...
 10                       describe what clustering is
 11  Identify when model parameters need to be opti...
 12              Qualitatively compare model classes
```

```
                                        Level 2  \
                                Unnamed: 3_level_1
0   python code that reliably runs, frequent pep8 ...
1   Describe and define each stage of the data sci...
2   Load data for processing from the most common ...
3                                apply basic merges
4   compute summary statndard statistics of a whol...
5   generate multiple plot types with complete lab...
6   apply data reshaping, cleaning, and filtering ...
7   Apply and interpret basic model evaluation met...
8   fit, apply, and interpret preselected classifi...
9          fit and interpret linear regression models
10                           apply basic clustering
11  Optimize basic model parameters such as model ...
12  Compare model classes in specific terms and fi...
13  Apply at least one representation to transform...
14  Solve well-strucutred, open-ended problems, ap...

                                        Level 3
                                Unnamed: 4_level_1
0   reliable, efficient, pythonic code that consis...
1   Compare different ways that data science can f...
2   access data from both common and uncommon form...
3        merge data that is not automatically aligned
4   Compute and interpret various summary statisti...
5   generate complex plots with pandas and plottin...
6   apply data reshaping, cleaning, and filtering ...
7   Evaluate a model with multiple metrics and cro...
8   fit and apply classification models and select...
9   fit and explain regrularized or nonlinear regr...
10  apply multiple clustering techniques, and inte...
11  Select optimal parameters based of mutiple qua...
12  Evaluate tradeoffs between different model com...
13  apply transformations in different contexts OR...
14  Independently scope and solve realistic data s...  ,
   Unnamed: 0_level_0                A1                A2  \
              keyword Unnamed: 1_level_1 Unnamed: 2_level_1
0             python                  1                 1
1            process                  1                 0
2             access                  0                 1
3          construct                  0                 0
4          summarize                  0                 0
5          visualize                  0                 0
6            prepare                  0                 0
7           evaluate                  0                 0
8     classification                  0                 0
9         regression                  0                 0
10        clustering                  0                 0
11          optimize                  0                 0
12           compare                  0                 0
13    representation                  0                 0
14          workflow                  0                 0

                  A3                A4                A5  \
   Unnamed: 3_level_1 Unnamed: 4_level_1 Unnamed: 5_level_1
0                   0                 1                 1
1                   0                 0                 0
2                   1                 1                 1
3                   0                 0                 1
4                   1                 1                 1
5                   1                 1                 0
6                   0                 1                 1
7                   0                 0                 0
8                   0                 0                 0
9                   0                 0                 0
10                  0                 0                 0
11                  0                 0                 0
```

```
14                  0                  0                  0

                   A6                 A7                 A8  \
   Unnamed: 6_level_1 Unnamed: 7_level_1 Unnamed: 8_level_1
0                   0                  0                  0
1                   1                  1                  1
2                   0                  0                  0
3                   0                  1                  1
4                   1                  1                  1
5                   1                  1                  1
6                   0                  0                  0
7                   1                  1                  1
8                   0                  1                  0
9                   0                  0                  1
10                  0                  0                  0
11                  0                  0                  0
12                  0                  0                  0
13                  0                  0                  0
14                  0                  0                  0

                   A9                 A10                A11  \
   Unnamed: 9_level_1 Unnamed: 10_level_1 Unnamed: 11_level_1
0                   0                  0                  0
1                   1                  1                  1
2                   0                  0                  0
3                   0                  0                  0
4                   1                  1                  1
5                   1                  1                  1
6                   0                  0                  0
7                   0                  1                  1
8                   0                  1                  0
9                   0                  0                  1
10                  1                  0                  1
11                  0                  1                  1
12                  0                  0                  1
13                  0                  0                  0
14                  0                  1                  1

                  A12                A13       # Assignments
   Unnamed: 12_level_1 Unnamed: 13_level_1 Unnamed: 14_level_1
0                   0                  0                  4
1                   0                  0                  7
2                   0                  0                  4
3                   0                  0                  3
4                   1                  1                 11
5                   1                  1                 10
6                   0                  0                  2
7                   0                  0                  5
8                   0                  0                  2
9                   0                  0                  2
10                  0                  0                  2
11                  0                  0                  2
12                  0                  1                  2
13                  1                  1                  2
14                  1                  1                  4  ]
```

this does not look like the others, it is a list of the DataFrames for all of the tables on the page. Even if there is only one, it will still be in a list.

Now that we know it is a list, we will save it in a variable and then, pick out one DataFrame.

```
ach_df_list = pd.read_html(achievements_url,)
ach_defn = ach_df_list[2]
```

Skip to main content

```
type(ach_defn)
```

```
pandas.core.frame.DataFrame
```

3.3. Slicing

We will use a small list to practice slicing, but it works the same in a DataFrame.

```
toy_list = [3,4,5,6]
```

we can get the last item with `-1`

```
toy_list[-1]
```

```
6
```

We can use `:` to go up to, but not including a particular point.

```
toy_list[:2]
```

```
[3, 4]
```

Note that using `2` picks out the value in position 2, but the above stopped before that.

```
toy_list[2]
```

```
5
```

we can slice from the right, but this time it is inclusive

```
toy_list[-2:]
```

```
[5, 6]
```

we can put numbers on both sides of the `:` to take a segment out

```
toy_list[1:3]
```

```
[4, 5]
```

Skip to main content

we can use two to specify an interval at which to pick values.

```
toy_list[::2]
```

```
[3, 5]
```

or we can state the starting point with it

```
toy_list[1::2]
```

```
[4, 6]
```

3.4. Built in iterable types

These are four different iterable constructions:

```
a = [char for char in 'abcde']
b =  {char:i for i, char in enumerate('abcde')}
c = ('a','b','c','d','e')
d = 'a b c d e'.split(' ')
```

We can see their types

```
type(a), type(b), type(c), type(d)
```

```
(list, dict, tuple, list)
```

Dictionaries are really useful because they consist of key, value pairs. This is really powerful and we will use it a lot to pass complex structures into functions.

we can make another one more manually that is much ike the above.

```
sample_dict = {'a':1,'b':2,'c':3}
sample_dict
```

```
{'a': 1, 'b': 2, 'c': 3}
```

we access values of a dicationary using the key

```
sample_dict['a']
```

Skip to main content

we can iterate over them using the `items` method to pop them off separately.

```
for char,num in sample_dict.items():
    print('char', char, 'is linked to number',num)
```

```
char a is linked to number 1
char b is linked to number 2
char c is linked to number 3
```

3.5. Saving an excerpt

We can find the rows only with a small `Number.of.Bags`, say <100:

```
small_batch_mask = coffee_df['Number.of.Bags']<100
small_batch = coffee_df[small_batch_mask]
```

After we make the mask, then subset the data, we see its a dataframe

```
type(small_batch)
```

```
pandas.core.frame.DataFrame
```

and then we can save it to csv.

```
small_batch.to_csv('robusta_small_batch.csv')
```

4. Exploratory Data Analysis

Now we get to start actual data science!

4.1. This week: Exploratory Data Analysis

- How to summarize data
- Interpretting summaries
- Visualizing data
- interpretting summaries

4.1.1. Summarizing and Visualizing Data are **very** important

- People cannot interpret high dimensional or large samples quickly
- Important in EDA to help you make decisions about the rest of your analysis

Skip to main content

- Summaries are similar calculations to performance metrics we will see later
- visualizations are often essential in debugging models

THEREFORE

- You have a lot of chances to earn summarize and visualize
- we will be picky when we assess if you earned them or not

```
import pandas as pd
```

Today we will work with a new dataset about Carbon emissions

```
carbon_data_url = 'https://github.com/rfordatascience/tidytuesday/raw/master/data/2024/2024-05-21/emissi
```

```
carbon_df = pd.read_csv(carbon_data_url)
```

data readme

```
carbon_df.head()
```

	year	parent_entity	parent_type	commodity	production_value	production_unit	total_emissions_MtCO2e
0	1962	Abu Dhabi National Oil Company	State-owned Entity	Oil & NGL	0.91250	Million bbl/yr	0.363885
1	1962	Abu Dhabi National Oil Company	State-owned Entity	Natural Gas	1.84325	Bcf/yr	0.134355
2	1963	Abu Dhabi National Oil Company	State-owned Entity	Oil & NGL	1.82500	Million bbl/yr	0.727770
3	1963	Abu Dhabi National Oil Company	State-owned Entity	Natural Gas	4.42380	Bcf/yr	0.322453
4	1964	Abu Dhabi National Oil Company	State-owned Entity	Oil & NGL	7.30000	Million bbl/yr	2.911079

In this case, `head` shows that these rows are very similar because it is sorted. Tail would probably be similar, so we will try `sample` to get a random subset.

```
carbon_df.sample(5)
```

Skip to main content

	year	parent_entity	parent_type	commodity	production_value	production_unit	total_emissions_MtC
11618	2013	TurkmenGaz	State-owned Entity	Oil & NGL	55.545980	Million bbl/yr	22.15(
5457	2006	Iraq National Oil Company	State-owned Entity	Oil & NGL	587.972000	Million bbl/yr	234.47(
9795	2012	Santos	Investor-owned Company	Oil & NGL	14.435150	Million bbl/yr	5.756
8324	2003	Petroleos de Venezuela	State-owned Entity	Natural Gas	802.356800	Bcf/yr	58.484
2403	2019	China (Coal)	Nation State	Metallurgical Coal	502.549532	Million tonnes/yr	1490.872

4.2. Describing a Dataset

So far, we've loaded data in a few different ways and then we've examined DataFrames as a data structure, looking at what different attributes they have and what some of the methods are, and how to get data into them.

The `describe` method provides us with a set of summary statistics that broadly

```
carbon_df.describe()
```

	year	production_value	total_emissions_MtCO2e
count	12551.000000	12551.000000	12551.000000
mean	1987.148116	412.712258	113.219734
std	29.202455	1357.569683	329.812666
min	1854.000000	0.004398	0.000321
25%	1973.000000	10.601353	8.785294
50%	1994.000000	63.203536	33.058688
75%	2009.000000	320.664764	102.154596
max	2022.000000	27192.000000	8646.905949

And these all give us a sense of the values and the distribution or spread fo the data in each column.

4.2.1. Individual statistics

We can also extract each of the statistics that the `describe` method calculates individually, by name.

```
carbon_df.max()
```

Skip to main content

```
year                                 2022
parent_entity                         YPF
parent_type               State-owned Entity
commodity                      Thermal Coal
production_value                    27192.0
production_unit            Million tonnes/yr
total_emissions_MtCO2e           8646.905949
dtype: object
```

4.2.2. Understanding Quantiles

The 50% has another more common name: the median. It means 50% of the data are lower (and higher) than this value.

The quantiles are tricky, we cannot just `.25%()` to get the 25% percentile, we have to use the `quantile` method and pass it a value between 0 and 1.

```
carbon_df['production_value'].quantile(.25)
```

```
10.6013534253315
```

```
carbon_df['production_value'].quantile(.8)
```

```
448.5
```

4.3. Individual variable

We can use the descriptive statistics on individual columns as well

4.4. What is the average total emissions in this dataset?

```
carbon_df['total_emissions_MtCO2e'].mean()
```

```
113.21973391166483
```

4.5. Working with categorical data

There are different columns in the describe than the the whole dataset:

```
carbon_df.columns
```

Skip to main content

```
Index(['year', 'parent_entity', 'parent_type', 'commodity', 'production_value',
       'production_unit', 'total_emissions_MtCO2e'],
      dtype='object')
```

So far, the stats above are only for numerical features.

We can get the prevalence of each one with `value_counts`, this can also allow us to answer certain questions. Let's try a few.

4.5.1. What is the most common parent type?

```
carbon_df['parent_type'].value_counts()
```

```
parent_type
Investor-owned Company    6583
State-owned Entity        3914
Nation State              2054
Name: count, dtype: int64
```

▶ Show code cell source

```
'Investor-owned Company'
```

This shows us that the most common one is 'Investor-owned Company'

We also notice that the data is relatively balanced enough that we can make good comparisons from the overall output.

For the specific question, we could instead use the `mode` method (the most frequent value).

```
carbon_df['parent_type'].mode()
```

```
0    Investor-owned Company
Name: parent_type, dtype: object
```

4.5.2. Is it reasonable to compare and use all of the the `commodity` types in the data

To do this, we again look at the value counts:

```
carbon_df['commodity'].value_counts()
```

Skip to main content

```
commodity
Oil & NGL              3733
Natural Gas            3452
Bituminous Coal        1370
Metallurgical Coal     1073
Lignite Coal           1008
Sub-Bituminous Coal     673
Thermal Coal            611
Anthracite Coal         368
Cement                  263
Name: count, dtype: int64
```

Here, we note that the top few are a lot more similar and the smallest one is a lot lower. Here we might drop only the bottom couple or keep them call becaus for a lot of things 200 measurements is enough.

Another choice is that we could combine the different types of coals together. We can do this by adding a column.

```
carbon_df['commodity_simple'] = carbon_df['commodity'].apply(lambda s: s if not('Coal' in s) else 'Coal'
```

The way this works is:

- `carbon_df['commodity']` picks out the column of interest
- the `.apply` method applies a function (that we pass to it) to all of the elements of a series (or to the rows or columns of a DataFrame)
- `lambda s: s if not('Coal' in s) else 'Coal'` is a lambda function that returns the same value if Col is not in it or only `Coal` if it is
- assigns values to a column `'commodity_simple'` `carbon_df['commodity_simple'] =`. It also creates that column since it did not exist

4.5.3. understanding lambda functions

To better understand the lambda function that we used above, let's assign it to a variable.

```
coal_strip = lambda s: s if not('Coal' in s) else 'Coal'
```

First we will inspect it a little:

```
type(coal_strip)
```

```
function
```

We can see it is a function here

Now we can make calls to it, like any other functions

```
coal_strip('Metallurgical Coal')
```

Skip to main content

```
'Coal'
```

this works and now we test the other case:

```
coal_strip('naything else')
```

```
'naything else'
```

4.5.4. How a column is added

Now we can see a sample of the DataFrame again to see how the `apply` method works.

```
carbon_df.sample(5)
```

	year	parent_entity	parent_type	commodity	production_value	production_unit	total_emissions_MtCC
3150	1996	CONSOL Energy	Investor-owned Company	Metallurgical Coal	10.118470	Million tonnes/yr	30.0176
3127	1985	CONSOL Energy	Investor-owned Company	Bituminous Coal	32.571085	Million tonnes/yr	88.4024
284	2019	Alpha Metallurgical Resources	Investor-owned Company	Bituminous Coal	5.936678	Million tonnes/yr	16.1129
7850	2019	Pemex	State-owned Entity	Natural Gas	1031.490000	Bcf/yr	75.1857
5917	1969	Libya National Oil Corp.	State-owned Entity	Oil & NGL	113.442000	Million bbl/yr	45.2381

We can further examine this column:

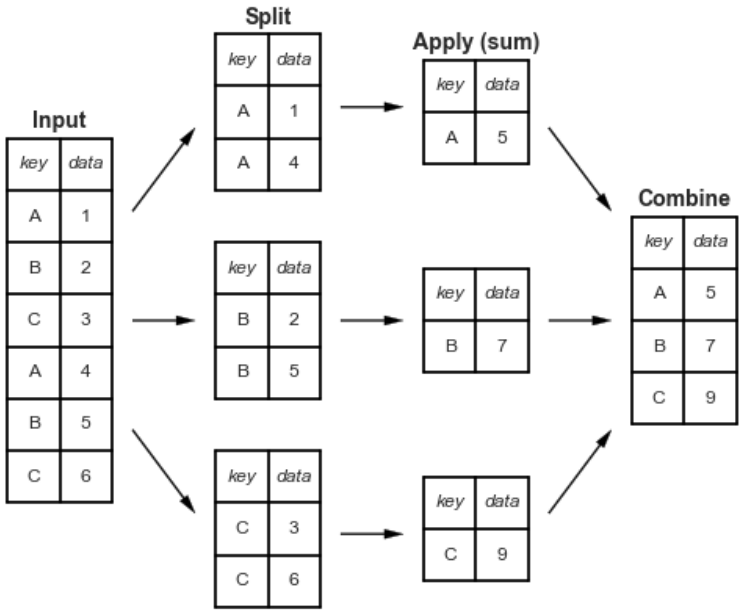
```
carbon_df['commodity_simple'].value_counts()
```

```
commodity_simple
Coal           5103
Oil & NGL      3733
Natural Gas    3452
Cement          263
Name: count, dtype: int64
```

Now the numbers are more similar, we might want to drop the cement, we can do that using a mask that we saw last week.

Skip to main content

4.6. Split-Apply-Combine



see it in action on pandas tutor

4.6.1. Which commodity type has the highest average emissions?

To do this we group by the categorical value, the new simplified commodity column we created, then we pick out thee emissions column, and take the average.

```
carbon_df.groupby('commodity_simple')['total_emissions_MtCO2e'].mean()
```

```
commodity_simple
Cement         112.859242
Coal           118.369779
Natural Gas     75.073875
Oil & NGL      141.479481
Name: total_emissions_MtCO2e, dtype: float64
```

4.6.2. How apply works

We can use `apply` with any function, that we write in advance, that comes from a library, or is built in. Above, we used a lambda we defined on the fly, but here we can make it first.

```
def first_chars(s):
  return s[:5]

carbon_df['commodity'].apply(first_chars)
```

Skip to main content

```
0        Oil &
1        Natur
2        Oil &
3        Natur
4        Oil &
         ...
12546    Natur
12547    Oil &
12548    Natur
12549    Oil &
12550    Natur
Name: commodity, Length: 12551, dtype: object
```

4.6.3. Grouping Multiple times

we can pass a list to group by two values

```
carbon_df.groupby(['commodity_simple','parent_type'])['total_emissions_MtCO2e'].mean()
```

```
commodity_simple  parent_type
Cement            Investor-owned Company     38.813769
                  Nation State              243.802816
Coal              Investor-owned Company     45.397065
                  Nation State              228.621342
                  State-owned Entity         94.977580
Natural Gas       Investor-owned Company     58.313678
                  Nation State              741.445688
                  State-owned Entity         80.152170
Oil & NGL         Investor-owned Company    106.583580
                  Nation State              885.428265
                  State-owned Entity        162.042650
Name: total_emissions_MtCO2e, dtype: float64
```

This is one Series, but is has a multi-index. We can check the type

```
em_by_comm_parent = carbon_df.groupby(['commodity_simple','parent_type'])['total_emissions_MtCO2e'].mean
type(em_by_comm_parent)
```

```
pandas.core.series.Series
```

and look at the index to confirm

```
em_by_comm_parent.index
```

Skip to main content

```
MultiIndex([(     'Cement', 'Investor-owned Company'),
            (     'Cement',           'Nation State'),
            (       'Coal', 'Investor-owned Company'),
            (       'Coal',           'Nation State'),
            (       'Coal',     'State-owned Entity'),
            ('Natural Gas', 'Investor-owned Company'),
            ('Natural Gas',           'Nation State'),
            ('Natural Gas',     'State-owned Entity'),
            (  'Oil & NGL', 'Investor-owned Company'),
            (  'Oil & NGL',           'Nation State'),
            (  'Oil & NGL',     'State-owned Entity')],
           names=['commodity_simple', 'parent_type'])
```

We can use `reset_index` to change from the multi-index into a new count from 0 index (and make it a DataFrame in the process)

```
carbon_df.groupby(['commodity_simple','parent_type'])['total_emissions_MtCO2e'].mean().reset_index()
```

	commodity_simple	parent_type	total_emissions_MtCO2e
0	Cement	Investor-owned Company	38.813769
1	Cement	Nation State	243.802816
2	Coal	Investor-owned Company	45.397065
3	Coal	Nation State	228.621342
4	Coal	State-owned Entity	94.977580
5	Natural Gas	Investor-owned Company	58.313678
6	Natural Gas	Nation State	741.445688
7	Natural Gas	State-owned Entity	80.152170
8	Oil & NGL	Investor-owned Company	106.583580
9	Oil & NGL	Nation State	885.428265
10	Oil & NGL	State-owned Entity	162.042650

4.7. Prepare for next class

Seaborn is a plotting library that gives us *opinionated defaults*

Important

Run this line one time before class on Thursday

```
import seaborn as sns
```

If you get `ModuleNotFound` error, open a terminal tab and run `pip install seaborn` .

Skip to main content

seaborn's alias is `sns` as an inside joke among the developers to the character, Samual Norman Seaborn from West Wing they named the library after per their FAQ

4.8. Questions After Class

> **Important**
>
> Questions about the assignment have been addreessed by adding hints and tips to the end of the instructions

4.8.1. when putting the two column names `["commodity_simple", "parent_type"]` in the square brackets, what did that do?

We cann answer a question like this by inspecting the code further

First we can look directly at it:

```
["commodity_simple", "parent_type"]
```

```
['commodity_simple', 'parent_type']
```

Next, we can check its type:

```
type(["commodity_simple", "parent_type"])
```

```
list
```

It makes a `list` which is then compatible with the `groupby` method.

We can look a its help to remember what it can be passed:

```
# the __doc__ attribute is a property of all functions
# it is a string so I used split to break it into lines,
# looked to see how many i needed then joined them back together
# and printed them (to have the newkines render)
print('\n'.join(carbon_df.groupby.__doc__.split('\n')[:18]))
```

```
Group DataFrame using a mapper or by a Series of columns.

A groupby operation involves some combination of splitting the
object, applying a function, and combining the results. This can be
used to group large amounts of data and compute operations on these
groups.

