About this Book

Contents

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

This class meets TTh 5-6:15pm in Tyler 108.

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

Course Calendar on BrightSpace

Tip

subscribe to that calendar in your favorite calendar application

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.

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.

Questions that are asked in class, but unanswered at that time will be answered in the notes and marked with a box like this. Short questions will be in the margin note

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.

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 either way by the end of the semester.

About this syllabus

This syllabus is a *living* document and accessible from BrightSpace, as a pdf for download directly online at rhodyprog4ds.github.io/BrownFall20/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.

About your instructor

Name: Dr. Sarah Brown Office hours: TBA via zoom, link in BrightSpace

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.

Note

Whether you use CSC or DSP does not matter.

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.

All of the tools and resources below are either:

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

BrightSpace

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

Important

TL;DR [1]

- check Brightspace
- Log in to Prismia Chat
- Make a GitHub Account
- Install Python
- Install Git

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.

Note

Seeing the BrightSpace site requires loging in with your URI SSO and being enrolled in the course

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

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 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

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.

Note

all Git instructions will be given as instructions for the command line interface and GitHub specific instructions via the web interface. You may choose to use GitHub desktop or built in IDE tools, but the instructional team may not be able to help.

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.

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.

Video install instructions for Anaconda:

- 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

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 TTh 5-6:15pm in Tyler 108. 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 page. Portfolio deadlines will be announced at least 2 weeks in advance.

> **Note**
>
> On the Course Calendar on BrightSpace page you can get a feed link to add to the calendar of your choice by clicking on the subscribe (star) button on the top right of the page. Class is for 1 hour there because of Brightspace/zoom integration limitations, but that calendar includes the zoom link.

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 participation, assignments, and portfolios 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.

keyword				
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 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 optimized	Optimize basic model parameters such as model order	Select optimal parameters based of mutiple quanttiateve criteria and automate parameter tuning
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

keyword				
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

> ⚠ **Warning**
>
> **process** achievements are accumulated a little slower. Prior to portfolio check 1, only level 1 can be earned. Portfolio check 1 is the first chance to earn level 2 for process, then level 3 can be earned on portfolio check 2 or later.

Portfolios and Skills

The objective of your portfolio submissions is to earn Level 3 achievements. The following table shows what Level 3 looks like for each skill and identifies which portfolio submissions you can earn that Level 3 in that skill.

keyword					
python	reliable, efficient, pythonic code that consistently adheres to pep8	1	1	0	1
process	Compare different ways that data science can facilitate decision making	0	1	1	1
access	access data from both common and uncommon formats and identify best practices for formats in different contexts	1	1	0	1
construct	merge data that is not automatically aligned	1	1	0	1
summarize	Compute and interpret various summary statistics of subsets of data	1	1	0	1
visualize	generate complex plots with pandas and plotting libraries and customize with matplotlib or additional parameters	1	1	0	1
prepare	apply data reshaping, cleaning, and filtering manipulations reliably and correctly by assessing data as received	1	1	0	1
evaluate	Evaluate a model with multiple metrics and cross validation	0	1	1	1
classification	fit and apply classification models and select appropriate classification models for different contexts	0	1	1	1
regression	fit and explain regrularized or nonlinear regression	0	1	1	1
clustering	apply multiple clustering techniques, and interpret results	0	1	1	1
optimize	Select optimal parameters based of mutiple quanttiateve criteria and automate parameter tuning	0	0	1	1
compare	Evaluate tradeoffs between different model comparison types	0	0	1	1
representation	apply transformations in different contexts OR compare and contrast multiple representations a single type of data in terms of model performance	0	0	1	1
workflow	Independently scope and solve realistic data science problems OR independently learn releated tools and describe strengths and weakensses of common tools	0	0	1	1

Detailed Checklists

python-level1

python code that mostly runs, occasional pep8 adherance

- [ ] logical use of control structures
- [ ] callable functions
- [ ] correct calls to functions
- [ ] correct use of variables
- [ ] use of logical operators

python-level2

python code that reliably runs, frequent pep8 adherance

- [ ] descriptive variable names
- [ ] pythonic loops
- [ ] efficient use of return vs side effects in functions
- [ ] correct, effective use of builtin python iterable types (lists & dictionaries)

python-level3

reliable, efficient, pythonic code that consistently adheres to pep8

- [ ] pep8 adherant variable, file, class, and function names
- [ ] effective use of multi-paradigm abilities for efficiency gains

process-level1

Identify basic components of data science

- [ ] identify component disciplines OR
- [ ] idenitfy phases

process-level2

Describe and define each stage of the data science process

- [ ] correctly defines stages
- [ ] identifies stages in use
- [ ] describes general goals as well as a specific processes

process-level3

Compare different ways that data science can facilitate decision making

- [ ] describes exceptions to process and iteration in process
- [ ] connects choices at one phase to impacts in other phases
- [ ] connects data science steps to real world decisions

access-level1

load data from at least one format; identify the most common data formats

- [ ] use at least one pandas `read_` function correctly
- [ ] name common types
- [ ] describe the structure of common types

access-level2

Load data for processing from the most common formats; Compare and constrast most common formats