Parameters
----------
by : mapping, function, label, pd.Grouper or list of such
    Used to determine the groups for the groupby.
    If ``by`` is a function, it's called on each value of the object's
    index. If a dict or Series is passed, the Series or dict VALUES
    will be used to determine the groups (the Series' values are first
    aligned; see ``.align()`` method). If a list or ndarray of length
    equal to the selected axis is passed (see the `groupby user guide
    <https://pandas.pydata.org/pandas-docs/stable/user_guide/groupby.html#splitting-an-object-into-groups
```

4.8.2. How does the apply function work?

I have explained it and linked resources above, plus the `pandas` docs have a section on it in their user guide

4.8.3. How should we be practicing this material. Its hard to tell whats the specific take away from this class vs memorizing methods.

Review the notes and see the analysis.

What we did is ann example data anlysis. We asked and answered several questions. This serves as example questions and types of questions. How to interpret the statistics.

4.8.4. what does the lambda do inside of the function we created to group the coal commodity

`lambda` is the keyword to define a function in line, like `def` is in a full length definition.

4.8.5. Is there a way to scrape data from a website and make it a csv file if the website does not allow for exporting?

Sometimes, we will learn webscraping in a few weeks. For now, download locally, or try a different dataset.

4.8.6. Will be working on forms of data visualization?

Yes

5. Visualization

Data Visualization is about using plots to convey information and get a better understanding of the data

5.1. Plotting in Python

There are several popular plotting libaries:

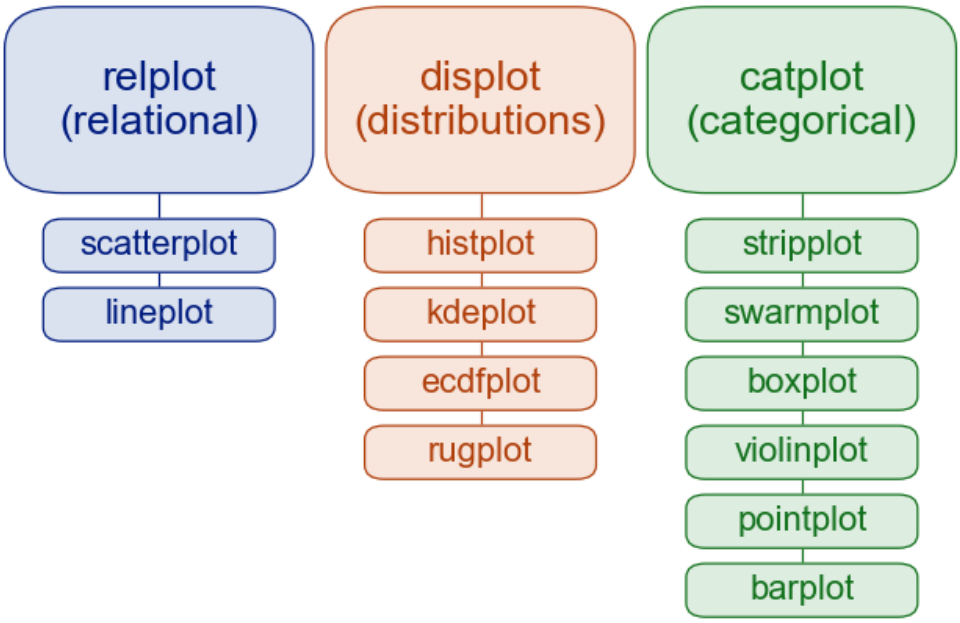
- matplotlib: low level plotting tools
- seaborn: high level plotting with opinionated defaults
- ggplot: plotting based on the ggplot library in R.

Plus pandas has a `plot` method

Pandas and seaborn use matplotlib under the hood.

Seaborn and ggplot both assume the data is set up as a DataFrame. Getting started with seaborn is the simplest, so we'll use that.

5.2. Figure and axis level plots

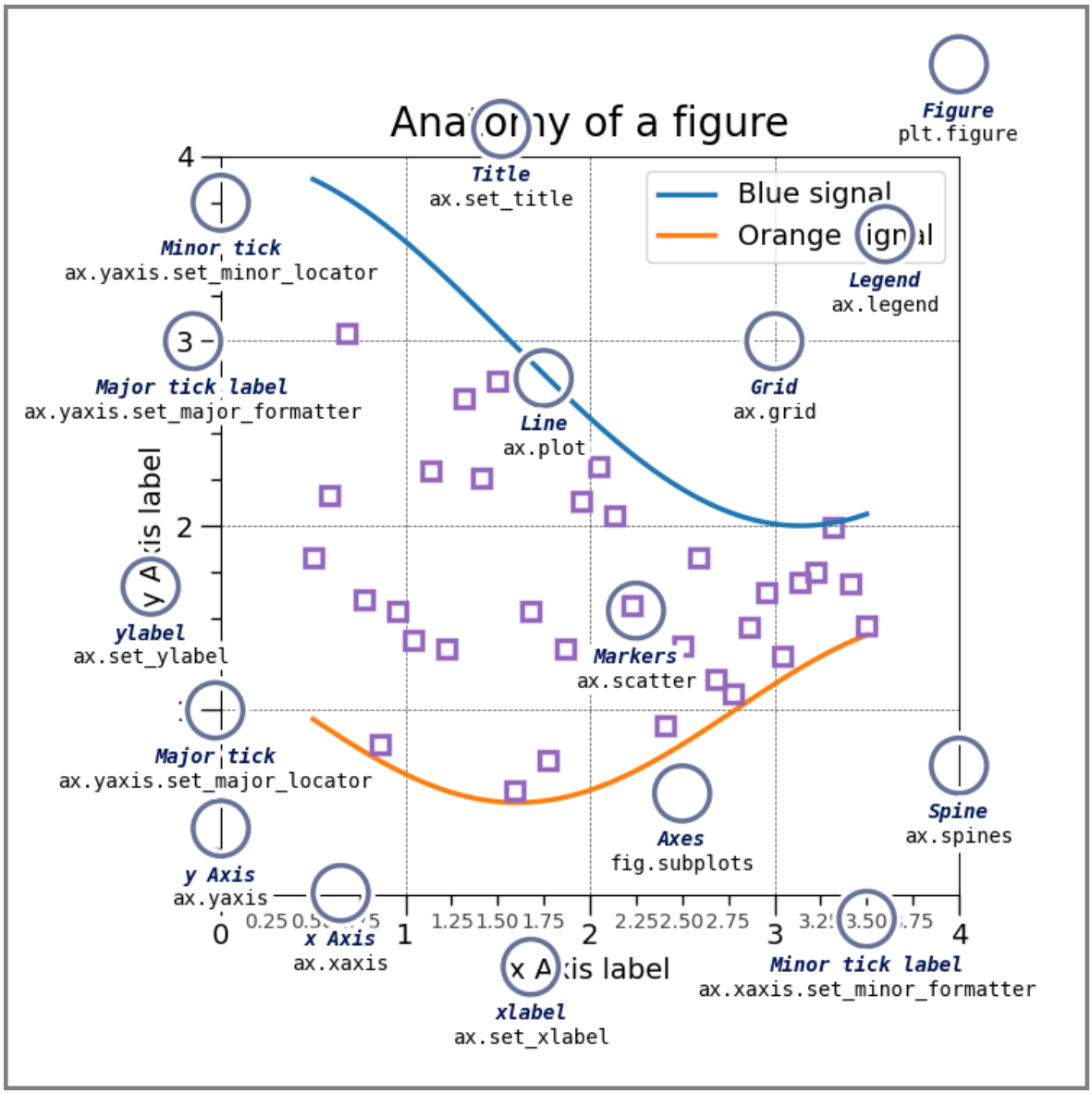


add the image to your notebook with the following:

```
![summary of plot types](https://seaborn.pydata.org/_images/function_overview_8_0.png)
```

Skip to main content

5.3. Anatomy of a figure



*this was drawn with code

add the image to your notebook with the following:

```
![annotated graph](https://matplotlib.org/stable/_images/sphx_glr_anatomy_001.png)
```

figure vs axes

we will load pandas and seaborn

```
import pandas as pd
import seaborn as sns
```

Skip to main content

```
carbon_data_url = 'https://github.com/rfordatascience/tidytuesday/raw/master/data/2024/2024-05-21/emissi
```

and the added column we made last class

```
carbon_df = pd.read_csv(carbon_data_url)
carbon_df['commodity_simple'] = carbon_df['commodity'].apply(lambda s: s if not('Coal' in s) else 'Coal'
```

```
carbon_df.head()
```

	year	parent_entity	parent_type	commodity	production_value	production_unit	total_emissions_MtCO2e
0	1962	Abu Dhabi National Oil Company	State-owned Entity	Oil & NGL	0.91250	Million bbl/yr	0.363885
1	1962	Abu Dhabi National Oil Company	State-owned Entity	Natural Gas	1.84325	Bcf/yr	0.134355
2	1963	Abu Dhabi National Oil Company	State-owned Entity	Oil & NGL	1.82500	Million bbl/yr	0.727770
3	1963	Abu Dhabi National Oil Company	State-owned Entity	Natural Gas	4.42380	Bcf/yr	0.322453
4	1964	Abu Dhabi National Oil Company	State-owned Entity	Oil & NGL	7.30000	Million bbl/yr	2.911079

5.4. How are the samples distributed in time?

We have not yet worked with th `year` column. An important first step we might want to know is how the measurements are distributed in time.

From last class, wemight try `value_counts`

```
carbon_df['year'].value_counts()
```

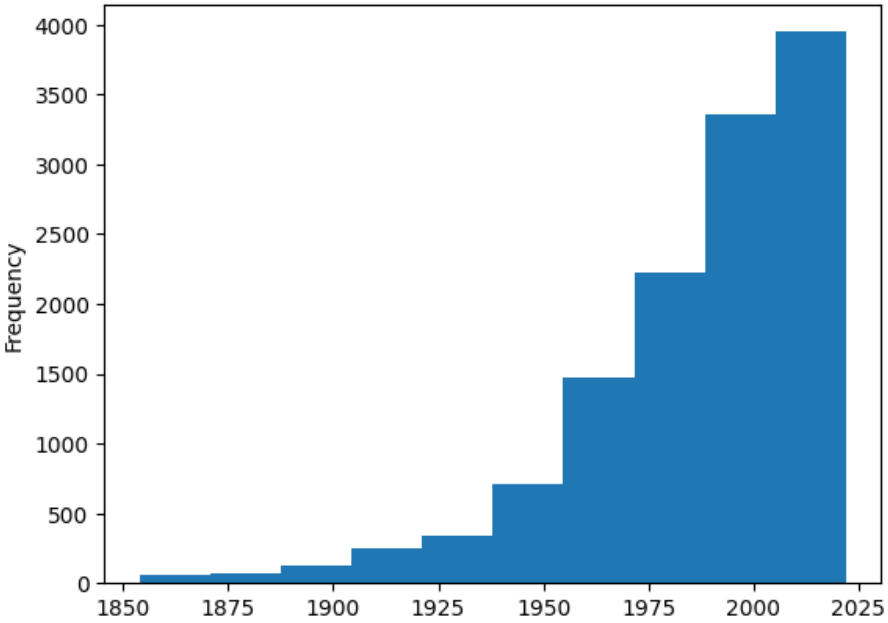
```
year
2021    238
2022    238
2018    237
2019    236
2020    235
       ...
1858      3
1857      3
1856      3
1854      3
1863      3
Name: count, Length: 169, dtype: int64
```

Skip to main content

but it's a little hard to read. A histogram might better

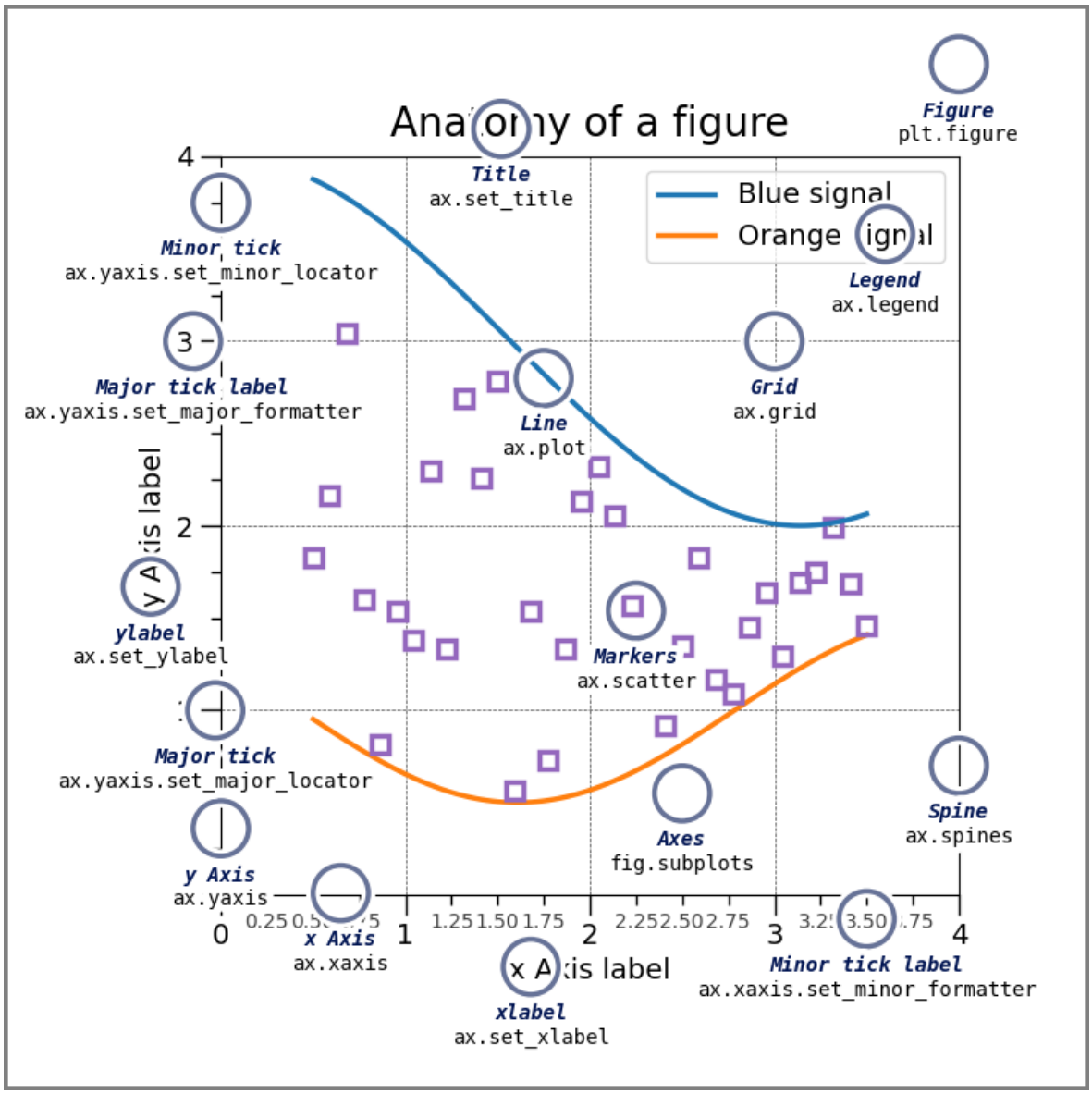
```
carbon_df['year'].plot(kind='hist')
```

```
<Axes: ylabel='Frequency'>
```



Here we see that thare are a lot more samples in more recent years.

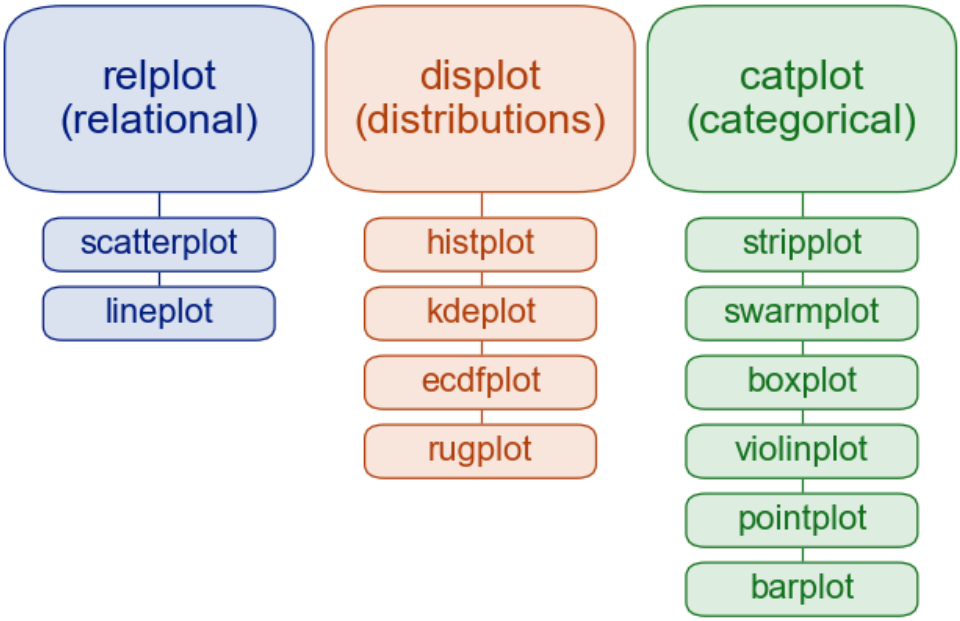
5.5. Anatomy of a plot



the above figure come from matplotlib's Anatomy of a Figure page which includes the *code* to generate that figure

Skip to main content

5.6. Figure and axis level plots



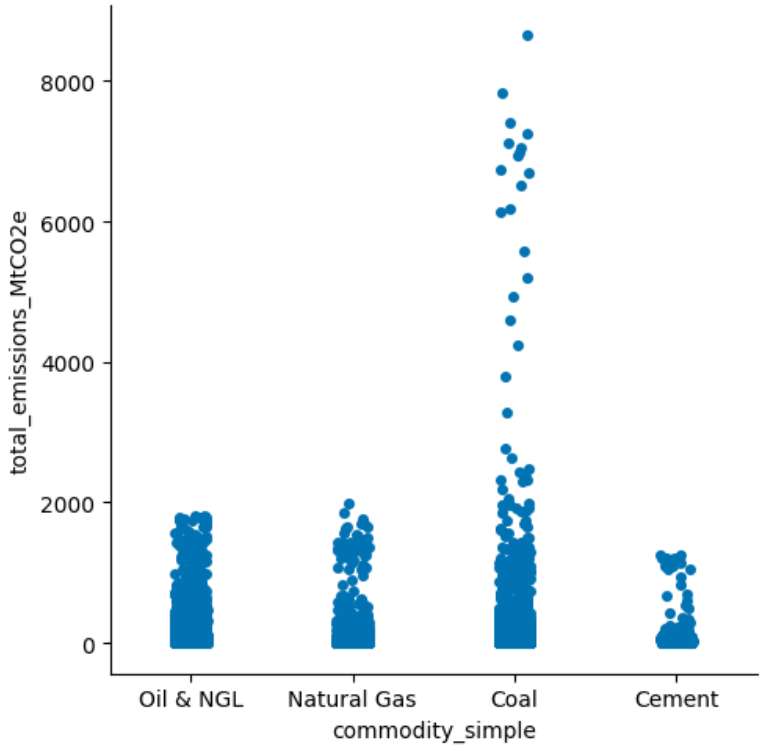
5.7. Changing colors

```
sns.set_palette('colorblind')
```

5.8. Emissions by type

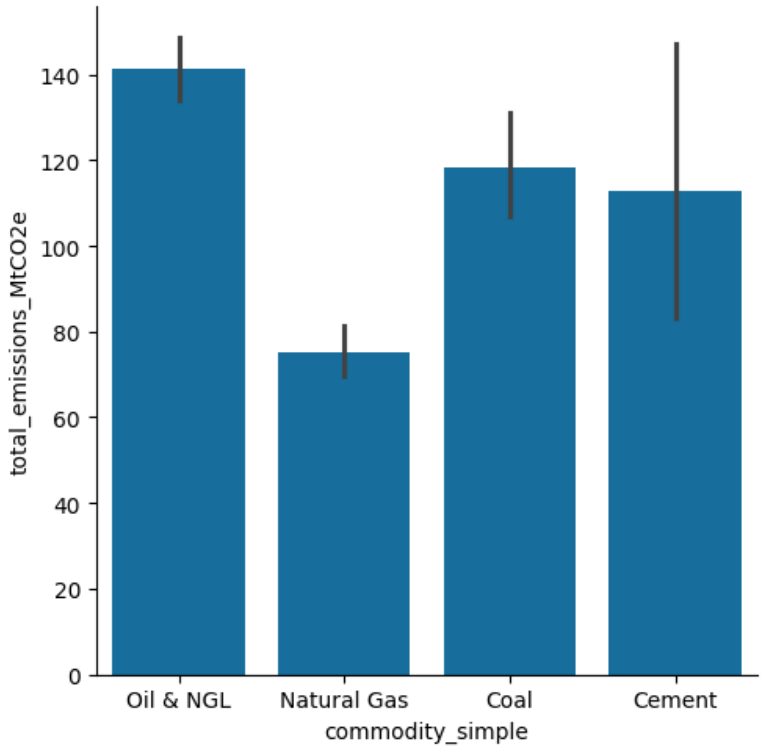
```
sns.catplot(data=carbon_df,x='commodity_simple', y='total_emissions_MtCO2e')
```

```
<seaborn.axisgrid.FacetGrid at 0x7f284f50b4c0>
```




```
sns.catplot(data=carbon_df,x='commodity_simple', y='total_emissions_MtCO2e',kind='bar')
```

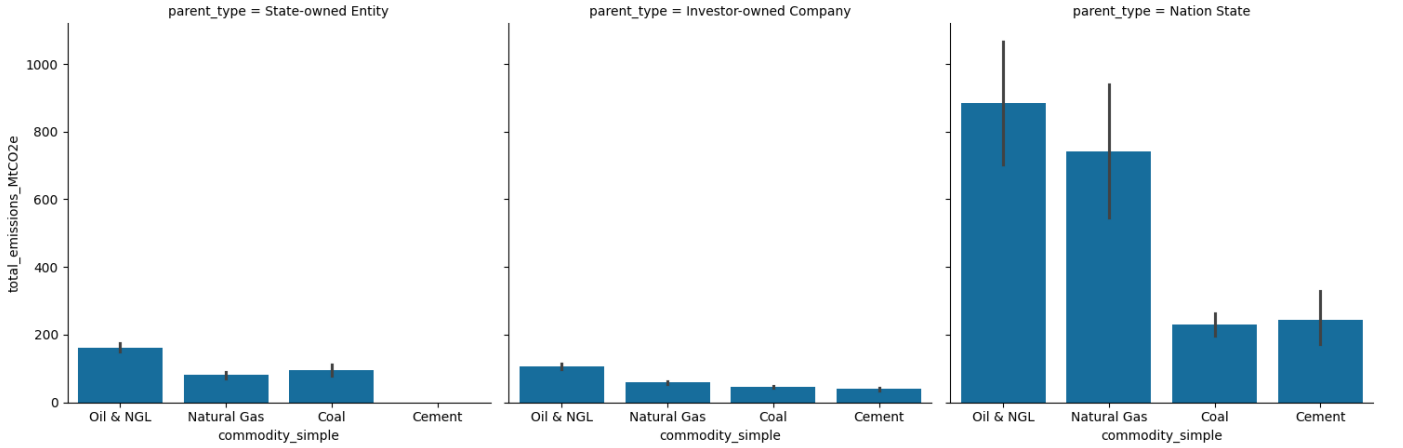
```
<seaborn.axisgrid.FacetGrid at 0x7f28482d6640>
```



Skip to main content

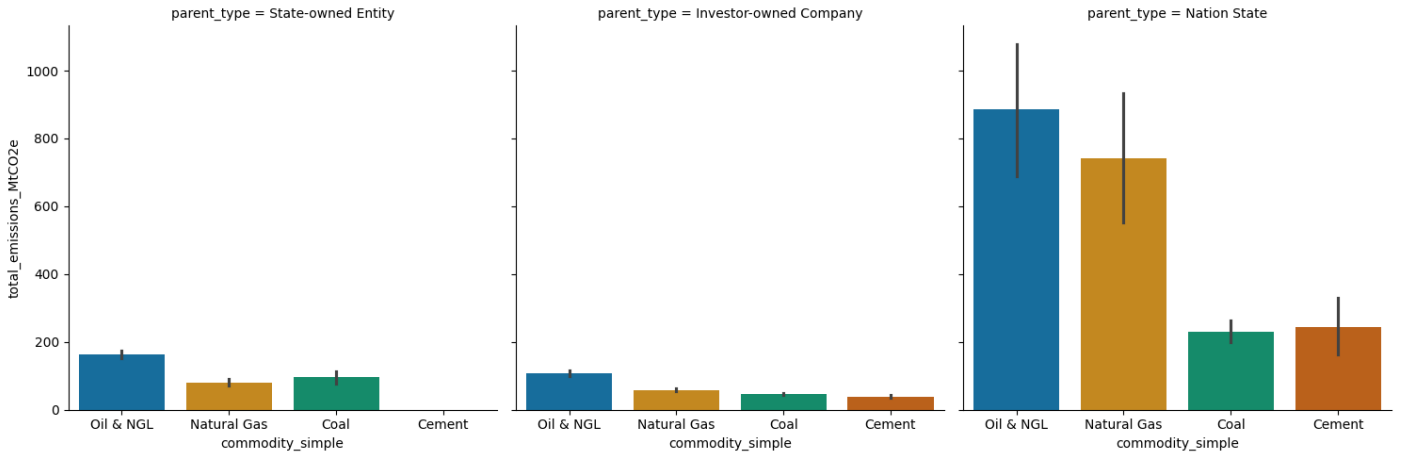
```
sns.catplot(data=carbon_df,x='commodity_simple', y='total_emissions_MtCO2e',kind='bar',
            col = 'parent_type')
```

```
<seaborn.axisgrid.FacetGrid at 0x7f288cdd5eb0>
```




```
sns.catplot(data=carbon_df,x='commodity_simple', y='total_emissions_MtCO2e',kind='bar',
            col = 'parent_type',hue='commodity_simple')
```

```
<seaborn.axisgrid.FacetGrid at 0x7f2847fc5fd0>
```



Example okay questions:

- which parent type has the most constent emissions across commodity type?
- which parent type has highest emission?

Example good questions

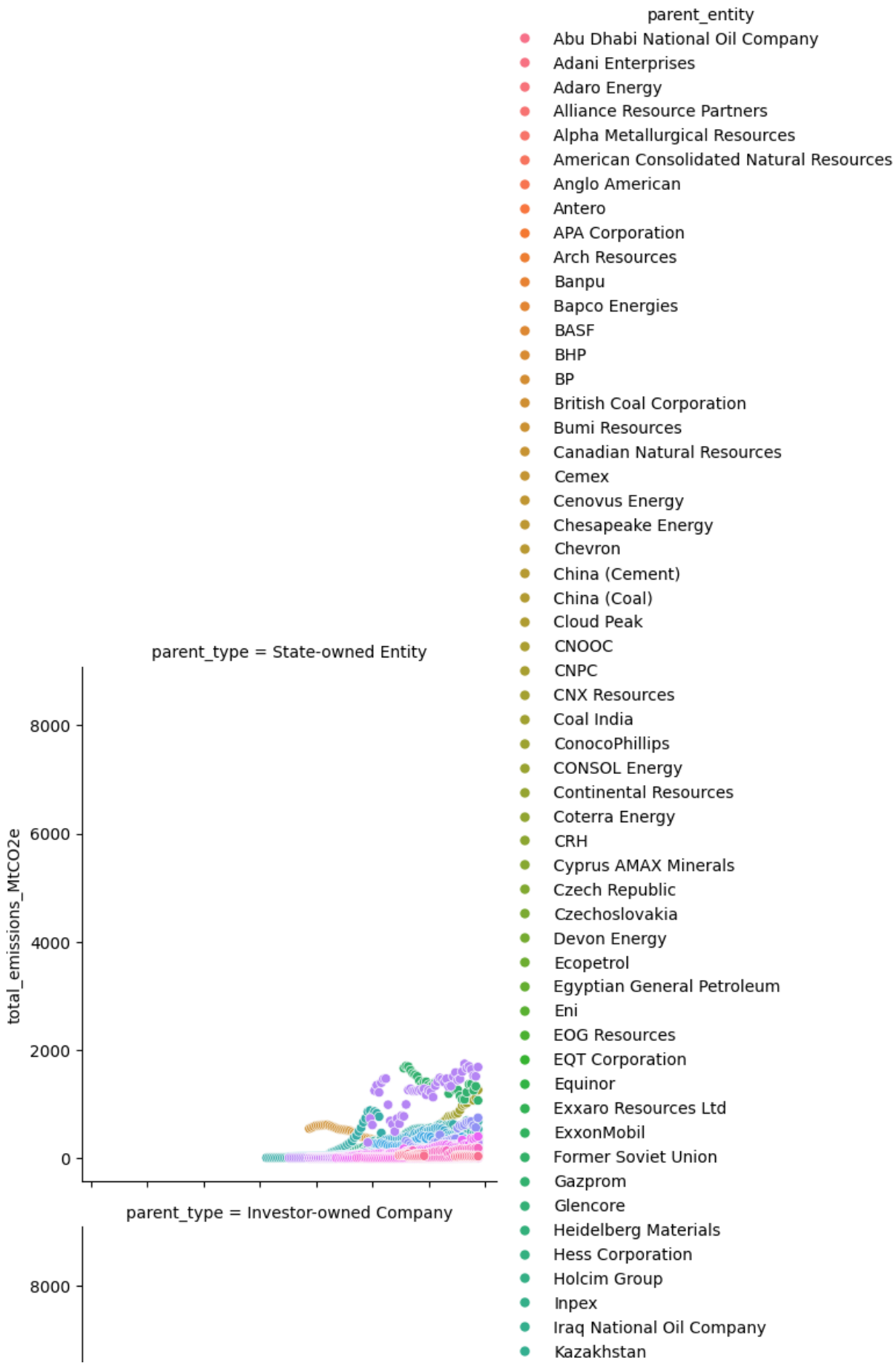
- which type of emissions should be targeted for interventions (the highest)?

Skip to main content

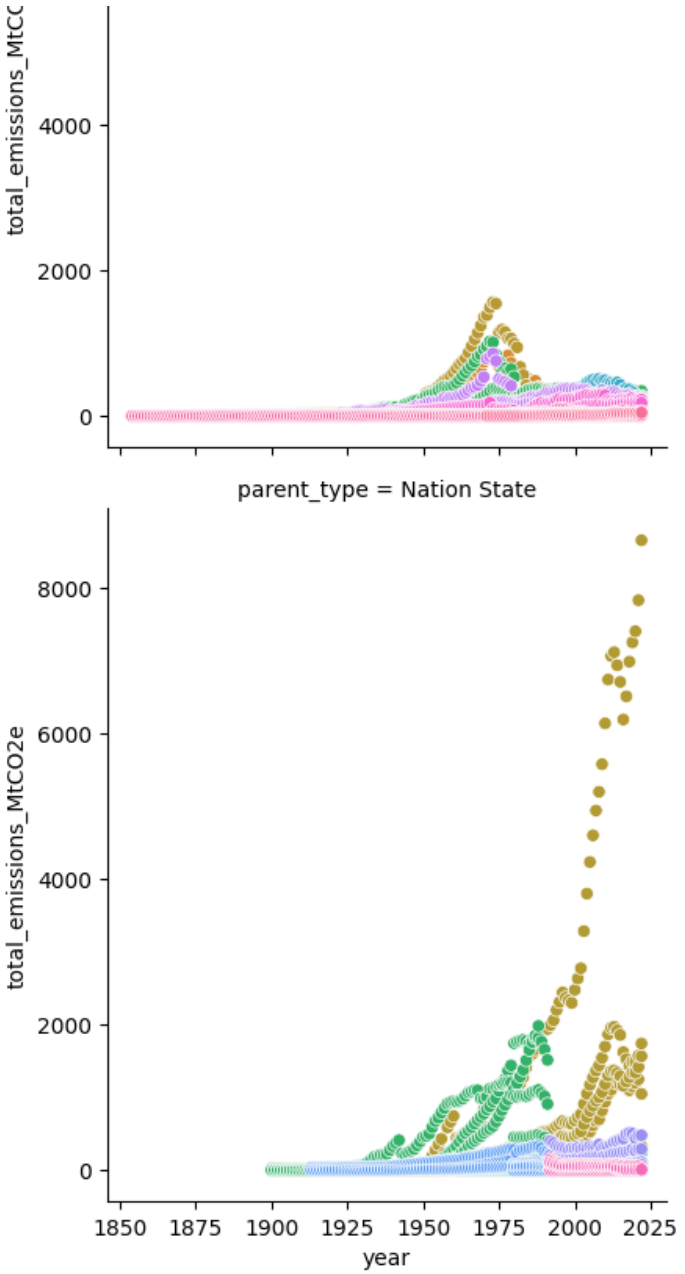
5.9. Emissions over time?

```
sns.relplot(data=carbon_df, x='year', y='total_emissions_MtCO2e',
            hue ='parent_entity',row ='parent_type')
```

```
<seaborn.axisgrid.FacetGrid at 0x7f2847794fd0>
```



Skip to main content



- Libya National Oil Corp.
- Lukoil
- Marathon Oil
- Murphy Oil
- Naftogaz
- National Iranian Oil Co.
- Navajo Transitional Energy Company
- Nigerian National Petroleum Corp.
- North American Coal
- North Korea
- Novatek
- Obsidian Energy
- Occidental Petroleum
- OMV Group
- ONGC India
- Orlen
- Ovintiv
- Peabody Coal Group
- Pemex
- Pertamina
- Petoro
- Petrobras
- PetroEcuador
- Petroleos de Venezuela
- Petroleum Development Oman
- Petronas
- Pioneer Natural Resources
- Poland
- PTTEP
- QatarEnergy
- Repsol
- Rio Tinto
- Rosneft
- Russian Federation
- RWE
- Santos
- Sasol
- Saudi Aramco
- Seriti Resources
- Shell
- Singareni Collieries
- Sinopec
- Slovakia
- SM Energy
- Sonangol
- Sonatrach
- Southwestern Energy
- Suncor Energy
- Surgutneftegas
- Syrian Petroleum
- Taiheiyo Cement
- Teck Resources
- TotalEnergies
- Tourmaline Oil
- Tullow Oil
- TurkmenGaz
- UK Coal

Skip to main content

- Vistra
- Westmoreland Mining
- Whitehaven Coal
- Wolverine Fuels
- Woodside Energy
- YPF

5.10. Variable types and data types

Related but not the same.

Data types are literal, related to the representation in the computer.

ther can be `int16, int32, int64`

We can also have mathematical types of numbers

- Integers can be positive, 0, or negative.
- Reals are continuous, infinite possibilities.

Variable types are about the meaning in a conceptual sense.

- categorical (can take a discrete number of values, could be used to group data, could be a string or integer; unordered)
- continuous (can take on any possible value, always a number)
- binary (like data type boolean, but could be represented as yes/no, true/false, or 1/0, could be categorical also, but often makes sense to calculate rates)
- ordinal (ordered, but appropriately categorical)

we'll focus on the first two most of the time. Some values that are technically only integers range high enough that we treat them more like continuous most of the time.

```
carbon_df.columns
```

```
Index(['year', 'parent_entity', 'parent_type', 'commodity', 'production_value',
       'production_unit', 'total_emissions_MtCO2e', 'commodity_simple'],
      dtype='object')
```

5.11. Questions After Class

Class Response Summary:

5.11.1. To what degree should we be familiarizing ourselves with these different kinds of graphs?

There is also a full semester data visualization class, so we will not cover *everything* it is useful to know a few basic ones and

5.11.2. Should I upload all parts of A2 today if I plan to go to office hours tomorrow? Or just the finished parts?

All of it with your questions written in the file(s).

5.11.3. is there ways to overlap the different parent types into the same graph?

This is called a stacked bar graph, there are examples in the seaborn tutorials for displot but with an important caveat that that can make some things hard to see and you can also stack with the low level features

5.11.4. is the ggplot option just the same method names as the version in R? or is the syntax updated to be similar also?

I think its mostly matching method names, attribute names, and conceptual ideas. Python libraries all have to use Python syntax.

5.11.5. Is the peer review just for assignment 3 or will we have the option to do it for future assignments?

Probably only 3, but possibly a couple more.

5.11.6. What is the typical range of sizes for a good dataset for this assignment

hundred to maybe 2000, you do not need more than that and too many can make it slow

5.11.7. if we don’t get any achievements on an assignment are we able to revise them to get an achievement?

If you are very close, yes, if you are not very close, you will get advice that we recommend you apply on future assignments.

5.11.8. What is a numpy array?

A DataType that one of the attributes of a DataFrame takes. See the glossary entry for numpy array and the intro to DataFrames

6. Cleaning Data - Structure

6.1. Intro

This week, we’ll be cleaning data

Cleaning data is **labor intensive** and requires making *subjective* choices.
We'll focus on, and assess you on, manipulating data correctly, making reasonable choices, and documenting the choices you make carefully.

We'll focus on the programming tools that get used in cleaning data in class this week:

- reshaping data
- handling missing or incorrect values
- renaming columns

```
import pandas as pd
import seaborn as sns

# make plots look nicer and increase font size
sns.set_theme(font_scale=2, palette='colorblind')
```

Note here we set the `theme` and pass the palette and font size as paramters. To learn more about this and the various options, see the seaborn aesthetics tutorial

6.2. What is Tidy Data?

Read in the three csv files described below and store them in a dictionary

```
url_base = 'https://raw.githubusercontent.com/rhodyprog4ds/rhodyds/main/data/'

data_filename_list = ['study_a.csv','study_b.csv','study_c.csv']
```

Here we can use a dictionary comprehension

```
example_data = {datafile.split('.')[0]:pd.read_csv(url_base+datafile) for datafile in data_filename_list}
```

- this is a *dictionary comprehension* which is like a list comprehension, but builds a dictionary instead.
- `split` is a string method that splits the string at the given character and returns a list
- `[0]` then takes the first item out
- `:` makes the part to the left a key and rigt the value
- we can add strings to combine them

6.2.1. Breakign down that comprehension

> **Tip**
>
> This section also illustrates a way you can work to understand **any** piece of code: *take small sections and run them one at a time*

We can do smaller examples of each step

Skip to main content

First a tiny dictionary comprehension

```
{char:ord(char) for char in 'abcdef'}
```

```
{'a': 97, 'b': 98, 'c': 99, 'd': 100, 'e': 101, 'f': 102}
```

this gives us the ascii number for each character in that string with the `char` as key and `ord(char)`'s *output* as the value. Note that it *runs* the code assigned in the value because of the `()` doing the function call.

Next the split method, we will use this sentence.

```
sentence = 'Next the split method, we will use this sentence.'
type(sentence), len(sentence)
```

```
(str, 49)
```

this is type `str` so we can use the string class methods in Python, such as `split`. As a `str` the `len` function tells us how many *characters* because a string is iterable over characters. We used this fact in the tiny dictionary above.

By default it splits at spaces

```
sentence.split()
```

```
['Next', 'the', 'split', 'method,', 'we', 'will', 'use', 'this', 'sentence.']
```

we can instead tell it on what string to split

```
clauses = sentence.split(',')
type(clauses)
```

```
list
```

`split` always returns a list, so we can pick the first item with `[0]`

```
clauses[0]
```

```
'Next the split method'
```

We can concatenate `str` objects with `+`

```
title = 'Dr.'
last = 'Brown'
title + last
```

Skip to main content

```
'Dr.Brown'
```

6.2.2. the same data 3 ways

We can pick a single on out this way

```
example_data['study_a']
```

	name	treatmenta	treatmentb
0	John Smith	-	2
1	Jane Doe	16	11
2	Mary Johnson	3	1

or with an additional import,

```
from IPython.display import display
```

THen we can iterate over the values in the dictionary (the three dataframes) and see them all:

```
[display(df) for df in example_data.values()];
```

	name	treatmenta	treatmentb
0	John Smith	-	2
1	Jane Doe	16	11
2	Mary Johnson	3	1

	intervention	John Smith	Jane Doe	Mary Johnson
0	treatmenta	-	16	3
1	treatmentb	2	11	1

	person	treatment	result
0	John Smith	a	-
1	Jane Doe	a	16
2	Mary Johnson	a	3
3	John Smith	b	2
4	Jane Doe	b	11
5	Mary Johnson	b	1

These three all show the same data, but let's say we have two goals:

Skip to main content

- find the average effect per treatment across people

This works differently for these three versions.

For `study_a` we can easilty get the average per treatment, but to get the average per person, we have to go across rows, which we can do here, but doesn't work as well with plotting

we can work across rows with the `axis` parameter if needed

For B, we get the average per person, but what about per treatment? again we have to go across rows instead.

For the third one, however, we can use groupby, because this one is tidy.

6.3. Encoding Missing values

We can see the impact of a bad choice to represent a missing value with `-`

```
example_data['study_c'].dtypes
```

```
person       object
treatment    object
result       object
dtype: object
```

one non numerical value changed the whole column!

```
example_data['study_c'].describe()
```

	person	treatment	result
count	6	6	6
unique	3	2	6
top	John Smith	a	-
freq	2	3	1

so `describe` treats it all like categorical data

We can use the `na_values` parameter to fix this. Pandas recognizes a lot of different values to mean missing and store as `NaN` but his is not one. Find the full list in the `pd.read_csv` documentation of the `na_values` parameter

```
example_data = {datafile.split('.')[0]:pd.read_csv(url_base+datafile,na_values='-')
                for datafile in data_filename_list}
```

now we can checka gain

```
example_data['study_c'].dtypes
```

Skip to main content

```
person        object
treatment     object
result       float64
dtype: object
```

and we see that it is `float` this is because `NaN` is a float.

```
example_data['study_c']
```

	person	treatment	result
0	John Smith	a	NaN
1	Jane Doe	a	16.0
2	Mary Johnson	a	3.0
3	John Smith	b	2.0
4	Jane Doe	b	11.0
5	Mary Johnson	b	1.0

Now it shows as Nan here

6.4. Computing on Tidy Data

So we can see how to compute on this data to compute both ways.

```
example_data['study_c'].groupby('person')['result'].mean()
```

```
person
Jane Doe        13.5
John Smith       2.0
Mary Johnson     2.0
Name: result, dtype: float64
```

```
example_data['study_c'].groupby('treatment')['result'].mean()
```

```
treatment
a    9.500000
b    4.666667
Name: result, dtype: float64
```

The original Tidy Data paper is worth reading to build a deeper understanding of these ideas.

6.5. Tidying Data

Let's reshape the first one to match the tidy one. First, we will save it to a DataFrame, this makes things easier to read and enables us to use the built in help in jupyter, because it can't check types too many levels into a data structure.

Before

```
example_data['study_a']
```

	name	treatmenta	treatmentb
0	John Smith	NaN	2
1	Jane Doe	16.0	11
2	Mary Johnson	3.0	1

After

```
df_a = example_data['study_a']
```

When we melt a dataset:

- the `id_vars` stay as columns
- the data from the `value_vars` columns become the values in the `value` column
- the column names from the `value_vars` become the values in the `variable` column
- we can rename the value and the variable columns.

```
tall_a = df_a.melt(id_vars='name',var_name='treatment',value_name='result')
tall_a
```

	name	treatment	result
0	John Smith	treatmenta	NaN
1	Jane Doe	treatmenta	16.0
2	Mary Johnson	treatmenta	3.0
3	John Smith	treatmentb	2.0
4	Jane Doe	treatmentb	11.0
5	Mary Johnson	treatmentb	1.0

see visualized on pandas tutor

7. Transforming the Coffee data

Skip to main content

For exmaple, lets only keep the data from the top 10 countries in terms of number of bags.

```
arabica_data_url = 'https://raw.githubusercontent.com/jldbc/coffee-quality-database/master/data/arabica_
# load the data
coffee_df = pd.read_csv(arabica_data_url)
coffee_df.head()
```

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Co
0	1	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	agr deve
1	2	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	agr deve
2	3	Arabica	grounds for health admin	Guatemala	san marcos barrancas "san cristobal cuch	NaN	NaN	NaN	
3	4	Arabica	yidnekachew dabessa	Ethiopia	yidnekachew dabessa coffee plantation	NaN	wolensu	NaN	yidne c pl
4	5	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	agr deve

5 rows × 44 columns

We have separate countries for the country and number of bags

So we make a table the totals how many bags per country

```
bags_per_country = coffee_df.groupby('Country.of.Origin')['Number.of.Bags'].sum()
bags_per_country.head(15)
```

```
Country.of.Origin
Brazil           30534
Burundi            520
China               55
Colombia         41204
Costa Rica       10354
Cote d?Ivoire        2
Ecuador              1
El Salvador       4449
Ethiopia         11761
Guatemala        36868
Haiti              390
Honduras         13167
India               20
Indonesia         1658
Japan               20
Name: Number.of.Bags, dtype: int64
```

Then we can take only the highest 10, and keep only the index

```
# sort descending, keep only the top 10 and pick out only the country names
top_bags_country_list = bags_per_country.sort_values(ascending=False)[:10].index

# filter the original data for only the countries in the top list
top_coffee_df = coffee_df[coffee_df['Country.of.Origin'].isin(top_bags_country_list)]
```

we can see which countries are in the list

```
top_bags_country_list
```

```
Index(['Colombia', 'Guatemala', 'Brazil', 'Mexico', 'Honduras', 'Ethiopia',
       'Costa Rica', 'Nicaragua', 'El Salvador', 'Kenya'],
      dtype='object', name='Country.of.Origin')
```

and confirm that is all that is in our new dataframe

```
top_coffee_df['Country.of.Origin'].value_counts()
```

```
Country.of.Origin
Mexico         236
Colombia       183
Guatemala      181
Brazil         132
Honduras        53
Costa Rica      51
Ethiopia        44
Nicaragua       26
Kenya           25
El Salvador     21
Name: count, dtype: int64
```

compared to the original, it is far fewer

Skip to main content

```
coffee_df['Country.of.Origin'].value_counts()
```

```
Country.of.Origin
Mexico                          236
Colombia                        183
Guatemala                       181
Brazil                          132
Taiwan                           75
United States (Hawaii)           73
Honduras                         53
Costa Rica                       51
Ethiopia                         44
Tanzania, United Republic Of     40
Thailand                         32
Uganda                           26
Nicaragua                        26
Kenya                            25
El Salvador                      21
Indonesia                        20
China                            16
Malawi                           11
Peru                             10
```

Warning

This section is changed slightly from in class to be easier to read.

This limits the rows, but keeps all of the columns

```
top_coffee_df.columns
```

```
Index(['Unnamed: 0', 'Species', 'Owner', 'Country.of.Origin', 'Farm.Name',
       'Lot.Number', 'Mill', 'ICO.Number', 'Company', 'Altitude', 'Region',
       'Producer', 'Number.of.Bags', 'Bag.Weight', 'In.Country.Partner',
       'Harvest.Year', 'Grading.Date', 'Owner.1', 'Variety',
       'Processing.Method', 'Aroma', 'Flavor', 'Aftertaste', 'Acidity', 'Body',
       'Balance', 'Uniformity', 'Clean.Cup', 'Sweetness', 'Cupper.Points',
       'Total.Cup.Points', 'Moisture', 'Category.One.Defects', 'Quakers',
       'Color', 'Category.Two.Defects', 'Expiration', 'Certification.Body',
       'Certification.Address', 'Certification.Contact', 'unit_of_measurement',
       'altitude_low_meters', 'altitude_high_meters', 'altitude_mean_meters'],
      dtype='object')
```

Since this has a lot of variables, we will make lists of how we want to use different columns in variables before using.