- [ ] load data from at least two of (.csv, .tsv, .dat, database, .json)
- [ ] describe advantages and disadvantages of most commone types
- [ ] descive how most common types are different

access-level3

access data from both common and uncommon formats and identify best practices for formats in different contexts

- [ ] load data from at least 1 uncommon format
- [ ] describe when one format is better than another

construct-level1

identify what should happen to merge datasets or when they can be merged

- [ ] identify what the structure of a merged dataset should be (size, shape, columns)
- [ ] idenitfy when datasets can or cannot be merged

construct-level2

- [ ] use 3 different types of merges
- [ ] choose the right type of merge for realistic scenarios

construct-level3

merge data that is not automatically aligned

- [ ] manipulate data to make it mergable
- [ ] identify how to combine data from many sources to answer a question
- [ ] implement stesp to combine data from multiple sources

summarize-level1

Describe the shape and structure of a dataset in basic terms

- [ ] use attributes to produce a description of a dataset
- [ ] display parts of a dataset

summarize-level2

compute and interpret summary standard statistics of a whole dataset and grouped data

- [ ] compute descriptive statistics on whole datasets
- [ ] apply individual statistics to datasets
- [ ] group data by a categorical variable for analysis
- [ ] apply split-apply-combine paradigm to analyze data
- [ ] interprete statistics on whole datasets
- [ ] interpret statistics on subsets of data

summarize-level3

Compute and interpret various summary statistics of subsets of data

- [ ] produce custom aggregation tables to summarize datasets
- [ ] compute multivariate summary statistics by grouping
- [ ] compute custom cacluations on datasets

visualize-level1

identify plot types, generate basic plots from pandas

- [ ] generate at least two types of plots with pandas
- [ ] identify plot types by name
- [ ] interpret basic information from plots

visualize-level2

generate multiple plot types with complete labeling with pandas and seaborn

- [ ] generate at least 3 types of plots
- [ ] use correct, complete, legible labeling on plots
- [ ] plot using both pandas and seaborn
- [ ] interpret multiple types of plots to draw conclusions

generate complex plots with pandas and plotting libraries and customize with matplotlib or additional parameters

- [ ] use at least two libraries to plot
- [ ] generate figures with subplots
- [ ] customize the display of a plot to be publication ready
- [ ] interpret plot types and explain them for novices
- [ ] choose appopriate plot types to convey information
- [ ] explain why plotting common best practices are effective

prepare-level1

identify if data is or is not ready for analysis, potential problems with data

- [ ] identify problems in a dataset
- [ ] anticipate how potential data setups will interfere with analysis
- [ ] describe the structure of tidy data
- [ ] label data as tidy or not

prepare-level2

apply data reshaping, cleaning, and filtering as directed

- [ ] reshape data to be analyzable as directed
- [ ] filter data as directed
- [ ] rename columns as directed
- [ ] rename values to make data more analyzable
- [ ] handle missing values in at least two ways
- [ ] transform data to tidy format

prepare-level3

apply data reshaping, cleaning, and filtering manipulations reliably and correctly by assessing data as received

- [ ] identify issues in a dataset and correctly implement solutions
- [ ] convert varialbe representation by changing types
- [ ] change variable representation using one hot encoding

evaluate-level1

Explain basic performance metrics for different data science tasks

- [ ] define at least two performance metrics
- [ ] describe how those metrics compare or compete

evaluate-level2

Apply and interpret basic model evaluation metrics to a held out test set

- [ ] apply at least three performance metrics to models
- [ ] apply metrics to subsets of data
- [ ] apply disparity metrics
- [ ] interpret at least three metrics

Evaluate a model with multiple metrics and cross validation

- [ ] explain cross validation
- [ ] explain importance of held out test and validation data
- [ ] describe why cross vaidation is important
- [ ] idenitfy appropriate metrics for different types of modeling tasks
- [ ] use multiple metriccs together to create a more complete description of a model's performance

classification-level1

identify and describe what classification is, apply pre-fit classification models

- [ ] describe what classification is
- [ ] describe what a dataset must look like for classifcation
- [ ] identify appliations of classifcation in the real world
- [ ] describe set up for a classification problem (tes,train)

classification-level2

fit, apply, and interpret preselected classification model to a dataset

- [ ] split data for training and testing
- [ ] fit a classification model
- [ ] apply a classification model to obtain predictions
- [ ] interpret the predictions of a classification model
- [ ] examine parameters of at least one fit classifier to explain how the prediction is made
- [ ] differentiate between model fitting and generating predictions
- [ ] evaluate how model parameters impact model performance

classification-level3

fit and apply classification models and select appropriate classification models for different contexts