```
value_vars = ['Aroma', 'Flavor', 'Aftertaste', 'Acidity', 'Body',
       'Balance', 'Uniformity', 'Clean.Cup', 'Sweetness', ]
id_vars = ['Species', 'Owner', 'Country.of.Origin']
coffee_tall = top_coffee_df.melt(id_vars=id_vars, value_vars=value_vars,
              var_name='rating')
coffee_tall.head(1)
```

Species	Owner	Country.of.Origin	rating	value

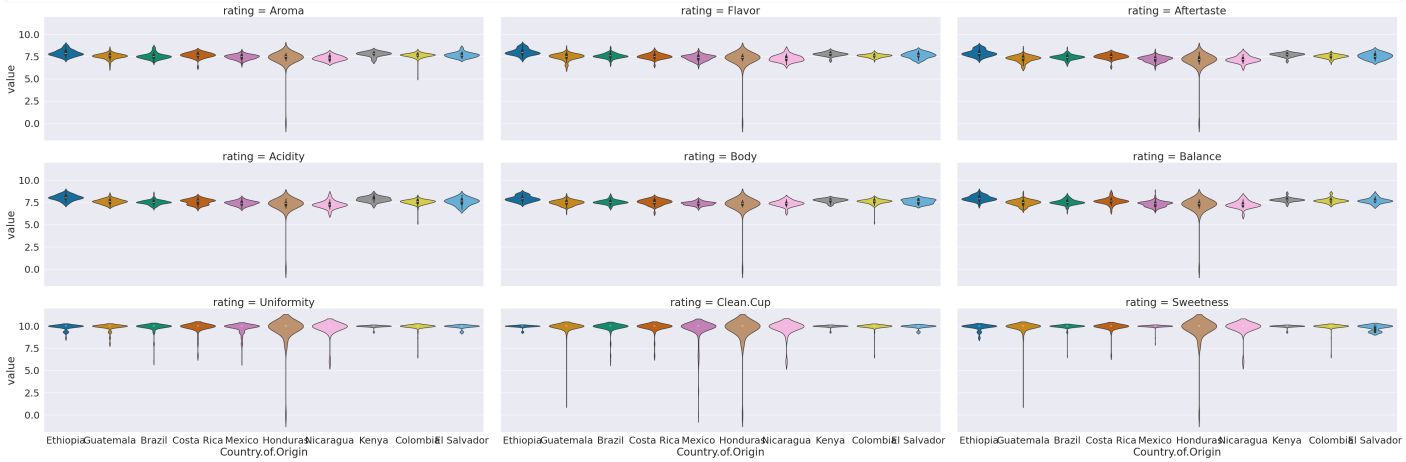
Skip to main content

This data is “tidy” all of the different measurements (ratings) are different rows and we have a few columsn that identify each sample

This version plays much better with plots where we want to compare the ratings.

```
sns.catplot(data = coffee_tall, col='rating', y='value',x='Country.of.Origin',
            hue='Country.of.Origin',aspect=3, col_wrap =3, kind='violin')
```

```
<seaborn.axisgrid.FacetGrid at 0x7f423c5e3370>
```



This now we can use the `rating` as a variable to break the data by in our subplots. I did a few new things in this:

- a violin plot allows us to compare how different data is distributed.
- I used both `col` and `col_wrap`, `col` alone would have made nine columns, and without `row` all in one row. `col_wrap` says to line wrap after a certain number of columns (here 3)
- `aspect` is an attribute of the figure level plots that controls the ratio of width to height (the aspect ratio). so `aspect=3` makes it 3 times as wide as tall in each subplot, this spreads out the x tick labels so can read them

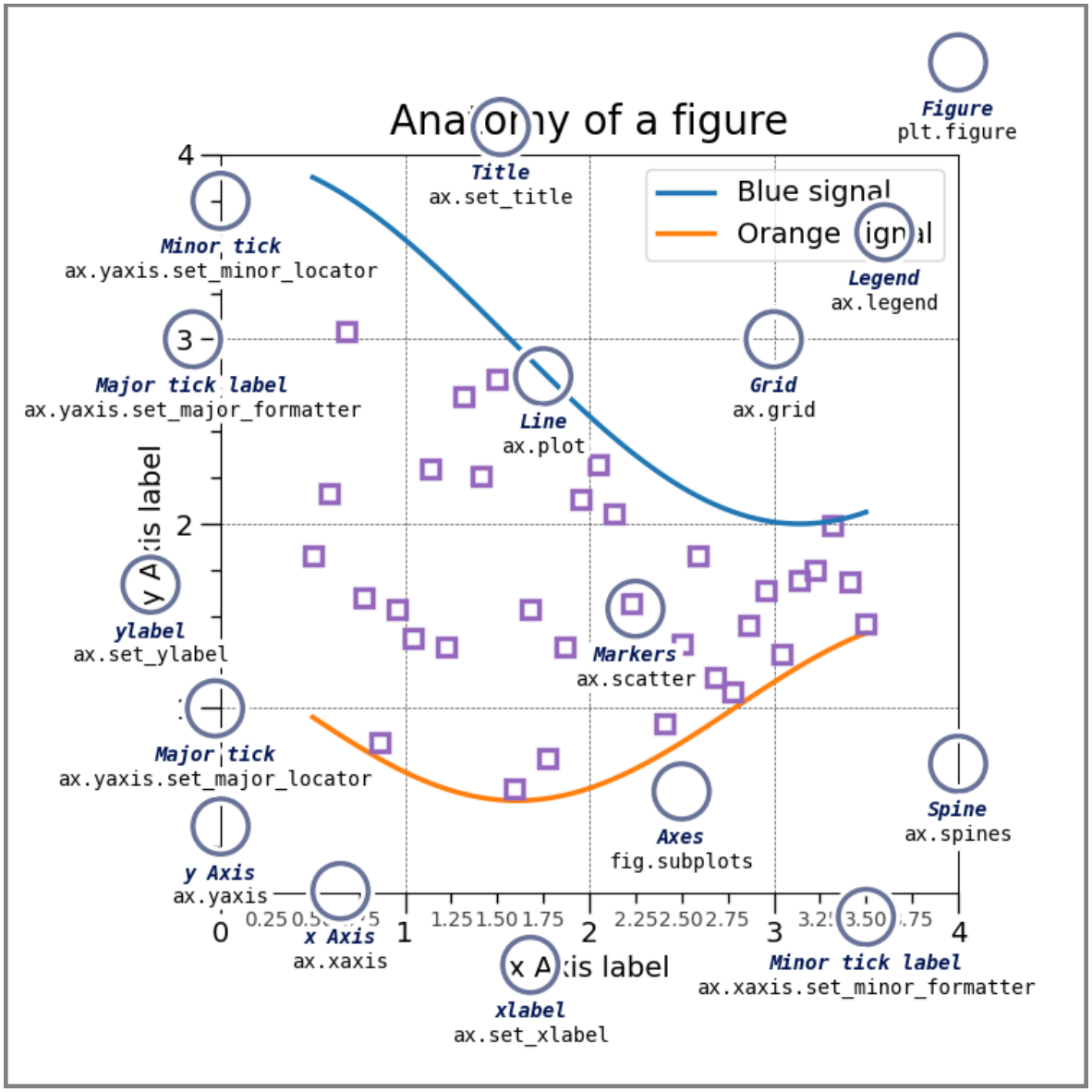
7.1. Questions After Class

Most of the questions today were about assignment 3 or asking to explain specific things that are above. Feel free to post an issue if a question comes up later.

7.1.1. Can you change the locations of the names next to a plot?

First, see the annotated graph and learn the technical name of the element you want to move.

Skip to main content



the above figure come from matplotlib's Anatomy of a Figure page which includes the *code* to generate that figure

You can control each of these but you may need to use `matplotlib`.

If this means the legend, seaborn can control that location. If it refers to fixing overlappign tick labels, I demonstrated that above, with `aspect`.

8. Fixing Values

o far, we've dealt with structural issues in data. but there's a lot more to cleaning.

Today, we'll deal with how to fix the values within the data.

8.1. Cleaning Data review

Instead of more practice with these manipulations, below are more examples of cleaning data to see how these types of

Skip to main content

Your goal here is not to memorize every possible thing, but to build a general idea of what good data looks like and good habits for cleaning data and keeping it reproducible.

- Cleaning the Adult Dataset
- All Shades Also here are some tips on general data management and organization.

This article is a comprehensive discussion of data cleaning.

8.1.1. A Cleaning Data Recipe

not everything possible, but good enough for this course

1. Can you use parameters to read the data in better?
2. Fix the index and column headers (making these easier to use makes the rest easier)
3. Is the data strucutred well?
4. Are there missing values?
5. Do the datatypes match what you expect by looking at the head or a sample?
6. Are categorical variables represented in usable way?
7. Does your analysis require filtering or augmenting the data?

8.2. What is clean enough?

This is a great question, without an easy answer.

It depends on what you want to do. This is why it's important to have potential questions in mind if you are cleaning data for others *and* why we often have to do a little bit more preparation after a dataset has been "cleaned"

Dealing with missing data is a whole research area. There isn't one solution.

in 2020 there was a whole workshop on missing

one organizer is the main developer of sci-kit learn the ML package we will use soon. In a 2020 invited talk he listed more automatic handling as an active area of research and a development goal for sklearn.

There are also many classic approaches both when training and when applying models.

example application in breast cancer detection

Even in pandas, dealing with missing values is under experimentation as to how to represent it symbolically

Missing values even causes the datatypes to change

That said, there are are om Pandas gives a few basic tools:

- dropna
- fillna

- you can approximate with another column
- you can approximate with that column from other rows

Special case, what if we're filling a summary table?

- filling with a symbol for printing can be a good choice, but not for analysis.

whatever you do, document it

```
import pandas as pd
import seaborn as sns
import numpy as np
na_toy_df = pd.DataFrame(data = [[1,3,4,5],[2 ,6, np.nan],[np.nan]*4,[np.nan,3,4,5]], columns=['a','b','

# make plots look nicer and increase font size
sns.set_theme(font_scale=2)

# todays data
arabica_data_url = 'https://raw.githubusercontent.com/jldbc/coffee-quality-database/master/data/arabica_

coffee_df = pd.read_csv(arabica_data_url)
```

8.3. Missing values

We tend to store missing values as `NaN` or use the constants:

```
pd.NA, np.nan
```

```
(<NA>, nan)
```

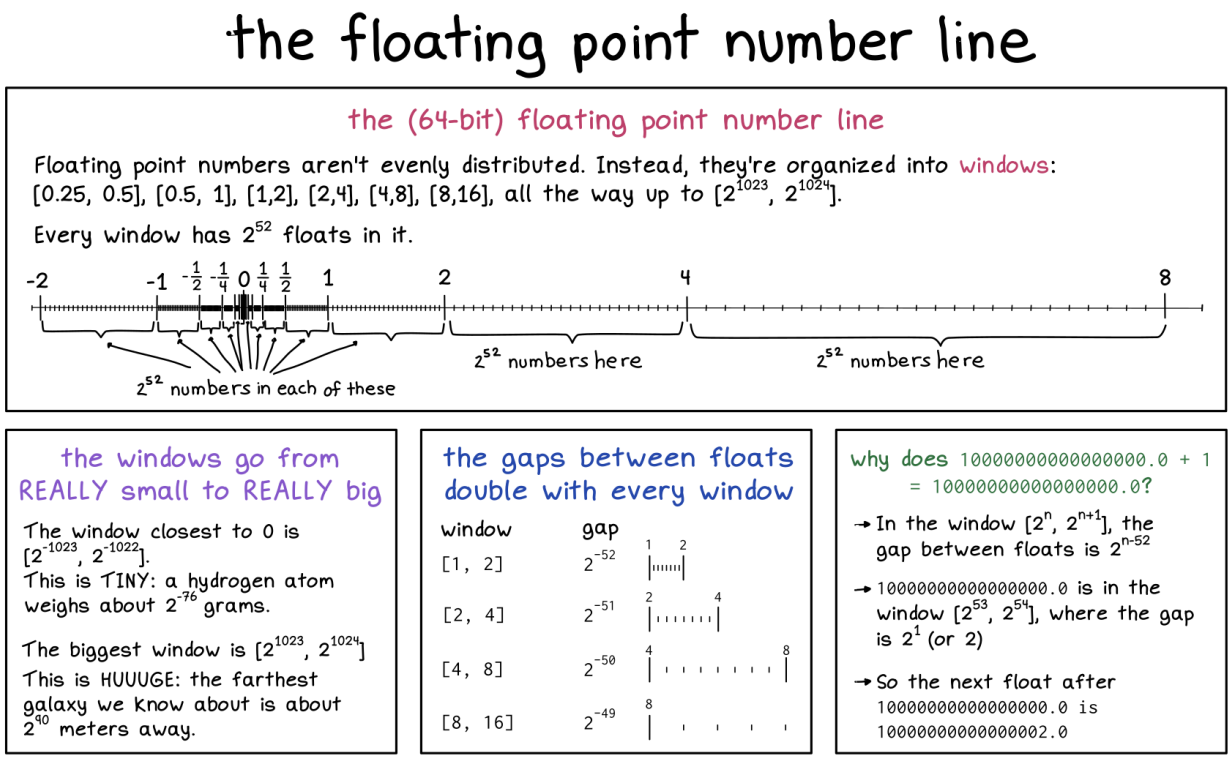
Pandas makes that a special typed object, but converts the whole *column* to float

Numpy uses *float* value for `NaN` that is defined by IEEE floating point standard

```
type(pd.NA),type(np.nan)
```

```
(pandas._libs.missing.NAType, float)
```

Floats are weird



there are values that cannot be represented.

- see float.exposed for infinity
- negative infinity
- see a nan (which bits can be changd without making it *not* nan)
- from 9007199254740992.0 the next closest value is 9007199254740994.0… no values in between can be stores in double precision float

We can see a few in this toy dataset

This data has some missing values

```
na_toy_df
```

	a	b	c	d
0	1.0	3.0	4.0	5.0
1	2.0	6.0	NaN	NaN
2	NaN	NaN	NaN	NaN
3	NaN	3.0	4.0	5.0

Let's try the default behavior of `dropna`

Skip to main content

```
na_toy_df.dropna()
```

	a	b	c	d
0	1.0	3.0	4.0	5.0

This is the same as

```
na_toy_df.dropna(how='any',subset=na_toy_df.columns,axis=0)
```

	a	b	c	d
0	1.0	3.0	4.0	5.0

by default it drops all of the *rows* where **any** of the elements are missing (1 or more) we can change `how` to its other mode:

```
na_toy_df.dropna(how='all')
```

	a	b	c	d
0	1.0	3.0	4.0	5.0
1	2.0	6.0	NaN	NaN
3	NaN	3.0	4.0	5.0

in `'all'` mode it only drops rows where **all** of the values are missing

we can also change it to work along columns (`axis=1`) instead

```
na_toy_df.dropna(how='all',axis=1)
```

	a	b	c	d
0	1.0	3.0	4.0	5.0
1	2.0	6.0	NaN	NaN
2	NaN	NaN	NaN	NaN
3	NaN	3.0	4.0	5.0

None of the columns are *all* missing so nothing is dropped

Let's say we had an analysis where we neded at least one of column `c` or `d` or else we could not use the row, we can check that this way:

```
na_toy_df.dropna(how='all',subset=['c','d']
```

Skip to main content

```
  Cell In[9], line 1
    na_toy_df.dropna(how='all',subset=['c','d']
                                               ^
SyntaxError: unexpected EOF while parsing
```

8.3.1. Filling missing values

Let's look at a real dataset now

```
coffee_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1311 entries, 0 to 1310
Data columns (total 44 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Unnamed: 0             1311 non-null   int64  
 1   Species                1311 non-null   object 
 2   Owner                  1304 non-null   object 
 3   Country.of.Origin      1310 non-null   object 
 4   Farm.Name              955 non-null    object 
 5   Lot.Number             270 non-null    object 
 6   Mill                   1001 non-null   object 
 7   ICO.Number             1163 non-null   object 
 8   Company                1102 non-null   object 
 9   Altitude               1088 non-null   object 
 10  Region                 1254 non-null   object 
 11  Producer               1081 non-null   object 
 12  Number.of.Bags         1311 non-null   int64  
 13  Bag.Weight             1311 non-null   object 
 14  In.Country.Partner     1311 non-null   object 
 15  Harvest.Year           1264 non-null   object 
 16  Grading.Date           1311 non-null   object 
 17  Owner.1                1304 non-null   object 
 18  Variety                1110 non-null   object 
 19  Processing.Method      1159 non-null   object 
 20  Aroma                  1311 non-null   float64
 21  Flavor                 1311 non-null   float64
 22  Aftertaste             1311 non-null   float64
 23  Acidity                1311 non-null   float64
 24  Body                   1311 non-null   float64
 25  Balance                1311 non-null   float64
 26  Uniformity             1311 non-null   float64
 27  Clean.Cup              1311 non-null   float64
 28  Sweetness              1311 non-null   float64
 29  Cupper.Points          1311 non-null   float64
 30  Total.Cup.Points       1311 non-null   float64
 31  Moisture               1311 non-null   float64
 32  Category.One.Defects   1311 non-null   int64  
 33  Quakers                1310 non-null   float64
 34  Color                  1044 non-null   object 
 35  Category.Two.Defects   1311 non-null   int64  
 36  Expiration             1311 non-null   object 
 37  Certification.Body     1311 non-null   object 
 38  Certification.Address  1311 non-null   object 
 39  Certification.Contact  1311 non-null   object 
 40  unit_of_measurement    1311 non-null   object 
 41  altitude_low_meters    1084 non-null   float64
 42  altitude_high_meters   1084 non-null   float64
 43  altitude_mean_meters   1084 non-null   float64
dtypes: float64(16), int64(4), object(24)
memory usage: 450.8+ KB
```

The ‘Lot.Number’ has a lot of NaN values, how can we explore it?

We can look at the type:

```
coffee_df['Lot.Number'].dtype
```

```
dtype('O')
```

Skip to main content

```
coffee_df['Lot.Number'].value_counts()
```

```
Lot.Number
1                             18
020/17                         6
019/17                         5
2                              3
102                            3
                              ..
11/23/0696                     1
3-59-2318                      1
8885                           1
5055                           1
017-053-0211/ 017-053-0212     1
Name: count, Length: 221, dtype: int64
```

We see that a lot are '1', maybe we know that when the data was collected, if the Farm only has one lot, some people recorded '1' and others left it as missing. So we could fill in with 1:

```
coffee_df['Lot.Number'].fillna('1').head()
```

```
0    1
1    1
2    1
3    1
4    1
Name: Lot.Number, dtype: object
```

```
coffee_df['Lot.Number'].head()
```

```
0    NaN
1    NaN
2    NaN
3    NaN
4    NaN
Name: Lot.Number, dtype: object
```

Note that even after we called `fillna` we display it again and the original data is unchanged. To save the filled in column, *technically* we have a few choices:

- use the `inplace` parameter. This doesn't offer performance advantages, but does It still copies the object, but then reassigns the pointer. Its under discussion to deprecate
- write to a new DataFrame
- add a column

we will add a column

```
coffee_df['lot_number_clean'] = coffee_df['Lot.Number'].fillna('1')
```

Skip to main content

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	
0	1	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	

1 rows × 45 columns

```
coffee_df.shape
```

```
(1311, 45)
```

8.4. Dropping

Dropping is a good choice when you otherwise have a lot of data and the data is missing at random.

Dropping can be risky if it's not missing at random. For example, if we saw in the coffee data that one of the scores was missing for all of the rows from one country, or even just missing more often in one country, that could bias our results.

here will will focus on how this impacts how much data we have:

```
coffee_df.dropna().shape
```

```
(130, 45)
```

we lose a lot this way.

We could instead tell it to only drop rows with `NaN` in a subset of the columns.

```
coffee_df.dropna(subset=['altitude_low_meters']).shape
```

```
(1084, 45)
```

Now, it drops any row with one or more `NaN` values in that column.

In the Open Policing Project Data Summary we saw that they made a summary information that showed which variables had at least 70% not missing values. We can similarly choose to keep only variables that have more than a specific threshold of data, using the `thresh` parameter and `axis=1` to drop along columns.

```
n_rows, _ = coffee_df.shape
coffee_df.dropna(thresh=.7*n_rows, axis=1).shape
```

```
(1311, 44)
```

Skip to main content

8.5. Inconsistent values

This was one of the things that many of you anticipated or had observed. A useful way to investigate for this, is to use `value_counts` and sort them alphabetically by the values from the original data, so that similar ones will be consecutive in the list. Once we have the `value_counts()` Series, the values from the `coffee_df` become the index, so we use `sort_index`.

Let's look at the `in_country_partner` column

```
coffee_df['In.Country.Partner'].value_counts().sort_index()
```

```
In.Country.Partner
AMECAFE                                                                205
Africa Fine Coffee Association                                          49
Almacafé                                                               178
Asociacion Nacional Del Café                                           155
Asociación Mexicana De Cafés y Cafeterías De Especialidad A.C.           6
Asociación de Cafés Especiales de Nicaragua                              8
Blossom Valley International                                            58
Blossom Valley International\n                                           1
Brazil Specialty Coffee Association                                     67
Central De Organizaciones Productoras De Café y Cacao Del Perú - Central Café & Cacao      1
Centro Agroecológico del Café A.C.                                       8
Coffee Quality Institute                                                 7
Ethiopia Commodity Exchange                                             18
Instituto Hondureño del Café                                            60
Kenya Coffee Traders Association                                        22
METAD Agricultural Development plc                                      15
NUCOFFEE                                                                36
Salvadoran Coffee Council                                               11
Specialty Coffee Ass                                                     1
```

We can see there's only one `Blossom Valley International\n` but 58 `Blossom Valley International`, the former is likely a typo, especially since `\n` is a special character for a newline. Similarly, with 'Specialty Coffee Ass' and 'Specialty Coffee Association'.

```
partner_corrections = {'Blossom Valley International\n':'Blossom Valley International',
  'Specialty Coffee Ass':'Specialty Coffee Association'}
coffee_df['in_country_partner_clean'] = coffee_df['In.Country.Partner'].replace(
  to_replace=partner_corrections)
coffee_df['in_country_partner_clean'].value_counts().sort_index()
```

Skip to main content

```
in_country_partner_clean
AMECAFE                                                                    205
Africa Fine Coffee Association                                              49
Almacafé                                                                   178
Asociacion Nacional Del Café                                               155
Asociación Mexicana De Cafés y Cafeterías De Especialidad A.C.               6
Asociación de Cafés Especiales de Nicaragua                                  8
Blossom Valley International                                                59
Brazil Specialty Coffee Association                                         67
Central De Organizaciones Productoras De Café y Cacao Del Perú - Central Café & Cacao      1
Centro Agroecológico del Café A.C.                                           8
Coffee Quality Institute                                                     7
Ethiopia Commodity Exchange                                                 18
Instituto Hondureño del Café                                                60
Kenya Coffee Traders Association                                            22
METAD Agricultural Development plc                                          15
NUCOFFEE                                                                    36
Salvadoran Coffee Council                                                   11
Specialty Coffee Association                                               296
Specialty Coffee Association of Costa Rica                                  42
```

8.6. Multiple values in a single column

Let's look at the column about the bag weights

```
coffee_df['Bag.Weight'].head()
```

```
0    60 kg
1    60 kg
2        1
3    60 kg
4    60 kg
Name: Bag.Weight, dtype: object
```

it has both the *value* and the *units* in a single column, which is not what we want.

This would be better in two separate columns

```
bag_df = coffee_df['Bag.Weight'].str.split(' ').apply(pd.Series).rename({0:'bag_weight_clean',
                                                                         1:'bag_weight_unit'},
                                                                        axis=1)
bag_df.head()
```

	bag_weight_clean	bag_weight_unit
0	60	kg
1	60	kg
2	1	NaN
3	60	kg
4	60	kg

Skip to main content

- picks the column
- treats it as a string with the pandas Series attribute `.str`
- uses base python `str.split` to split at `' '` spaces and makes a list
- casts each list to Series with `.apply(pd.Series)`
- renames the resulting columns from being numbered to usable names `rename({0:'bag_weight_clean', 1:'bag_weight_unit'}, axis=1)`

> **Tip**
>
> The `.apply(pd.Series)` works on dictionaries too (anything hte series constructor can take to its `data` parameter) so this is good for json data

The following subsections break down the casting and string methods in more detail

8.6.1. String methods

Python has a powerful string class. There is also an even more powerful `string` module

we only need the base `str` methods most of the time

```
example_str = 'kjksfjds sklfjsdl'
type(example_str)
```

```
str
```

Some helpful ones:

```
example_str.split()
```

```
['kjksfjds', 'sklfjsdl']
```

this gives a list

you can also change the separator

```
'phrases-with-hyphens'.split('-')
```

```
['phrases', 'with', 'hyphens']
```

there are also mehtods for chaning the case and other similar things. **Use these* instead of implementing your own string operations!!

Skip to main content

```
('KJKSFJDS SKLFJSDL', 'Kjksfjds sklfjsdl')
```

8.6.2. Casting Review

If we have a variable that is not the type we want like this:

```
a='5'
```

we can check type

```
type(a)
```

```
str
```

and we can use the name of the type we want, as a function to *cast* it to the new type.

```
int(5)
```

```
5
```

and check

```
type(int(a))
```

```
int
```

8.7. Combining parts of dataframes

```
bag_df.head()
```

	bag_weight_clean	bag_weight_unit
0	60	kg
1	60	kg
2	1	NaN
3	60	kg
4	60	kg

we can pass [illegible] and iterable of pandas objects (here a [illegible] of [illegible]) and it will, by default stack them

Skip to main content

```
pd.concat([coffee_df,bag_df],axis=1)
```

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number
0	1	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015
1	2	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015
2	3	Arabica	grounds for health admin	Guatemala	san marcos barrancas "san cristobal cuch	NaN	NaN	NaN
3	4	Arabica	yidnekachew dabessa	Ethiopia	yidnekachew dabessa coffee plantation	NaN	wolensu	NaN
4	5	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015
...	...	...	...	...	...	...	...	...
1306	1307	Arabica	juan carlos garcia lopez	Mexico	el centenario	NaN	la esperanza, municipio juchique de ferrer, ve...	1104328663
1307	1308	Arabica	myriam kaplan-pasternak	Haiti	200 farms	NaN	coeb koperativ ekselsyo basen (350 members)	NaN
1308	1309	Arabica	exportadora atlantic, s.a.	Nicaragua	finca las marías	017-053-0211/ 017-053-0212	beneficio atlantic condega	017-053-0211/ 017-053-0212
1309	1310	Arabica	juan luis alvarado romero	Guatemala	finca el limon	NaN	beneficio serben	11/853/165
1310	1312	Arabica	bismarck castro	Honduras	los hicaques	103	cigrah s.a de c.v.	13-111-053

1311 rows × 48 columns

8.8. Quantizing a variable

Sometimes a variable is recorded continous, or close (like age in years, technically integers are discrete, but for wide enough range it is not very categorical) but we want to analyze it as if it is categorical

Skip to main content

We can add a new variable that is *calculated* from the original one.

Let's say we want to categorize coffes as small, medium or large batch size based on the quantiles for the `'Number.of.Bags'` column.

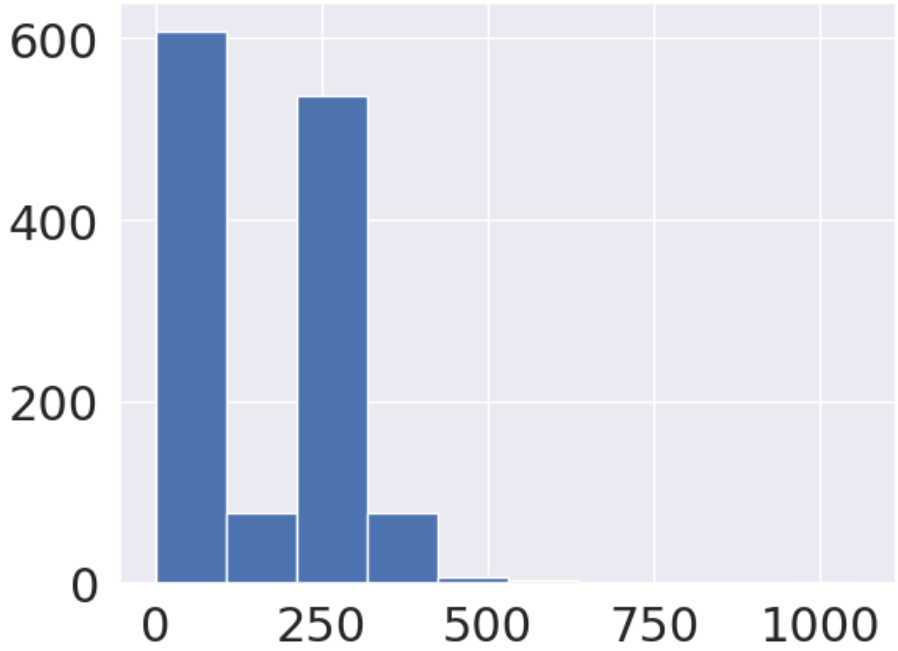
First, we get an idea of the distribution with EDA to make our plan:

```
coffee_df_bags = pd.concat([coffee_df,bag_df],axis=1)
coffee_df_bags['Number.of.Bags'].describe()
```

```
count    1311.000000
mean      153.887872
std       129.733734
min         0.000000
25%        14.500000
50%       175.000000
75%       275.000000
max      1062.000000
Name: Number.of.Bags, dtype: float64
```

```
coffee_df_bags['Number.of.Bags'].hist()
```

```
<Axes: >
```



We see that most are small, but there is at least one major outlier, 75% are below 275, but the max is 1062.

We can use `pd.cut` to make discrete values

```
pd.cut(coffee_df_bags['Number.of.Bags'],bins=3).sample(10)
```

Skip to main content

```
1060    (-1.062, 354.0]
574     (-1.062, 354.0]
422     (-1.062, 354.0]
1244    (-1.062, 354.0]
34      (-1.062, 354.0]
377      (354.0, 708.0]
286     (-1.062, 354.0]
754     (-1.062, 354.0]
403     (-1.062, 354.0]
124     (-1.062, 354.0]
Name: Number.of.Bags, dtype: category
Categories (3, interval[float64, right]): [(-1.062, 354.0] < (354.0, 708.0] < (708.0, 1062.0]]
```

by default, it makes bins of equal size, meaning the range of values. This is not good based on what we noted above. Most will be in one label

```
pd.cut(coffee_df_bags['Number.of.Bags'],bins=3).hist()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[38], line 1
----> 1 pd.cut(coffee_df_bags['Number.of.Bags'],bins=3).hist()

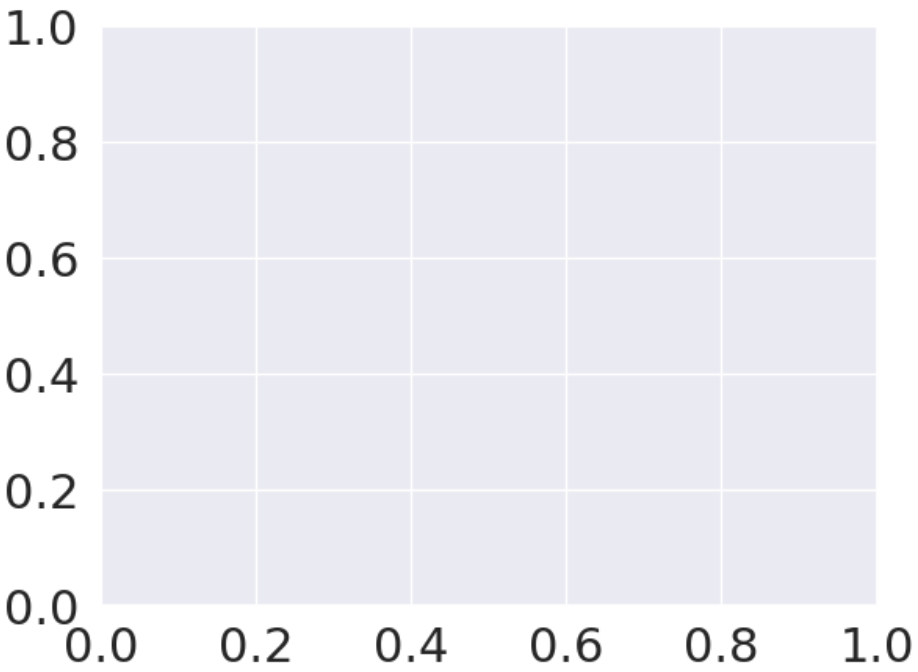
File /opt/hostedtoolcache/Python/3.8.18/x64/lib/python3.8/site-packages/pandas/plotting/_core.py:99, in h
     50 """
     51 Draw histogram of the input series using matplotlib.
     52
   (...)
     96 matplotlib.axes.Axes.hist : Plot a histogram using matplotlib.
     97 """
     98 plot_backend = _get_plot_backend(backend)
---> 99 return plot_backend.hist_series(
    100     self,
    101     by=by,
    102     ax=ax,
    103     grid=grid,
    104     xlabelsize=xlabelsize,
    105     xrot=xrot,
    106     ylabelsize=ylabelsize,
    107     yrot=yrot,
    108     figsize=figsize,
    109     bins=bins,
    110     legend=legend,
    111     **kwargs,
    112 )

File /opt/hostedtoolcache/Python/3.8.18/x64/lib/python3.8/site-packages/pandas/plotting/_matplotlib/hist.
    423 if legend:
    424     kwds["label"] = self.name
--> 425 ax.hist(values, bins=bins, **kwds)
    426 if legend:
    427     ax.legend()

File /opt/hostedtoolcache/Python/3.8.18/x64/lib/python3.8/site-packages/matplotlib/__init__.py:1446, in _
   1443 @functools.wraps(func)
   1444 def inner(ax, *args, data=None, **kwargs):
   1445     if data is None:
-> 1446         return func(ax, *map(sanitize_sequence, args), **kwargs)
   1448     bound = new_sig.bind(ax, *args, **kwargs)
   1449     auto_label = (bound.arguments.get(label_namer)
   1450                   or bound.kwargs.get(label_namer))

File /opt/hostedtoolcache/Python/3.8.18/x64/lib/python3.8/site-packages/matplotlib/axes/_axes.py:6763, in
   6759 for xi in x:
   6760     if len(xi):
   6761         # python's min/max ignore nan,
   6762         # np.minnan returns nan for all nan input
-> 6763         xmin = min(xmin, np.nanmin(xi))
   6764         xmax = max(xmax, np.nanmax(xi))
   6765 if xmin <= xmax:  # Only happens if we have seen a finite value.

TypeError: '<' not supported between instances of 'pandas._libs.interval.Interval' and 'float'
```



TO make it better, we can specify the bin edges instead of only the number

```
min_bags = coffee_df_bags['Number.of.Bags'].min()
sm_cutoff = coffee_df_bags['Number.of.Bags'].quantile(.33)
md_cutoff = coffee_df_bags['Number.of.Bags'].quantile(.66)
max_bags = coffee_df_bags['Number.of.Bags'].max()
pd.cut(coffee_df_bags['Number.of.Bags'],
       bins=[min_bags,sm_cutoff,md_cutoff,max_bags]).head()
```

```
0    (250.0, 1062.0]
1    (250.0, 1062.0]
2        (0.0, 28.0]
3    (250.0, 1062.0]
4    (250.0, 1062.0]
Name: Number.of.Bags, dtype: category
Categories (3, interval[float64, right]): [(0.0, 28.0] < (28.0, 250.0] < (250.0, 1062.0]]
```

here, we made cutoffs individually and pass them as a list to `pd.cut`

This is okay for 3 bins, but if we change our mind, it's a lot of work to make more. Better is to make the bins more programmatically:

```
[coffee_df_bags['Number.of.Bags'].quantile(pct) for pct in np.linspace(0,1,4)]
```

```
[0.0, 29.0, 250.0, 1062.0]
```

`np.linspace` returns a numpyarray of evenly (linearly; there is also logspace) spaced numbers. From the start to the end value for the number you specify. Here we said 4 evenly spaced from 0 to 1.

this is the same as we had before (up to rounding error)

Skip to main content

```
[min_bags,sm_cutoff,md_cutoff,max_bags]
```

```
[0, 28.0, 250.0, 1062]
```

Now we can use these and optionally, change to text labels (which then means we have to update that too if we change the number 4 to another number, but still less work than above)

```
bag_num_bins = [coffee_df_bags['Number.of.Bags'].quantile(pct) for pct in np.linspace(0,1,4)]
pd.cut(coffee_df_bags['Number.of.Bags'],
        bins=bag_num_bins,labels = ['small','medium','large']).head()
```

```
0    large
1    large
2    small
3    large
4    large
Name: Number.of.Bags, dtype: category
Categories (3, object): ['small' < 'medium' < 'large']
```

we could then add this to the dataframe to work with it

8.9. Questions

8.9.1. How can I rename without a dicionary

Really, best practice is a dictionary or function, that is what `rename` uses.

You can assign to the columns attribute, but then you have to provide all of the column names

8.9.2. Why are strings `object`?

it's largely for backwards compatibility

9. Webscraping

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
```

Warning

If it says it cannot load one of the libraries, use pip inside your notebook to install,

```
pip install beautifulsoup4
```

then restart your kernel (Kernel menu, choose restart)

9.1. Getting Data From Websites

We have seen that `read_html` can get content from an actual website, not a data file that is hosted somewhere on the internet, that takes tables on a website and returns a list of DataFrames.

```
pd.read_html('https://rhodyprog4ds.github.io/BrownSpring23/syllabus/achievements.html')
```

▶ Show code cell output

This gives us a list of DataFrames that come from the website. `pandas` gets tables by looking in the html for the site and finding the `<table>` tags.

If we store it in a variable , we can see that

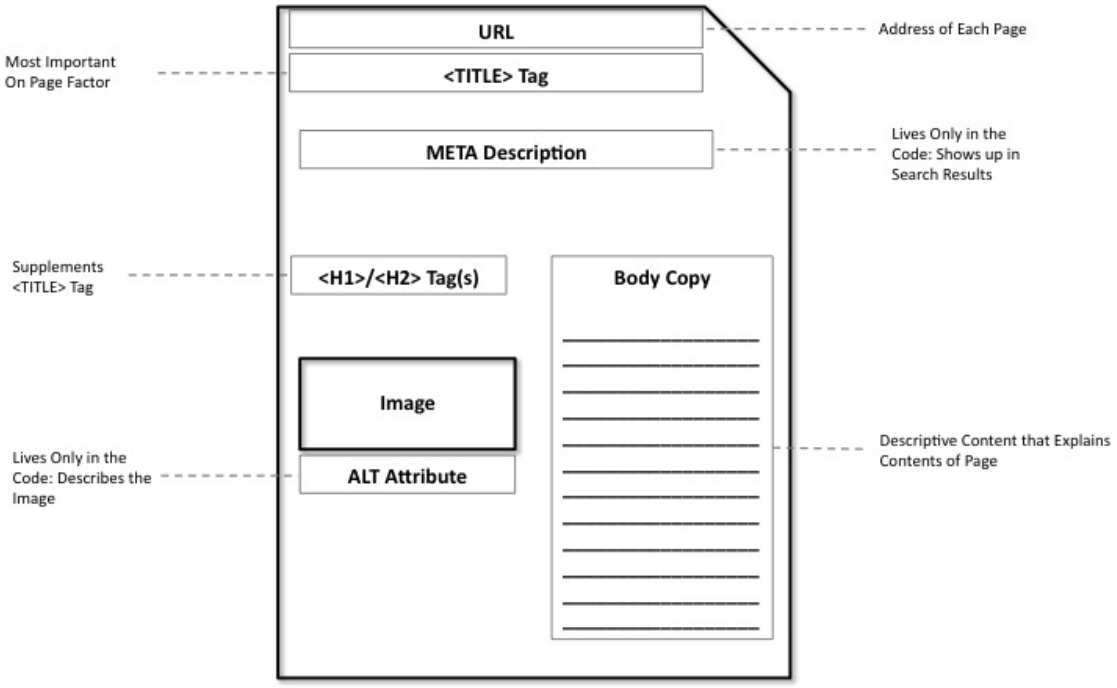
```
df_list = pd.read_html('https://rhodyprog4ds.github.io/BrownFall24/syllabus/achievements.html')
[type(d) for d in df_list]
```

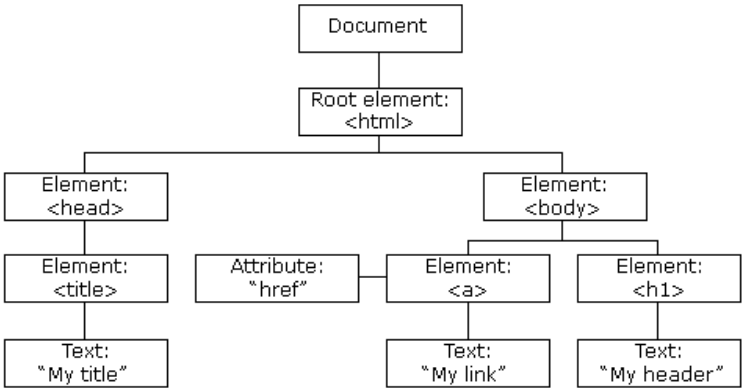
```
[pandas.core.frame.DataFrame,
 pandas.core.frame.DataFrame,
 pandas.core.frame.DataFrame]
```

9.2. Everything is Data

For the purpose of this class, it is best to think of the content on a web page like a datastructure.

Skip to main content





there are tags `<>` that define the structure, and these can be further classified with `classes`

9.3. Scraping a URI website

We're going to create a DataFrame about URI CS & Statistics Faculty.

from the people page of the department website.

We can inspect the page to check that it's well structured.

Skip to main content

Warning

With great power comes great responsibility.

- always check the robots.txt
- do not do things that the owner says not to do
- government websites are typically safe

We'll save the URL for easy use

```
cs_people_url  = 'https://web.uri.edu/cs/people/'
```

Then we can use the `requests` library to make a call to the internet. It actually gets back a response object which has a lot of extra information. For today we only need the `content` from the page which is an attrtibute of that object:

```
cs_people_html = requests.get(cs_people_url).content

cs_people = BeautifulSoup(cs_people_html,'html.parser')
```

This is raw:

```
cs_people_html[:100]
```

```
b'\n<!DOCTYPE html>\n<html lang="en-US">\n\t\n<head>\n<meta charset="UTF-8"><script type="text/javascript
```

But we do not need to manually write search tools, that's what `BeautifulSoup` is for.

```
cs_people
```

```
<!DOCTYPE html>

<html lang="en-US">
<head>
<meta charset="utf-8"/><script type="text/javascript">(window.NREUM||(NREUM={})).init={privacy:{cookies_e
(()=>{var e,t,r={8122:(e,t,r)=>{"use strict";r.d(t,{a:()=>i});var n=r(944);function i(e,t){try{if(!e||"ob
<meta content="width=device-width, initial-scale=1" name="viewport"/>
<link href="http://gmpg.org/xfn/11" rel="profile"/>
<title>People – Department of Computer Science and Statistics</title>
<meta content="max-image-preview:large" name="robots">
<link href="//web.uri.edu" rel="dns-prefetch">
<link href="https://web.uri.edu/cs/feed/" rel="alternate" title="Department of Computer Science and Stati
<link href="https://web.uri.edu/cs/comments/feed/" rel="alternate" title="Department of Computer Science
<script type="text/javascript">
/* <![CDATA[ */
window._wpemojiSettings = {"baseUrl":"https:\/\/s.w.org\/images\/core\/emoji\/14.0.0\/72x72\/","ext":".pr
/*! This file is auto-generated */
!function(i,n){var o,s,e;function c(e){try{var t={supportTests:e,timestamp:(new Date).valueOf()};sessionS
/* ]]> */
</script>
```

Skip to main content

```
type(cs_people)
```

```
bs4.BeautifulSoup
```

9.3.1. Looking at tags

In this object we can use any tag from the file and get back the first instance

the `<a>` tag in HTML makes a link

```
cs_people.a
```

```
<a class="skip-link screen-reader-text" href="#content">Skip to content</a>
```

> **Tip**
>
> This content is a best practice for accessible web design

We also see `<h3>` in the code so we can get the first one like this:

```
cs_people.h3
```

```
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/gavino-puggioni/">Gavino Puggioni</a></h3>
```

this cheatsheet shows lots of html tags, but for this purpose you do not really need it. You'll be inspecting the page and then looking for what you want

9.3.2. Searching the source

More helpful is the `find_all` method we want to find all `div` tags that are "peopleitem" class. We decided this by inspecting the code on the website.

```
type(cs_people.find_all('div','peopleitem'))
```

```
bs4.element.ResultSet
```

```
cs_people.find_all('div','peopleitem')
```

```
[<div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>
 <a href="https://web.uri.edu/cs/meet/gavino-puggioni/"><img alt="" class="u-photo wp-post-image" decodir
 </figure>
 <h3 class="p-name"><a href="https://web.uri.edu/cs/meet/gavino-puggioni/">Gavino Puggioni</a></h3>
 </div>
 </header>
 <div class="inside">
 <p class="people-title p-job-title">Associate Professor |  Chair</p>
 <p class="people-department">Statistics</p>
 <p class="people-misc"><span class="p-tel">401.874.4388</span> <br/> <a class="u-email" href="mailto:gpu
 <div style="clear:both;"></div>
 </div>
 </div>,
 <div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>
```

this is a long, object and we can see it looks iterable (`[` at the start)

```
people_items = cs_people.find_all('div','peopleitem')
```

We can look at a single item

```
people_items[0]
```

```
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>
<a href="https://web.uri.edu/cs/meet/gavino-puggioni/"><img alt="" class="u-photo wp-post-image" decoding
</figure>
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/gavino-puggioni/">Gavino Puggioni</a></h3>
</div>
</header>
<div class="inside">
<p class="people-title p-job-title">Associate Professor |  Chair</p>
<p class="people-department">Statistics</p>
<p class="people-misc"><span class="p-tel">401.874.4388</span> <br/> <a class="u-email" href="mailto:gpug
<div style="clear:both;"></div>
</div>
</div>
```

and how many it finds

```
len(people_items)
```

```
29
```

Important

answer to questions about searching from the docs

We notice that the name is inside a `<h3>` tag with class `p-name` and then inside an a tag after that. We also know from looking at the overall page that there are lots of other a tags, so we do not want to search all of those.

```
people_items[0].find('h3','p-name')
```

```
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/gavino-puggioni/">Gavino Puggioni</a></h3>
```

Then we see in this, there is an `<a>` tag, so we can pull that out next, we can use the tag attribute, because the first instance of the tag is exactly what we want.

```
people_items[0].find('h3','p-name').a
```

```
<a href="https://web.uri.edu/cs/meet/gavino-puggioni/">Gavino Puggioni</a>
```

inside that there is the text in a string, so we can pull that out

```
people_items[0].find('h3','p-name').a.string
```

```
'Gavino Puggioni'
```

Finally, now that we know how to get one out, we can put it all in a list comprehension

```
names = [person.find('h3','p-name').a.string for person in people_items]
```

pull out the titles for each person and store in a variable `titles`

9.4. Pulling more information

First, we look at the whole person entry again.

```
people_items[0]
```

Skip to main content

```
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>
<a href="https://web.uri.edu/cs/meet/gavino-puggioni/"><img alt="" class="u-photo wp-post-image" decoding
</figure>
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/gavino-puggioni/">Gavino Puggioni</a></h3>
</div>
</header>
<div class="inside">
<p class="people-title p-job-title">Associate Professor |  Chair</p>
<p class="people-department">Statistics</p>
<p class="people-misc"><span class="p-tel">401.874.4388</span> <br/> <a class="u-email" href="mailto:gpug
<div style="clear:both;"></div>
</div>
</div>
```

on one item, the `p` tag with the `people-title` class gets us what we want and then we can

```
title = [person.find('p','people-title').string for person in people_items]
```

We can also use the whole page object, but in some cases it might be safer or faster to only work with the chunk we had separate (like above with `people_items`). There are many ways and all are valid, but it is worth thinking about pros and cons. If there were any of the thing we were searching for in a different part that we did not want here, then the subset would be more accurate.

We can pull out two more things, the people-department indicates who is CS & who is Statistics.

```
disciplines = [d.string for d in cs_people.find_all("p",'people-department')]
emails = [e.string for e in cs_people.find_all("a",'u-email')]
```

We can finally use the DataFrame constructor to make it a table. I chose to use a dictionary in class

```
css_df = pd.DataFrame({'name':names,'title':title,
             'discipline':disciplines, 'email':emails})
css_df.head()
```

	name	title	discipline	email
0	Gavino Puggioni	Associate Professor | Chair	Statistics	gpuggioni@uri.edu
1	Lisa DiPippo	Professor | Director of Undergraduate Studies	Computer Science	ldipippo@uri.edu
2	Natallia Katenka	Associate Professor | Director of Data Science	Statistics	nkatenka@uri.edu
3	Krishna Venkatasubramanian	Associate Professor | Director of Graduate Stu...	Computer Science	krish@uri.edu
4	Jing Wu	Associate Professor | Director of Graduate Stu...	Statistics	jing_wu@uri.edu

We could also use a list of list and separate list of column names.

```
css_df_b = pd.DataFrame(data=[names,titles,emails,disciplines],
                        columns =['names','titles','emails','disciplines'])
css_df_b
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[24], line 1
----> 1 css_df_b = pd.DataFrame(data=[names,titles,emails,disciplines],
      2                         columns =['names','titles','emails','disciplines'])
      3 css_df_b

NameError: name 'titles' is not defined
```

9.5. Crawling and scraping

Remember we pulled the names out of links, when in the browser, we click on the links, we see that they are to a profile page. On these pages, they have the office number. Let's add those to our dataframe.

First, we will do it for one person, then make a loop.

```
people_items[0].find('h3','p-name').a
```

```
<a href="https://web.uri.edu/cs/meet/gavino-puggioni/">Gavino Puggioni</a>
```

We see that the information that we want is in the `href` attribute, to read that, we check the documentation. This tells us there is a `.attrs` attribute of the python object we are working with.

```
people_items[0].find('h3','p-name').a.attrs
```

```
{'href': 'https://web.uri.edu/cs/meet/gavino-puggioni/'}
```

the classes are allso attributes in HTML so we could access that informaiton this way:

```
people_items[0].find('h3','p-name').attrs
```

```
{'class': ['p-name']}
```

It's a dictionary and the attribute we want is the key we want.

```
chair_url = people_items[0].find('h3','p-name').a.attrs['href']
chair_url
```

```
'https://web.uri.edu/cs/meet/gavino-puggioni/'
```

Skip to main content

Now, we do the same thing we did above, request, pull the content from the response and then use the parser.

```
chair_html = requests.get(chair_url).content
chair_info = BeautifulSoup(chair_html,'html.parser')
```

then we find the tag and class we need from inspecting and pull that.

```
chair_info.find('li','people-location')
```

```
<li class="people-location"><strong>Office Location:</strong> Tyler Hall 254</li>
```

This structure is different and we tried string without success

Here, we could go to the documentation and look up what the object contains, or, we can use object serialization.
We can use the python `__dict__` to inspect the object and see where it stored what we want.

```
chair_info.find('li','people-location').__dict__
```

```
{'parser_class': bs4.BeautifulSoup,
 'name': 'li',
 'namespace': None,
 '_namespaces': {},
 'prefix': None,
 'sourceline': 408,
 'sourcepos': 303,
 'known_xml': False,
 'attrs': {'class': ['people-location']},
 'contents': [<strong>Office Location:</strong>, ' Tyler Hall 254'],
 'parent': <ul class="people-list">
<li class="people-title">Associate Professor |  Chair</li> <li class="people-department">Statistics</li>
</ul>,
 'previous_element': ' ',
 'next_element': <strong>Office Location:</strong>,
 'next_sibling': '\n',
 'previous_sibling': ' ',
 'hidden': False,
 'can_be_empty_element': False,
 'cdata_list_attributes': {'*': ['class', 'accesskey', 'dropzone'],
  'a': ['rel', 'rev'],
  'link': ['rel', 'rev'],
  'td': ['headers'],
  'th': ['headers'],
  'form': ['accept-charset'],
  'object': ['archive'],
  'area': ['rel'],
  'icon': ['sizes'],
  'iframe': ['sandbox'],
  'output': ['for']},
 'preserve_whitespace_tags': {'pre', 'textarea'},
 'interesting_string_types': (bs4.element.NavigableString, bs4.element.CData)}
```

we see its the second element in a list in the `'content'` value

```
chair_info.find('li','people-location').contents
```

Skip to main content

```
[<strong>Office Location:</strong>, ' Tyler Hall 254']
```

so we can pull it out

```
chair_info.find('li','people-location').contents[1]
```

```
' Tyler Hall 254'
```

Now tht we know how to do it, we can put it in a loop.

```
offices = []
for person in people_items:
    person_url = person.find('h3','p-name').a.attrs['href']

    person_html = requests.get(person_url).content
    person_info = BeautifulSoup(person_html,'html.parser')
    try:
        office_loc = person_info.find('li','people-location').contents[1]
        offices.append(office_loc)
    except:
        offices.append(pd.NA)
```

We added the `try` and `except` to handle when there is no office location. This is something in practice you would often think to do due an error.

Next we add the offices to our dataframe:

```
css_df['offices'] = offices
```

and peak at the head

```
css_df.head()
```

	name	title	discipline	email	offices
0	Gavino Puggioni	Associate Professor | Chair	Statistics	gpuggioni@uri.edu	Tyler Hall 254
1	Lisa DiPippo	Professor | Director of Undergraduate Studies	Computer Science	ldipippo@uri.edu	Tyler 246
2	Natallia Katenka	Associate Professor | Director of Data Science	Statistics	nkatenka@uri.edu	Tyler 247
3	Krishna Venkatasubramanian	Associate Professor | Director of Graduate Stu...	Computer Science	krish@uri.edu	Tyler 131
4	Jing Wu	Associate Professor | Director of Graduate Stu...	Statistics	jing_wu@uri.edu	Tyler 260

Here we check the `info()` and we can see many were missing.

Skip to main content

```
css_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29 entries, 0 to 28
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   name        29 non-null     object
 1   title       29 non-null     object
 2   discipline  29 non-null     object
 3   email       29 non-null     object
 4   offices     26 non-null     object
dtypes: object(5)
memory usage: 1.3+ KB
```

9.6. Questions

> **Note**
>
> Some of these are questions asked in previous semesters

9.6.1. what does .a do?

it gives the first instance of the `<a>` tag

9.6.2. is it worth it to try and web scrape a page that is poorly written?

If it is important information. In these cases, you might have to do more manual parsing or even some manual fixes.

For this class, no.

9.6.3. Is there a way to check robots.txt through BeautifulSoup or must that be done manually in a browser?

it could maybe be read programmaticlaly, but it doesn't necessarily save time to do it that way.

9.6.4. What else can I do with inspect?

It lets you view the code. It's most often used to debug websites.

9.6.5. when web scraping if the html is not set up well is it possible to change the html to make it easier to parse

Technically you could manually edit a copy of it.

Skip to main content

9.6.6. Are there instances where you can get data from websites that are not in tabular form?

Web scraping is *for* when the website is not in tabular form. It should be strucutred, but the structure does not need to come from a single page. It could be that there are many pages strucutred similarly and you build most of the columns from the other pages, not the starting page.

For example from the teams page of the nba you can get to a page with info about each team that includes all time records and the current rosters. On these individual pages, most info is an actual table, so you can use `pd.read_html` for those, but the crawing part from the first page would count.

10. Merging Data & Databases

10.1. Merging Data

```
import pandas as pd
import sqlite3
from urllib import request
```

we're going to work with a set of datasets today that are stored in a repo.

```
course_data_url = 'https://raw.githubusercontent.com/rhodyprog4ds/rhodyds/main/data/'
```

We can load in two data sets of NBA player information.

```
df_18 = pd.read_csv(course_data_url+'2018-players.csv')
df_19 = pd.read_csv(course_data_url+'2019-players.csv')
```

and take a peek at each

```
df_18.head()
```

	TEAM_ID	PLAYER_ID	SEASON
0	1610612761	202695	2018
1	1610612761	1627783	2018
2	1610612761	201188	2018
3	1610612761	201980	2018
4	1610612761	200768	2018

```
df_19.head()
```

Skip to main content

	PLAYER_NAME	TEAM_ID	PLAYER_ID	SEASON
0	Royce O'Neale	1610612762	1626220	2019
1	Bojan Bogdanovic	1610612762	202711	2019
2	Rudy Gobert	1610612762	203497	2019
3	Donovan Mitchell	1610612762	1628378	2019
4	Mike Conley	1610612762	201144	2019

Important

Remember `shape` is a property, not a method, so it does not need `()`

Let's make note of the shape of each

```
df_18.shape, df_19.shape
```

```
((748, 3), (626, 4))
```

this created a tuple

10.1.1. What if we want to analyze them together?

We can stack them, but this does not make it easy to see , for example, who changed teams.

```
pd.concat([df_18,df_19])
```

	TEAM_ID	PLAYER_ID	SEASON	PLAYER_NAME
0	1610612761	202695	2018	NaN
1	1610612761	1627783	2018	NaN
2	1610612761	201188	2018	NaN
3	1610612761	201980	2018	NaN
4	1610612761	200768	2018	NaN
...	...	...	...	...
621	1610612745	203461	2019	Anthony Bennett
622	1610612737	1629034	2019	Ray Spalding
623	1610612744	203906	2019	Devyn Marble
624	1610612753	1629755	2019	Hassani Gravett
625	1610612754	1629721	2019	JaKeenan Gant

1374 rows × 4 columns

Skip to main content

```
pd.concat([df_18,df_19]).shape
```

```
(1374, 4)
```

we can see that this is the total number of rows:

```
748+626
```

```
1374
```

Note that this has the maximum number of columns (because both had some overlapping columns) and the total number of rows.

10.1.2. How can we find which players changed teams?

To do this we want to have one player column and a column with each year's team.

We can use a merge to do that.

```
pd.merge(df_18,df_19).head(2)
```

TEAM_ID	PLAYER_ID	SEASON	PLAYER_NAME

if we merge them without any parameters, it tries to merge on all shared columns. We want to merge them using the `PLAYER_ID` column though, we would say that we are "merging on player ID" and we use the `on` parameter to do it. In this case, it looks for the values in the `PLAYER_ID` column that appear in both DataFrames and combines them into a single row.

```
pd.merge(df_18,df_19,on='PLAYER_ID',suffixes=('_18','_19')).head(2)
```

	TEAM_ID_18	PLAYER_ID	SEASON_18	PLAYER_NAME	TEAM_ID_19	SEASON_19
0	1610612761	202695	2018	Kawhi Leonard	1610612746	2019
1	1610612761	1627783	2018	Pascal Siakam	1610612761	2019

Since there are other columns that appear in both DataFrames, they get a suffix, which by default is `x` or `y`, we can specify them though.

```
pd.merge(df_18,df_19,on='PLAYER_ID',suffixes=('_18','_19')).shape
```

```
(538, 6)
```

Skip to main content

10.1.3. Which players played in 2018, but not 2019?

We have different types of merges, inner is both, out is either. Left and right keep all the rows of one dataFrame. We can use left with `df_18` as the left DataFrame to see which players played only in 18.

```
pd.merge(df_18,df_19,on='PLAYER_ID',how='left',suffixes=('_18','_19')).shape
```

```
(754, 6)
```

```
pd.merge(df_18,df_19,on='PLAYER_ID',how='left',suffixes=('_18','_19')).info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 754 entries, 0 to 753
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   TEAM_ID_18   754 non-null    int64  
 1   PLAYER_ID    754 non-null    int64  
 2   SEASON_18    754 non-null    int64  
 3   PLAYER_NAME  538 non-null    object 
 4   TEAM_ID_19   538 non-null    float64
 5   SEASON_19    538 non-null    float64
dtypes: float64(2), int64(3), object(1)
memory usage: 35.5+ KB
```

IF we save this to a variable, we can answer our question

```
df_18_only = pd.merge(df_18,df_19,on='PLAYER_ID',suffixes=('_18','_19'),how='left')
df_18_only.head(2)
```

	TEAM_ID_18	PLAYER_ID	SEASON_18	PLAYER_NAME	TEAM_ID_19	SEASON_19
0	1610612761	202695	2018	Kawhi Leonard	1.610613e+09	2019.0
1	1610612761	1627783	2018	Pascal Siakam	1.610613e+09	2019.0

```
len(df_18_only[df_18_only['TEAM_ID_19'].isna()]['PLAYER_ID'].unique())
```

```
178
```

Also, note that this has different types than before. There are some players who only played one season, so they have a NaN value in some colums. pandas always casts a whole column.

Skip to main content

10.2. Getting Data from Databases

10.2.1. What is a Database?

A common attitude in Data Science is:

> If your data fits in memory there is no advantage to putting it in a database: it will only be slower and more frustrating. — Hadley Wickham

Businesses and research organizations nearly always have too much data to feasibly work without a database. Instead, they use different tools which are designed to scale to very large amounts of data. These tools are largely databases like Snowflake or Google's BigQuery and distributed computing frameworks like Apache Spark.

> ⚠ **Warning**
>
> We are going to focus on the case of getting data out of a Database so that you can use it and making sure you know what a Database is.
>
> You could spend a whole semester on databases:
>
> - CSC436 covers how to implement them in detail (recommended, but requires CSC212)
> - BAI456 only how to use them (counts for DS majors, but if you want to understand them deeper, the CSC one is recommended)

For the purpose of this class the key attributes of a database are:

- it is a collection of tables
- the data is accessed live from disk (not RAM)
- you send a query to the database to get the data (or your answer)

Databases can be designed in many different ways. For examples two popular ones.

- SQLite is optimized for transactional workloads, which means a high volume of requests that involving inserting or reading a couple things. This is good for eg a webserver.
- DuckDB is optimized for analytical workloads, which means a small number of requests that each require reading many records in the database. This is better for eg: data science

Experimenting with DuckDB is a way to earn construct level 3

10.2.2. Accessing a Database from Python

We will use pandas again, as well as the `request` module from the `urllib` package and `sqlite3`.

Off the shelf, pandas cannot read databased by default. We'll use the `sqlite3` library, but there are others, depending on the

Skip to main content

First we need to download the database to work with it.

```
request.urlretrieve('https://github.com/rhodyprog4ds/rhodyds/raw/main/data/nba1819.db',
      'nba1819.db')
```

```
('nba1819.db', <http.client.HTTPMessage at 0x7f98c0985c40>)
```

Next, we set up a connection, that links the the notebook to the database. To use it, we add a cursor.

```
conn = sqlite3.connect('nba1819.db')
cursor = conn.cursor()
```

We can use execute to pass SQL queries through the cursor to the database.

```
cursor.execute("SELECT name FROM sqlite_master WHERE type='table';")
```

```
<sqlite3.Cursor at 0x7f98c09979d0>
```

Then we use `fetchall` to get the the results of the query.

```
cursor.fetchall()
```

```
[('teams',),
 ('conferences',),
 ('playerGameStats2018',),
 ('playerGameStats2019',),
 ('teamGameStats2018',),
 ('teamGameStats2019',),
 ('playerTeams2018',),
 ('playerTeams2019',),
 ('teamDailyRankings2018',),
 ('teamDailyRankings2019',),
 ('playerNames',)]
```

If we fetch again, there is nothing to fetch. Fetch pulls what was queued by execute.

```
cursor.fetchall()
```

```
[]
```

```
pd.read_sql("SELECT name FROM sqlite_master WHERE type='table';",conn)
```

Skip to main content

	name
0	teams
1	conferences
2	playerGameStats2018
3	playerGameStats2019
4	teamGameStats2018
5	teamGameStats2019
6	playerTeams2018
7	playerTeams2019
8	teamDailyRankings2018
9	teamDailyRankings2019
10	playerNames

10.3. Querying with pandas

We can use `pd.read_sql` to send queries, get the result sand transform them into a DataFrame all at once

We can pass the exact same queries if we want.

```
pd.read_sql("SELECT name FROM sqlite_master WHERE type='table';",conn)
```

	name
0	teams
1	conferences
2	playerGameStats2018
3	playerGameStats2019
4	teamGameStats2018
5	teamGameStats2019
6	playerTeams2018
7	playerTeams2019
8	teamDailyRankings2018
9	teamDailyRankings2019
10	playerNames

or we can get all of one of the tables:

```
pd.read_sql('SELECT * FROM teams',conn).head(1)
```

Skip to main content

	index	LEAGUE_ID	TEAM_ID	MIN_YEAR	MAX_YEAR	ABBREVIATION	NICKNAME	YEARFOUNDED	
0	0	0	1610612737	1949	2019	ATL	Hawks	1949	A

10.3.1. Which player was traded the most during the 2018 season? How many times?

There is one row in players per team a played for per season, so if a player was traded (changed teams), they are in there multiple times.

First, we'll check the column names

```
pd.read_sql("SELECT * FROM playerTeams2018 LIMIT 1",conn)
```

	index	TEAM_ID	PLAYER_ID
0	0	1610612761	202695

then get the 2018 players, we only need the `PLAYER_ID` column for this question

```
p18 =pd.read_sql("SELECT PLAYER_ID FROM playerTeams2018 ",conn)
```

Then we can use value counts

```
p18.value_counts().sort_values(ascending=False).head(10)
```

```
PLAYER_ID
1629150      4
202325       3
203092       3
201160       3
202328       3
1626150      3
1628393      3
202083       3
202692       3
203477       3
Name: count, dtype: int64
```

and we can get the player's name from the player name **remember our first query told us all the tables**

```
pd.read_sql("SELECT PLAYER_NAME FROM playerNames WHERE PLAYER_ID = 1629150",conn)
```

	PLAYER_NAME
0	Emanuel Terry

Skip to main content

10.3.2. Did more players who changed teams from the 2018 season to the 2019 season stay in the same conferences or switch conferences?

In the NBA, there are 30 teams organized into two conferences: East and West; the `conferences` table has the columns `TEAM_ID` and `CONFERENCE`

Let's build a Dataframe that could answer the question.

I first pulled 1 row from each table I needed to see the columns.

```
pd.read_sql('SELECT * FROM conferences LIMIT 1',conn)
```

	index	TEAM_ID	CONFERENCE
0	0	1610612744	West

```
pd.read_sql('SELECT * FROM playerTeams2018 LIMIT 1',conn)
```

	index	TEAM_ID	PLAYER_ID
0	0	1610612761	202695

```
pd.read_sql('SELECT * FROM playerTeams2019 LIMIT 1',conn)
```

	index	TEAM_ID	PLAYER_ID
0	0	1610612762	1626220

Then I pulled the columns I needed from each of the 3 tables into a separate DataFrame.

```
conf_df = pd.read_sql('SELECT TEAM_ID,CONFERENCE FROM conferences',conn)
df18 = pd.read_sql('SELECT TEAM_ID,PLAYER_ID FROM playerTeams2018',conn)
df19 = pd.read_sql('SELECT TEAM_ID,PLAYER_ID FROM playerTeams2019',conn)
df18_c = pd.merge(df18,conf_df,on='TEAM_ID')
df19_c = pd.merge(df19,conf_df,on='TEAM_ID')
df1819_conf = pd.merge(df18_c,df19_c, on='PLAYER_ID',suffixes=('_2018','_2019'))
df1819_conf
```

Skip to main content

	TEAM_ID_2018	PLAYER_ID	CONFERENCE_2018	TEAM_ID_2019	CONFERENCE_2019
0	1610612761	202695	East	1610612746	West
1	1610612761	1627783	East	1610612761	East
2	1610612761	201188	East	1610612761	East
3	1610612763	201188	West	1610612761	East
4	1610612761	201980	East	1610612747	West
...	...	...	...	...	...
533	1610612739	1628021	East	1610612751	East
534	1610612739	201567	East	1610612739	East
535	1610612739	202684	East	1610612739	East
536	1610612739	1628424	East	1610612766	East
537	1610612739	1627819	East	1610612761	East

538 rows × 5 columns

Then I merged the conference with each set of player informationon the teams. Then I merged the two expanded single year DataFrames together.

Now, to answer the question, we have a bit more work to do. I'm going to use a `lambda` and `apply` to make a column that says same or new for the relative conference of the two seasons.

```
labels = {False:'new',True:'same'}
change_conf = lambda row: labels[row['CONFERENCE_2018']==row['CONFERENCE_2019']]
df1819_conf['conference_1819']= df1819_conf.apply(change_conf,axis=1)
df1819_conf.head()
```

	TEAM_ID_2018	PLAYER_ID	CONFERENCE_2018	TEAM_ID_2019	CONFERENCE_2019	conference_1819
0	1610612761	202695	East	1610612746	West	new
1	1610612761	1627783	East	1610612761	East	same
2	1610612761	201188	East	1610612761	East	same
3	1610612763	201188	West	1610612761	East	new
4	1610612761	201980	East	1610612747	West	new

Then I can use this DataFrame grouped by my new column to get the unique players in each situation new or same conference.

```
df1819_conf.groupby('conference_1819')['PLAYER_ID'].apply(pd.unique)
```

```
conference_1819
new     [202695, 201188, 201980, 203961, 1626153, 1011...
same    [1627783, 201188, 200768, 1627832, 201586, 162...
Name: PLAYER_ID, dtype: object
```

Skip to main content

And finally, get the length of each of those lists.

```
df1819_conf.groupby('conference_1819')['PLAYER_ID'].apply(pd.unique).apply(len)
```

```
conference_1819
new     119
same    385
Name: PLAYER_ID, dtype: int64
```

This, however, includes players who stayed on the same team, so we also need to split for who changed teams. First we add the team comparison column, then groupby by both and count unique players.

```
new_team = lambda row: labels[row['TEAM_ID_2018']==row['TEAM_ID_2019']]
df1819_conf['team_1819']= df1819_conf.apply(new_team,axis=1)
df1819_conf.groupby(['conference_1819','team_1819'])['PLAYER_ID'].apply(pd.unique).apply(len)
```

```
conference_1819  team_1819
new              new          119
same             new          135
                 same         263
Name: PLAYER_ID, dtype: int64
```

This is good, we could read the answer from here. It's good practice, though, to be able to pull that value out programmatically.

```
player_counts_1819_team =  df1819_conf.groupby(['conference_1819','team_1819'])['PLAYER_ID'].apply(pd.un
player_counts_1819_team.idxmax()
```

```
('same', 'same')
```

This tells us that the largest number of players stayed on the same team (and therefore same conference). We're not interested inthis thoug, we're itnerested in those that changed teams, so we can drop the `(same,same)` value and then do this again.

```
player_counts_1819_team.drop(('same','same')).idxmax()
```

```
('same', 'new')
```

This tells us that more players changed teams within the same conference than changed teams and conferences. We can compare the two directly:

```
player_counts_1819_team['new','new'], player_counts_1819_team['same','new']
```

```
(119, 135)
```

Skip to main content

We can also make this a little neater to print it as a DataFrame. If we use `reset_index` it will make a DataFrame, but the count column will still be named `PLAYER_ID` so we can rename it.

```
player_counts_1819_team.reset_index().rename(columns={'PLAYER_ID':'num_players'})
```

	conference_1819	team_1819	num_players
0	new	new	119
1	same	new	135
2	same	same	263

All in all, this gives us a good answer that we can get with data and display answers and this is one way that using multiple data sources can help answer richer questions.

```
conn.close()
```

1. Assignment 1: Setup, Syllabus, and Review

Due: 2024-09-10 2:00pm

> **Warning**
>
> You must *complete* it by 2pm on Tuesday, but if you are confused on anything from the syllabus put `question: <your question here>`, replacing the `<>` part with our actual question in that section and then ask in class. There will be time in class to make revisions to your work before it is officially graded.
>
> I will be reading everything before class (and I can use GitHub timestamps to see what was done before and later)

1.1. Evaluation

Eligible skills:

- Python
- Process

1.2. Related notes

- Welcome & What is Data Science

1.3. Instructions

> **Important**
>
> If you have trouble, check the GitHub FAQ on the left first

Your task is to:

1. Install required software from the Tools & Resource page (should have been done before the first class)
2. Create your portfolio, using the link on Brightspace
3. Learn about your portfolio from the README file on your repository.
4. Follow instructions in the README to make your portfolio your own with information about yourself(completeness only) and your own definition of data science (graded for **level 1 process**)
5. complete the `success.md` file as per the instructions in the comments in that file (it is a syllabus quiz)
6. Create a Jupyter notebook called `grading.ipynb` and write a function that computes a grade for this course, with the docstring below. Your notebook will need to follow the course style guide, following style from other courses will not earn credit.
7. In the same notebook, iterate over the example anonymized gradebook below and compute a grade for each one.
8. Upload your `grading.ipynb` file to your portfolio (upload to main branch).

> **Warning**
>
> Do not merge your “Feedback” Pull Request

1.3.1. Docstring

```
    '''
    Computes a grade for CSC/DSP310 from numbers of achievements at each level

    Parameters:
    ------------
    level1_acheivements : int
      number of level 1 achievements earned
    level2_acheivements : int
      number of level 2 achievements earned
    level3_acheivements : int
      number of level 3 achievements earned

    Returns:
    --------
    letter_grade : string
      letter grade with possible modifier (+/-)
    '''
```

1.3.2. Students to check

Skip to main content

n all to a list.

```
acheivements_only = [[15,14,1],[15,15,10],[12,12,12]]
```

1.4. Help

(optional, but useful information)

1.4.1. Sample tests

Here are some sample tests you could run to confirm that your function works correctly, they are in one block here, but you should run them one at a time and with text between:

```
assert compute_grade(15,15,15) == 'A'
assert compute_grade(15,15,13) == 'A-'
assert compute_grade(15,14,14) == 'B-'
assert compute_grade(14,14,14) == 'C-'
assert compute_grade(4,3,1) == 'D'
assert compute_grade(15,15,6) =='B+'
```

1.4.2. Notebook Checklist

- a Markdown cell with a heading
- your function called `compute_grade`
- three calls to your function that verify it returns the correct value for different number of badges that produce at three different letter grades.
- markdown cells with explanation of the calls and anything else so that the notebook reads like a report.

2. Assignment 2: Practicing Python and Accessing Data

2.1. Deadline

2024-09-19 end of day

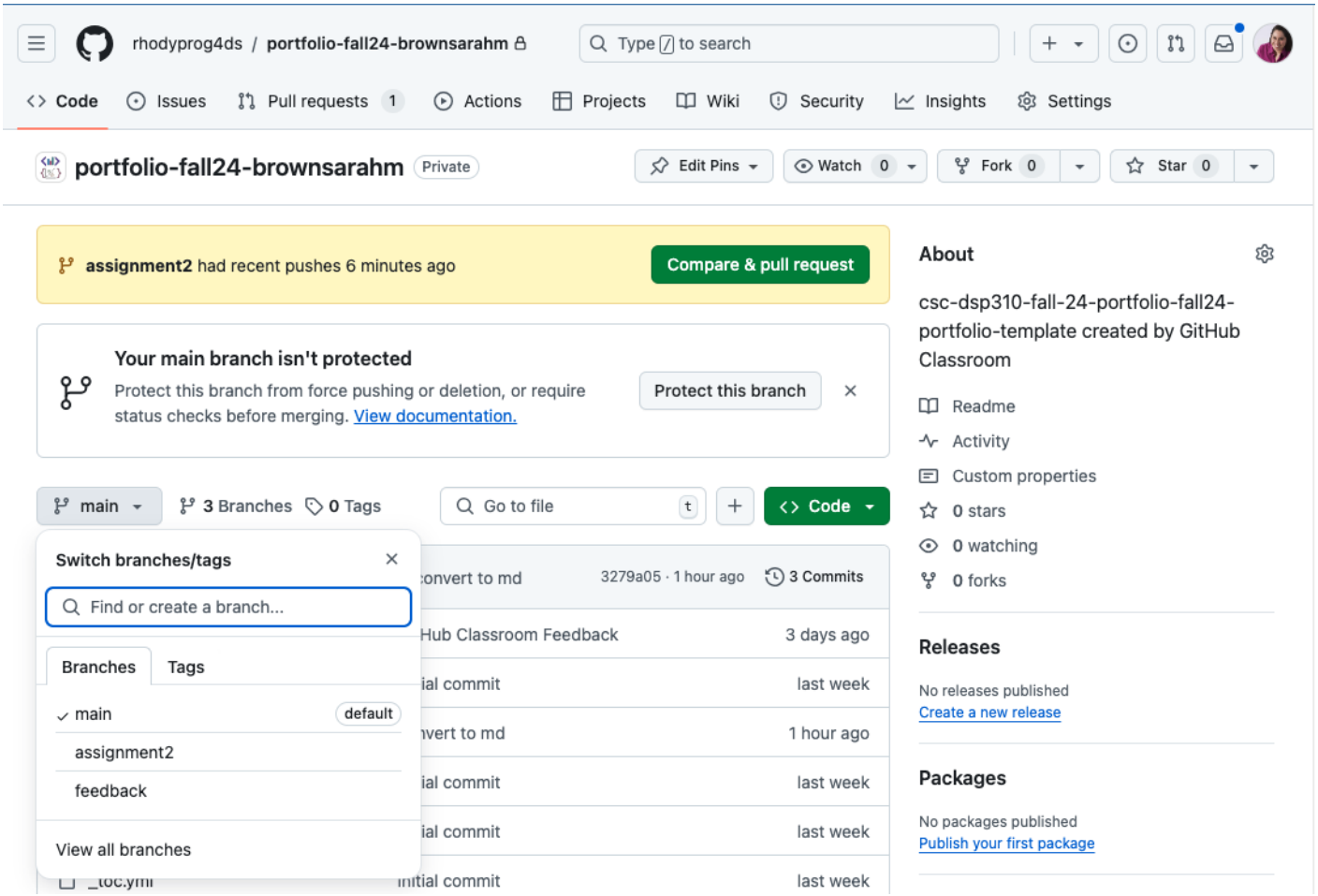
However, if you upload an attempt and specific questions on time and then attend office hours on 9/20 then any revisions made based on our conversation will count as on time

2.2. Submission and Template

Complete your work on the `assignment2` branch in your portfolio, a template file for `datasets.py` is located there, as well as

Skip to main content

In your repo you can see a menu on the left hand side of the page, select `assignment2` from the list and work from there.



2.3. Objective & Evaluation

This assignment is an opportunity to earn level 1 and 2 achievements in `python` and `access` and begin working toward level 1 for `summarize`. You can also earn level 1 for `process`.

Eligible skills: (links to checklists)

- process level 1
- python level 1 or 2
- access level 1
- summarize level 1

The goal of this assignment is for you to practice

1. finding datasets
2. getting familiar with different critieria we describe datasets with
3. practice loading data
4. practice getting basic facts about a dataset
5. manipulating dictionaries do do things efficiently

Skip to main content

7. creating your own DataFrame from other Python objects
8. saving a dataframe to csv (preview for cleaning)

These are all important skills that help you prepare to think in the right way to do more typical data analyses starting with assignment 3.

2.4. Related notes

- Iterables and Pandas Data Frames
-

2.5. Store info about data for loading

Find 3 datasets of interest to you that are provided in at least two different file formats. Choose datasets that are not too big, so that they do not take more than a few second to load. At least one dataset, must have non numerical (eg string or boolean) data in at least 1 column.

> **Important**
>
> So that we can use this in class next week, this part is due early

Create a list of dictionaries in file called `datasets.py`, so that there is one dictionary for each dataset. Each dictionary should have the following keys:

Table 2.1 Meta data of the dictionaries

`url`	the full url of the dataset
`short_name`	a short name
`load_function`	(the actual function handle) what function should be used to load the data into a `pandas.DataFrame`.

> **Hint**
>
> See below for how you will use the dictionary as help for how you should construct it

2.6. Analyze Your Datasets

Do the following in a notebook called `loading_data.ipynb`.

2.6.1. Document your data

In your notebook, create a markdown cell for each dataset that includes:

- heading 2 with the dataset's name
- a 1-2 sentence summary of what the dataset contains and why it was collected
- a "more info" link to where someone can learn about the dataset
- 2-3 questions you would like to answer with that dataset in a bulleted (`-`) or numbered list

Hint

markdown links are written like: `[text to show](url/to/go/to)`

2.6.2. Make a dataset about your datasets

In a notebook called `dataset_of_datasets.ipynb`, import the list of dictionaries from the `datasets` module you created in the step above. Then iterate over the list of dictionaries, and:

1. load each dataset like `dataset_dict['load_function'](dataset_dict['url'])`
2. save it to a local csv using the short name you provided for the dataset as the file name, without writing the index column to the file.
3. record attributes about the dataset as in the table below in a list or dictionary of lists
4. Use that to create a DataFrame with columns that match the rows of the following table.

Table 2.2 Meta Data Description of the DataFrame to build

name	a short name for the dataset
source	a url to where you found the data
num_rows	number of rows in the dataset
num_columns	number of columns in the dataset
num_numerical	number of numerical variables in the dataset

2.6.3. Explore Your Datasets

Create a second notebook file called `exploration.ipynb`:

For one dataset that includes nonnumerical data:

- read it in from your local csv using a relative path
- display the heading and the first 4 rows
- make a numpy array of only the numerical data and save it to a new variable (select these programmatically)
- was the format that the data was provided in a good format? why or why not?

For any other dataset:

- read it in from your local csv using a relative path

Skip to main content

- display the pandas datatype for each column
- Are there any variables where pandas may have read in the data as a datatype that's not what you expect (eg a numerical column mistaken for strings)? If so, investigate and try to figure out why.

For the third dataset:

- read it in from your local csv using a relative path
- save every third row (3,6,9,…) of the data for two columns of your choice into a new DataFrame and display that

2.6.4. Exploring data files

There are two files in the data folder, both can be read in with `read_csv` but need some options or fixing.

- try to read in the `german.data` file, what happens with the default settings? What option do you need to use to make it look right?
- try to read in the `.csv` file that's included in the template repository, use the error messages you get to try to fix the file manually (any text editor, including jupyter can edit a `.csv`), making notes about what changes you made in a markdown cell.

> **Hint**
>
> For the csv file in the template's data folder, in Jupyter Lab, it will not let you edit a .csv file, but you can chagne the file name to txt (in your code too) and then it will work.

2.7. Tips and hints

2.7.1. the goal

- I am not looking for "an answer"
- You should not be either
- I am looking for evidence that you **understand** the material (thus far including prereqs)
- You should be trying to **understand** material
- Review all of the methods the DataFrame has

This means that in office hours, I am going to:

- ask you questions to help you think about the problem and the material
- help direct your attention to the right part of the error message to figure out what is wrong
- `dtype` shows you the type of each column read more
- there is also a select dtypes method

Skip to main content

2.8. Thinking ahead

> **Important**
>
> This section is not required, but is intended to help you get started thinking about how you could extend this assignment. If you complete it, we'll give your feedback to help shape your ideas to get to level 3 achievements. If you want to focus only on level 2 at this moment in time, feel free to skip this part. You could also think about these after submitting the assignment. If you want, you could discuss these ideas in office hours.

1. When might you prefer one datatype over another?
2. How does PEP 8 standard code help you be collaborative?
3. Learn about Datasheets for Datasets and find some examples, (eg this google scholar result) How could something like this impact your work as a data scientist?

3. Assignment 3: Exploratory Data Analysis

Due:2023-10-01 end of day

3.1. Submission

> **Important**
>
> You have the option to work with a partner. You must plan this in advance so that you have access to collaborate.

3.1.1. Solo

Add your work to the assignment3 branch in your portfolio and you do not need to edit the `a3_location` file

3.1.2. Group

1. coordinate so that the first person makes the team when they accept the assigment
2. the second (and third) joins the same team when they accept the assigment.
3. Each person should upload their work to a branch named `d1`, `d2` or `d3` for which dataset checklist you followed from below and open a PR. (each person should do a different dataset)
4. In your portfolio, replace the contents of the `a3_location.md` file on the assignment3 branch with your team name. We will use that to create a PR to give you your individualized achievements update.

3.2. Objective & Evaluation

choice.

Eligible skills: (links to checklists)

- process 1
- access 1 and 2
- summarize 1 and 2
- visualize 1

3.3. Related notes

- Exploratory Data Analysis
- Visualization

3.4. Choose Datasets

Each Dataset must have at least three variables, but can have more. Both datasets must have multiple types of variables. These **can** be datasets you used for Assignment 2, if they meet the criteria below. All datasets must be different datasets even in a group

3.4.1. Dataset 1 (d1)

must include at least:

- two continuous valued variables **and**
- one categorical variable.

> **Hint**
>
> a dataset from the UCI data repository that's for classification and has continuous features would work for this

3.4.2. Dataset 2 (d2)

must include at least:

- two categorical variables **and**
- one continuous valued variable

3.4.3. Dataset 3 (d3)

> **Warning**
>
> This is only for groups of 3

- two continuous valued variables **and**
- one categorical variable.

3.5. EDA

Use a separate notebook for each dataset, name them `dataset_0x.ipynb` where `x` is the number checklist you are following.

For **each** dataset, in the corresponding notebook complete the following:

1. Load the data to a notebook as a `DataFrame` from url or local path, if local, include the data file in your repository.
2. Write a short description of what the data contains and what it could be used for
3. Explore the dataset in a notebook enough to describe its structure. Use the heading `## Description` and include at least the following with interpretation. *What does the strucutre imply about the conclusions you can draw from this data? Are there limitations in how to safely interpret the data that the summary helps you see? are the variables what you expect?*
    - shape
    - columns
    - variable types
    - overall summary statisics
4. Ask and answer **at least** 3 questions by using and interpreting statistics and visualizations as appropriate. Include a heading for each question using a markdown cell and H2:`##`. **Make sure your analyses meet the criteria in the check lists below.** Use the checklists to think of what kinds of questions would use those type of analyses and help shape your questions. Your questions can be related or different levels of detail or views on a big picture question as long as the analysis addresses the checklist.
5. Describe what, if anything might need to be done to clean or prepare this data for further analysis in a finale `## Future analysis` markdown cell in your notebook.

3.5.1. (overall) Question checklist

be sure that every question (all six, 3 per dataset) has:

- a heading
- at least 1 statistic or plot
- interpretation that answers the question
- the question does not include the name of the statistic or plot in it

3.5.2. Dataset 1 Checklist

make sure that your `dataset_01.ipynb` has:

- Overall summary statistics grouped by a categorical variable
- A single statistic grouped by a categorical variable

Skip to main content

- a plot and summary table that convey the same information. This can be one statistic or many.

3.5.3. Dataset 2 Checklist

make sure that your `dataset_02.ipynb` has:

- overall summary statistics
- two individual summary statistics for one variable
- one summary statistic grouped by two categorical variables
- a figure with a grid of subplots that correspond to two categorical variables

3.5.4. Dataset 3 Checklist

> **Warning**
>
> This is only for groups of 3

make sure that your `dataset_03.ipynb` has:

- overall summary statistics
- two individual summary statistics for one variable
- at least one plot that uses 3 total variables
- a plot and summary table that convey the same information. This can be one statistic or many.

3.6. Peer Review

> **Note**
>
> If you work alone and complete 2 analyses you do not need to do this, but you might review these questions because they are similar to how we will grade.

With a partner (or group of 3 where person 1 review's 2 work, 2 reviews 3, and 3 reviews 1) read your partner's notebook and complete a peer review on their pull request. You can do peer review when you have done most of your analysis, and explanation, even if some parts of the code do not work.

You will complete your review on a PR, by reviewing it. If you want a big picture overview on that, the github PR review "course" is a good place to go, it is designed to take <30 minutes.

1. start a review
2. (optional) Use PR comments to denote places that are confusing or if you see solutions to problems your classmate could not solve *this is hard on notebook files, so it is okay to skip*
3. Prepare to submit your review
4. Use the list of questions below for your summary review (copy the template into the box and fill in.)

Skip to main content

> **Important**
>
> Your review should use the **template** for organization, but the **questions** guide what sorts of aspects to consider across the sections.

3.6.1. Review Questions

1. Describe overall how it was to read the analysis overall to read. Was it easy? hard? cohesive? jumpy?
2. How did the data summaries help prepare you to read the rest of the analysis? What do you think might be missing?
3. For each question, consider the following and write any tips for improvement
    1. Does the question make sense based on the data? How does it relate to the real world is there a reasonable audience? How could the question be improved
    2. How well do the statistics and plots match the question?
    3. Are the interpretations complete, clear, and consistent with the statistics and plots?
    4. What could be done to make the explanations more clear and complete?
    5. What additional analysis might make the analysis more compelling and clear?

3.6.1.1. Template

```
## Overall
 <!-- Describe overall how it was to read the analysis overall to read. Was it easy? hard? cohesive? jump


## Intro

## Question 1

## Question 2

## Question 3
```

3.6.2. Response

Respond to the review on your notebook either with inline comments, replies, or by updating your analysis accordingly.

3.7. Tips and Hints

- Remember you can also use masking in your EDA even though we did not do any in class
- To ensure you understand the checklist you can **optionally** make an issue using the appropriate issue type from your repo and fill in what it should be to get early feedback that you are on track
- variable types are in the notes
- the DataFrame API reference shows all the methods (and more) grouped by high level concepts.

3.8. Think Ahead

This can be addded to any or all of the datasets

```
## Thinking Ahead
1. How could you make more customized summary tables?
1. Could you use any of the variables in this dataset to add more variables that would make interesting w
1. Are there multiple ways to answer your big picture question (like different thresholding or subsets of
1. Could any cleaning improve your analyis?
```

4. Assignment 4: Cleaning Data

Due: 2023-10-03

Eligible skills:

- prepare 1
- access 1
- python 1,2

4.1. Submission

Work on your assignment4 branch. When you are done, open a PR with:

base: feedback, compare: assignment4

and request a review from @surbhir08

4.2. Related notes

- Cleaning Data - Structure
- Fixing Values

4.3. Check the Datasets you have worked with already

In the datasets you have used or come across but decided you could not work with in your past assignments identify at least one thing you could not do because the data was not in an appropriate format.

In a notebook file called `dataset_fix.ipynb` apply one fix and show one summary statistic or plot that was not possible before to show that it works.

Some examples:

- a column that was a list or dictionary

Skip to main content

- missing values
- a column that was continuous, but more interesting as a categorical
- too many header rows
- a data set that was wide, but tall would be better for plotting

4.4. CS Degrees

See the notebook on your assignment4 branch and complete the instructions there.

4.5. Study Cleaned Datasets

> **Tip**
>
> there is a dedicated section in the Data Sources page

Read example data cleaning notes or scripts. To do this find at least one dataset for which the messy version, clean version, and a script or notes about how it was cleaned are available, answer the following questions in a markdown file, named `cleaning_notes.md`. (some example datasets are on the datasets page and one is in the notes are added to the course website)

1. What are 3 common problems to look for in a dataset? Describe them with examples.
2. Using one of the examples you found of cleaned data, give an example of a question or context that would require making different choices for cleaning than were made. Include a bit about the data, what was done, the question, what would need to be done instead and justification.
3. Explain in your own words, with a concrete example, how domain expertise can help you when cleaning data. Use either a made up example or one that you read about.

> **Warning**
>
> Some of these examples have both the clean and messy data files and an R script to do the cleaning. You are not required to *know* R, but looking at their R cleaning script could give hints of what things they fixed or changed. You could also compare the clean and messy versions by looking at them with a tool of your choice.

5. Assignment 5: Constructing Datasets and Using Databases

due date : 2023-10-10

Skills:

- prepare level 1
- summarize 1,2
- visualize 1,2

Skip to main content

5.1. Related notes

-
-

5.2. Constructing Datasets

Your goal is to programmatically construct a ready to analyze dataset that combines information from multiple sources. This can be in a crawing fashion like we did for the CS people or by combining two tables with a merge. If you use a merge to meet the multiple sources criterion, only one source must be scraped, the second can be provided as tabular data.

The notebook you submit should include:

- a motivating question for why your are building the dataset you are building
- code and description of how you built and prepared your dataset. For each step, describe what you're about to do, the code with output, interpretation that leads into the next step.
- exploratory data analysis that shows why you built the data and confirms that is prepared enough to analyze.
- also save your dataset to csv

For construct only, this can be very minimal EDA.

5.3. Additional achievements

To earn additional achievements, you must do more cleaning and/or exploratory data analysis.

5.3.1. Prepare level 2

To earn level 2 for prepare, you must manipulate either a component table or the final dataset. Sample manipulations include:

- transform into a tidy format
- add a new column by computing from others
- handle NaN values by dropping or filling
- drop a column, row, or duplicates in another way
- change a continuous value to categorical (there is an added section in the notes on quantizing that we did not do in class, but should be easy to follow)

5.3.2. Summarize and Visualize level 2

To earn level 2 for summarize and/or visualize, include additional analyses after building the datasets.

Connect your EDA to questions, and demonstrate demonstrate items from one of the checklists in A3.

Skip to main content

5.3.3. Python Level 2

Use pythonic naming conventions throughout, AND:

- Use pythonic loops and a list or dictionary OR
- use a list or dictionary comprehension

this can be in your cleanup or your EDA

> **Thinking Ahead**
>
> Compare the level 2 skill definitions to level 3, how could you extend and adapt what you've done to meet level 3?

> **Thinking Ahead**
>
> You could also demonstrate understanding of how merges work by converting a dataset that is provided as a single table with redundant information into a number of smaller tables in a database.

Earning Level 3

Starting in week 3 it is recommended that you spend some time each week working on extensions.

Use the feedback you get on assignments to inspire your extensions.

Formatting Tips

> **Warning**
>
> This is all based on you having accepted the portfolio assignment on github and having a cloned copy of the template. If you are not enrolled or the initial assignment has not been issued, you can view the template on GitHub

Your portfolio is a jupyter book. This means a few things:

- it uses myst markdown
- it will run and compile Jupyter notebooks

This page will cover a few basic tips.

Managing Files and versions

> **Important**

Skip to main content

Organization

The summary of for the `part` or whole submission, should match the skills to the chapters. Which prompt you're addressing is not important, the prompts are a *starting point* not the end goal of your portfolio.

Data Files

Also note that for your portfolio to build, you will have to:

- include the data files in the repository and use a relative path OR
- load via url

using a full local path(eg that starts with `///file:`) **will not work** and will render your portfolio unreadable.

Structure of plain markdown

Use a heading like this:

```
# Heading of page
## Heading 2
### Heading 3
```

in the file and it will appear in the sidebar.

You can also make text *italic* or **bold** with either `*asterics*` or `__underscores__` with `_one for italic_` or `**two for bold**` in either case

File Naming

It is best practice to name files *without* spaces, underscores `_` or hyphens `-` are both good. Each `chapter` or file should have a descriptive file name (`with_no_spaces`) and descriptive title for it.

Adding annotations with formatting or margin notes

You can either install jupytext and convert locally or upload /push a notebook to your repository and let GitHub convert. Then edit the .md file with a text editor of your choice. You can run by uploading if you don't have jupytext installed, or locally if you have installed jupytext or jupyterbook.

In your .md file use backticks to mark special content blocks

```
```{note}
Here is a note!
```
```

```
```{warning}
Here is a warning!
```
```

```
```{tip}
Here is a tip!
```
```

```
```{margin}
Here is a margin note!
```
```

For a complete list of options, see the `sphinx-book-theme` documentation.

Links

Markdown syntax for links

```
[text to show](path/or/url)
```

Configurations

Things like the menus and links at the top are controlled as settings, in `_config.yml`. The following are some things that you might change in your configuration file.

Show errors and continue

To show errors and continue running the rest, add the following to your configuration file:

```
# Execution settings
execute:
  allow_errors              : true
```

Using additional packages

You'll have to add any additional packages you use (beyond pandas and seaborn) to the `requirements.txt` file in your portfolio.

Generic Extension Ideas

Extension

If there were parts of your previous assignments that you thought were interesting and you want to work with those data more, you can. You need to do *more complex* analyses of them, but you can build off of what you already have done, especially for assignments 2, 3, and 5.

Extend Assignment 7, 8, or 9

Assignments 7-9 help you think through what machine learning tasks are. Extend those ideas by adding additional experiments based on your own questions or the questions in your feedback.

Learn a new model

Repeat what you did in 7, 8, or 9, with a different model.

Alternatives to Extending Assignments for Level 3

These are other ways you can earn level 3 acheivements besides adding onto previous assigments.

If your goal is, for example, a B+ (you need 5 level 3s) you could only do 1-2 skills per portfolio check (there are 4 checks).

Tutorial

Write a notebook that explains a concept related to a skill with examples in a real dataset and with visuals or a toy dataset (minimal number of columns rows)

Cheatsheet

Make a detailed reference with code outputs on a topic or a few topics.

Blog post

Write a blog post styled Notebook that compares or analyzes something, for example:

- how do different ways of loading data compare
- describe best practices you've learned and show why they're good with examples

Skip to main content

Practice Problems and Solutions

Based on the level 3 rubric descriptions, write practice problems that build off of the lecture notes. Include solutions and descriptions for each. These can be open ended or multiple choice questions with plausible distractors. A plausible distractor is an incorrect answer that represents a way that you think someone could misunderstand.

For example if the question is 37 + 15 = ?, MCQ with plausible distractors might be:

- 52 (correct)
- 412 (didn't carry the one, correctly: 7+5 = 12, 3+1 = 4)
- 42 (dropped the one 7+5 = 12, ones place is 2, 3+1 = 4)
- 43 (carried one into wrong column, 7 + 5 = 12, 1+2 = 3, 3+1 = 3)

Long single analysis

Collect data from multiple sources, prepare each for analysis, and merge them together then do some exploratory data analysis. Describe each step, interpret all outputs, and put the analysis in context of the Data Science Process.

This would be one long notebook that covers many skills at once.

Create datasets that fail

Create datasets that violate assumptions of a model we have learned. The sklearn data generators are a good place to start.

Build a data set for Prediction

Build a dataset that works for prediction (classification, regression, or clustering) from other sources.

Organize your knowledge

Develop some sort of visual aid that demonstrates how you understand some aspect(s) of data science working. Think of this as something that future students could use to help them learning, so assume prior knowledge topics covered earlier than the one you are demonstrating.

This could be a concept map, a table that shows how you've traced how something works or any other sort of conceptual tol that helps convey your understanding.

Try alternative libraries/ tools

One option for workflow level 3 is to use other data science skills and reflect on how what we have learned so far helped you learn a new set of tool as an alternative way to do things.

Skip to main content

Try feature engineering or representation learning

Try different transformations and see how they impact how well a model performs. This could be using `sklearn.feature_extraction` tools or trying different types of neural network layers at the beginning.

Process Level 3

Process level 3

Process level 3 is a little different than most of the others. You may be able to work it into an analysis notebook, but likely, you'll need to do one of the following.

Data Science Pipeline Comparisons

Find two different sources that describe the data science pipeline or lifecycle. Write a blog style post that discusses their differences and hypothesizes about why they may be different? Are they for different audiences? Is one domain specific? How do they emphasize different modeling tasks? Include a Recommendation for when you think each one is better

Write a short story

Write a short story that explains the concepts of data science to demonstrate your understanding of process.

Media Review

Watch/listen/read to an episode of a high quality[1] podcast or other type of media and write a blog style summary and review. Highlight what you learned and how it relates to topics covered in class.

Approved Media:

- Pod of Asclepius, Fall Series: The Philosophy of Data Science
- Chapter 1 & 2 of Think like a Data Scientist in particular, if you think these would be helpful to assign as reading or teach from at the beginning of the semester next year.
- Algorithms of Oppression (book)
- Weapons of Math Destruction (book)
- Coded Bias (film, available on netflix & PBS)

---

[1] approved Dr. Brown by creating a pull request to add it to the list on this page that is successfully merged. To create a PR, use the suggest an edit button at the top of this page.

FAQ

This section will grow as questions are asked and new content is introduced to the site. You can submit questions:

- via e-mail to Dr. Brown (brownsarahm) or TA
- via Prismia.chat during class
- by creating an issue

Syllabus and Grading FAQ

How much does assignment x, class participation, or a portfolio check weigh in my grade?

What time are assignments due?

Can I submit this assignment late if …?

I don’t understand my grade on this assignment

Git and GitHub

I can’t push to my repository, I get an error that updates were rejected

The content I added to my portfolio isn’t in the pdf

My command line says I cannot use a password

My .ipynb file isn’t showing in the staging area or didn’t push

My portfolio won’t compile

Help! I accidentally merged the Feedback Pull Request before my assignment was graded

Skip to main content

Skip to main content

Skip to main content

Skip to main content

Skip to main content

Code Errors

Key Error

<bound method

Glossary

aggregate

to combine data in some way, a function that can produce a customized summary table

Skip to main content

a function that's defined on the fly, typically to lighten syntax or return a function within a function. In python, they're defined with the lambda keyword.

BeautifulSoup

a python library used to assist in web scraping, it pulls data from html and xml files that can be parsed in a variety of different ways using different methods.

conditional

a logical control to do something, conditioned on something else, for example the `if`, `elif` `else`

corpus

(NLP) a set of documents for analysis

DataFrame

a data structure provided by pandas for tabular data in python.

dictionary

(data type) a mapping array that matches keys to values. (in NLP) all of the possible tokens a model knows

document

unit of text for analysis (one sample). Could be one sentence, one paragraph, or an article, depending on the goal

gh

GitHub's command line tools

git

a version control tool; it's a fully open source and always free tool, that can be hosted by anyone or used without a host, locally only.

GitHub

a hosting service for git repositories

index

(verb) to index into a data structure means to pick out specified items, for example index into a list or a index into a data frame. Indexing usually invovlees square brackets `[]` (noun) the index of a dataframe is like a column, but it can be used to refer to the rows. It's the list of names for the rows.

interpreter

the translator from human readable python code to something the computer can run. An interpreted language means you can work with python interactively

iterate

To do the same thing to each item in an iterable data structure, typically, an iterable type. Iterating is usually described as iterate over some data structure and typically uses the `for` keyword

iterable

Skip to main content

t there are others.

kernel

in the jupyter environment, the kernel is a language specific computational engine

lambda

they keyword used to define an anonymous function; lambda functions are defined with a compact syntax `<name> = lambda <parameters>: <body>`

numpy array

a type provided by numpy to represent matrices, used by `pd.DataFrame.values` doc and accessed by `pd.DataFrame.to_numpy` doc

PEP 8

Python Enhancement Proposal 8, the Style Guide for Python Code.

repository

a project folder with tracking information in it in the form of a .git file

Series

a data structure provided by pandas for single columnar data with an index. Subsetting a Dataframe or applying a function to one will often produce a Series

shape

of a dataframe, or matrix is the number of rows and columns.

Split Apply Combine

a paradigm for splitting data into groups using a column, applying some function(aggregation, transformation, or filtration) to each piece and combinging in the individual pieces back together to a single table

stop words

Words that do not convey important meaning, we don't need them (like a, the, an,). Note that this is context dependent. These words are removed when transforming text to numerical representation

suffix

additional part of the name that gets added to end of a name in a merge operation

test accuracy

percentage of predictions that the model predict correctly, based on held-out (previously unseen) test data

Tidy Data Format

Tidy data is a database format that ensures data is easy to manipulate, model and visualize. The specific rules of Tidy Data are as follows: Each variable is a column, each row is an observation, and each observable unit is a table.

token

a sequence of characters in some particular document that are grouped together as a useful semantic unit for processing (typically a word, but more gneeral)

Skip to main content

an error message in python that traces back from the line of code that had caused the exception back through all of the functions that called other functions to reach that line. This is sometimes call tracing back through the stack

training accuracy

percentage of predictions that the model predict correctly, based on the training data

Web Scraping

the process of extracting data from a website. In the context of this class, this is usually done using the python library beautiful soup and a html parser to retrieve specific data.

References on Python

Official Documentation

- Python
- Pandas
- Matplotlib
- Seaborn

Key Resources

- Course Text this book roughly covers things that we cover in the course, but since things change quickly, we don't rely on it too closely
- Real Python this site includes high quality tutorials
- Towards Data Science this blog has some good tutorials, but old ones are not always updated, so always check the date and don't rely too much on posts more than 2 years old.

> **Ram Token Opportunity**
>
> If you find other high quality, reliable sources that you want to share, you can earn ram tokens.

Cheatsheet

Patterns and examples of how to use common tips in class

How to use brackets

symbol	use
`[val]`	indexing item val from an object; `val` is int for iterables, or any for mapping
`[val : val2]`	slicing elemtns val to val2-1 from a listlike object
`[ item1,item2 ]`	creating a list consisting of `item1` and `item2`
`(param)`	function calls
`(item1,item2)`	defining a tuple of `item1` and `item2`
`{item1,item2}`	defining a set of `item1` and `item2`
`{key:val1, key2: val2}`	defining a dictionary where key1 indexes to val2

Axes

First build a small dataset that's just enough to display

```
data = [[1,0],[5,4],[1,4]]
df = pd.DataFrame(data = data,
  columns = ['A','B'])

df
```

	A	B
0	1	0
1	5	4
2	1	4

This data frame is originally 3 rows, 2 columns. So summing across rows will give us a Series of length 3 (one per row) and long columns will give length 2, (one per column). Setting up our toy dataset to not be a square was important so that we can use it to check which way is which.

```
df.sum(axis=0)
```

```
A    7
B    8
dtype: int64
```

```
```

Skip to main content

```
0    1
1    9
2    5
dtype: int64
```

```
df.apply(sum,axis=0)
```

```
A    7
B    8
dtype: int64
```

```
df.apply(sum,axis=1)
```

```
0    1
1    9
2    5
dtype: int64
```

Indexing

```
df['A'][1]
```

```
5
```

```
df.iloc[0][1]
```

```
0
```

Data Sources

This page is a semi-curated source of datasets for use in assignments. The different sections have datasets that are good for different assignments.

Best for loading directly into a notebook

- Tidy Tuesday inside the folder for each year there is a README file with list of the datasets. These are .csv files
- Json Datasets
- National Center for Education Statistics Digest 2019 These data tables are available for download as excel and visible on the page.

Skip to main content

Cleaning Examples

- Messy Artists .csv file, that needs to be cleaned, containing data on artists
- Messy Wheels .csv file, that needs to be cleaned, containing data on various wheel attractions around the globe
- Clean Artists .csv file, already cleaned, containing data on artists
- Clean Wheels, .csv file, already cleaned, containing data on various wheel attractions around the globe
- Women's Rugby
- Web page metrics
- data cleaning with open refine on survey data this is a tutorial for cleaning data with another tool, but it demonstrates common problems with data well.
- data clearning for ecology this is a tutorial for cleaning data with another tool, but it demonstrates common problems with data well.
- us solar data
- NYT Data Preparation document
- Corporate Repuation Rankings

General Sources

These may require some more work

- Stackoverflow Developer Survey This data comes with readme info all packaged together in a .zip. You'll need to unzip it first.
- Google Dataset Search
- Kaggle most Kaggle datasets will require you to download and unzip them first and then you can copy them into your repo folder.
- UCI Data Repository Machine Learning focused datasets, can filter by task
- A curated list of datasets by task It includes datasets for cleaning, visualization, machine learning, and "data analysis" which would align with EDA in this course.
- Hugging Face NLP Datasets lots of text datasets

Datasets in many parts

- Makeup Shades
- Kenya Census
- Wealth and Income over time
- UN Votes
- Deforestation
- Survivor
- Billboard

- Video games from steam 2021 and from 2019
- BBC Rap Artists
- character psychometrics
- weather forecast accuracy

Datasets with time

- Superbowl commercials

Databases

- SQLite Databases

If you have others please share by creating a pull request or issue on this repo (from the GitHub logo at the top right, `suggest edit`).

General Tips and Resources

This section is for materials that are not specific to this course, but are likely useful. They are not generally required readings or installs, but are options or advice I provide frequently.

on email

- how to e-mail professors

How to Study in this class

This is a programming intensive course and it's about data science. This course is designed to help you learn how to program for data science and in the process build general skills in both programming and using data to understand the world. Learning two things at once is more complex. In this page, I break down how I expect learning to work for this class.

Remember the goal is to avoid this:

Why this way?

Learning to program requires iterative practice. It does not require memorizing all of the specific commands, but instead learning the basic patterns.
Using reference materials frequently is a built in part of programming, most languages have built in help as a part of the language for this reason. This course is designed to have you not only learn the material, but also to build skill in learning to program. Following these guidelines will help you build habits to not only be successful in this class, but also in future programming.

A new bo
programm
Program
available
that linke

Skip to main content

Where are your help tools?

In Python and Jupyter notebooks, what help tools do you have?

Learning in class

Important

My goal is to use class time so that you can be successful with *minimal frustration* while working outside of class time.

Programming requires both practical skills and abstract concepts. During class time, we will cover the practical aspects and introduce the basic concepts. You will get to see the basic practical details and real examples of debugging during class sessions. Learning to debug something you've never encountered before and setting up your programming envrionment, for example, are *high frustration* activities, when you're learning, because you don't know what you don't know. On the other hand, diving deeper into options and more complex applications of what you have already seen in class, while challenging, is something I'm confident that you can all be successful at with minimal frustration once you've seen basic ideas in class. My goal is that you can repeat the patterns and processes we use in class outside of class to complete assignments, while acknowledging that you will definitely have to look things up and read documentation outside of class.

Each class will open with some time to review what was covered in the last session before adding new material.

To get the most out of class sessions, you should have a laptop with you. During class you should be following along with Dr. Brown, typing and running the same code. You'll answer questions on Prismia chat, when you do so, you should try running necessary code to answer those questions. If you encounter errors, share them via prismia chat so that we can see and help you.

After class

After class, you should practice with the concepts introduced.
This means reviewing the notes: both yours from class and the annotated notes posted to the course website.
When you review the notes, you should be adding comments on tricky aspects of the code and narrative text between code blocks in markdown cells. While you review your notes and the annotated course notes, you should also read the documentation for new modules, libraries, or functions introduced that day.

In the annotated notes, there will often be extra questions or ideas on how to extend and practice the concepts. Try these out.

If you find anything hard to understand or unclear, write it down to bring to class the next day.

Assignments

In assignments, you will be asked to practice with specific concepts at an intermediate level. Assignments will apply the concepts from class with minimal extensions. You will probably need to use help funcitons and read documentation to complete assignments, but mostly to look up things you saw in class and make minor variations. Most of what you need for

Skip to main content

Portfolios

In portfolios, your goal is to extend and apply the concepts taught in class and practiced in assignments to solve more realistic problms. You may also reflect on your learning in order to demonstrate deep understanding. These will require significant reading beyond what we cover in class.

Getting Help with Programming

Screenshots

Sending me a screenshot is almost guaranteed to *not* get you help. Not because I do not want to, but because I literally do not have the information to get you an answer.

Typically when someone does not know how to fix something from the error message, it is because they are reading the wrong part of the error message or looking at the wrong part of the code trying to find the problem.

This means they end up screenshotting that wrong thing, so I literally **cannot** tell what is wrong from the screenshot.

I am not being stubborn, but I do need the right information to be able to tell what is wrong. Debugging code requires context, if you deprive me that, then I cannot help.

To get asynchronous help:

- upload your whole notebook, errors and all
- create an issue on that repo

Asking Questions

One of my favorite resources that describes how to ask good questions is this blog post by Julia Evans, a developer who writes comics about the things she learns in the course of her work and publisher of wizard zines.

Describing what you have so far

Stackoverflow is a common place for programmers to post and answer questions.
As such, they have written a good guide on creating a minimal, reproducible example.

Creating a minimal reproducible example may even help you debug your own code, but if it does not, it will definitely make it easier for another person to understand what you have, what your goal is, and what's working.

Understanding Errors

Error messages from the compiler are not always straight forward.

The TraceBack can be a really long list of errors that seem like they are not even from your code. It will trace back to all of the places that the error occurred. It is often about how you called the functions from a library, but the compiler cannot tell that.

Skip to main content

To understand what the traceback is, how to read one, and common examples, see this post on Real Python.

One thing to try, is friendly traceback a python package that is designed to make that error message text more clear and help you figure out what to do next.

Ram Token Opportunity

If you try out friendly traceback and find it helpful, add a testimonial here. using

```
```{epigraph}
```
```

Terminals and Environments

Why all this work?

Managing environments is **one of the hardest parts of programming** so, as instructors, we often design our courses around not having to do it. In this class, however, I'm choosing to take the risk and help you all through beginning to manage your own environments.

These issues will be the most painful in the course, I promise.

I think it's worth this type of pain though, because all fo the code you ever run must run in *some* sort of environment. By giving you control, I'm hoping to increase your indepence as a programmer. This also means responsibility and some messy debugging, but I think this is a good tradeoff. This is an upper level (300+) level course, so increasing some complexity is expected and I want as much as possible to keep you close to realisitc programming environments; so that what you see in this course is **directly, and immediately,** applicable in real world contexts. You should be able to pick up data science side projects or an internship with ease after this course.

I *know* some of these things will be frustrating at times, but I want you to feel supported in that and know that your grade will not be blocked by you having environment issues, as long as you ask for help in a timely matter.

No

We
tea
so i
new
that
und
this
follo
here

If, for exa
have nev
and you'
will be hu
behind a
often.

Windows

Windows has a sort of multiverse of terminal environments.

The least setup required involves using anaconda prompt and `conda` to manage you python environment and GitBash to work with git (and it can also do other bash related things).

Instead of managing two terminals, you may configure your path in GitBash to make Anaconda work

MacOS

MacOS has one terminal app, but it can run different shells

Skip to main content

On MacOS You may want to switch to bash (using the `bash` command or make it your default and update bash.

Getting Organized for class

The only **required** things are in the Tools section of the syllabus, but this organizational structure will help keep you on top of what is going on.

Your username will be appended to the end of of the repository name for each of your assignments in class.

File structure

I recommend the following organization structure for the course:

```
CSC310
  |- notes
  |- portfolio-username
  |- 02-accessing-data-username
  |- ...
```

This is one top level folder will all materials in it. A folder inside that for in class notes, and one folder per repository.

Please **do not** include all of your notes or your other assignments all inside your portflio, it will make it harder to grade.

Finding repositories on github

Each assignment repository will be created on GitHub with the `rhodyprog4ds` organization as the owner, not your personal acount. Since your account is not the owner, they do not show on your profile.

Your assignment repositories are all private during the semester. At the end, you may take ownership of your portfolio[^pttrans] if you would like.

If you go to the main page of the organization you can search by your username (or the first few characters of it) and see only your repositories.

> ⚠ **Warning**
>
> Don't try to work on a repository that does not end in your username; those are the template repositories for the course and you don't have edit permission on them.