- [ ] choose appropriate classifiers based on application context
- [ ] explain how at least 3 different classifiers make predictions
- [ ] evaluate how model parameters impact model performance and justify choices when tradeoffs are necessary

regression-level1

identify what data that can be used for regression looks like

- [ ] identify data that is/not appropriate for regression
- [ ] describe univariate linear regression
- [ ] identify appliations of regression in the real world

regression-level2

fit and interpret linear regression models

- [ ] split data for training and testing
- [ ] fit univariate linear regression models
- [ ] interpret linear regression models
- [ ] fit multivariate linear regression models

fit and explain regrularized or nonlinear regression

- [ ] fit nonlinear or regrularized regression models
- [ ] interpret and explain nonlinear or regrularized regresion models

clustering-level1

describe what clustering is

- [ ] differentiate clustering from classification and regression
- [ ] identify appliations of clustering in the real world

clustering-level2

apply basic clustering

- [ ] fit Kmeans
- [ ] interpret kmeans
- [ ] evaluate clustering models

clustering-level3

apply multiple clustering techniques, and interpret results

- [ ] apply at least two clustering techniques
- [ ] explain the differences between two clustering models

optimize-level1

Identify when model parameters need to be optimized

- [ ] identify when parameters might impact model performance

optimize-level2

Optimize basic model parameters such as model order

- [ ] automatically optimize multiple parameters
- [ ] evaluate potential tradeoffs
- [ ] interpret optimization results in context

optimize-level3

Select optimal parameters based of mutiple quanttiateve criteria and automate parameter tuning

- [ ] optimize models based on multiple metrics
- [ ] describe when one model vs another is most appropriate

compare-level1

Qualitatively compare model classes

- [ ] compare models within the same task on complexity

compare-level2

- [ ] compare models in multiple terms
- [ ] interpret cross model comparisons in context

compare-level3

Evaluate tradeoffs between different model comparison types

- [ ] compare models on multiple criteria
- [ ] compare optimized models
- [ ] jointly interpret optimization result and compare models
- [ ] compare models on quanttiateve and qualitative measures

representation-level1

Identify options for representing text and categorical data in many contexts

- [ ] describe the basic goals for changing the representation of data

representation-level2

Apply at least one representation to transform unstructured or inappropriately data for model fitting or summarizing

- [ ] transform text or image data for use with ML

representation-level3

apply transformations in different contexts OR compare and contrast multiple representations a single type of data in terms of model performance

- [ ] transform both text and image data for use in ml
- [ ] evaluate the impact of representation on model performance

workflow-level1

Solve well strucutred fully specified problems with a single tool pipeline

- [ ] pseudocode out the steps to answer basic data science questions

workflow-level2

Solve well-strucutred, open-ended problems, apply common structure to learn new features of standard tools

- [ ] plan and execute answering real questions to an open ended question
- [ ] describe the necessary steps and tools

workflow-level3

Independently scope and solve realistic data science problems OR independently learn releated tools and describe strengths and weakensses of common tools

- [ ] scope and solve realistic data science problems
- [ ] compare different data science tool stacks

Grading

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 problems as instructed by, for example, a supervisor in an internship or entry level job. This is a very accessible 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

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. 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. You will have three *types* of opportunities to demonstrate your current skill level: participation, assignments, and a portfolio.

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 earn level 1 achievements in class, assignments, or portfolio submissions.
- 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. You can earn level 2 achievements in assignments or portfolio submissions.
- 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. You can earn level 3 achievements only through your portfolio submissions.

For each skill these are defined in the Achievement Definition Table

Participation

through the classroom chat platform Prismia.chat; these records will be used to update your skill progression. You can also earn level 1 achievements from adding annotation to a section of the class notes.

Assignments

For your learning to progress and earn level 2 achievements, you must practice with the skills outside of class time.

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.

> **Warning**
>
> If you will skip an assignment, please accept the GitHub assignment and then close the Feedback pull request with a comment. This way we can make sure that you have support you need.

Portfolio Checks

To earn level 3 achievements, you will build a portfolio consisting of reflections, challenge problems, and longer analyses over the course of the semester. You will submit your portfolio for review 4 times. The first two will cover the skills taught up until 1 week before the submission deadline.

The third and fourth portfolio checks will cover all of the skills. The fourth will be due during finals. This means that, if you have achieved mastery of all of the skills by the 3rd portfolio check, you do not need to submit the fourth one.

Portfolio prompts will be given throughout the class, some will be strucutred questions, others may be questions that arise in class, for which there is not time to answer.

TLDR

You *could* earn a C through in class participation alone, if you make nearly zero mistakes. To earn a B, you must complete assignments and participate in class. To earn an A you must participate, complete assignments, and build a portfolio.

Detailed mechanics

On Brightspace there are 45 Grade items that you will get a 0 or a 1 grade for. These will be revealed, so that you can view them as you have an opportunity to demonstrate each one. The table below shows the minimum number of skills at each level to earn each letter grade.

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+.

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.

> In this example, you will have also achieved level 1 on all of the skills, because it is a prerequisite to level 2.

The letter grade can be computed as follows

> **Important**
>
> this will be revealed after assignment 1

For example you can run the code like this in a cell to see the output

```
compute_grade(15,15,15)
```

```
'A'
```

```
compute_grade(14,14,14)
```

```
'C-'
```

Or use `assert` to test it formally

```
assert compute_grade(14,14,14) == 'C-'
```

```
assert compute_grade(15,15,15) == 'A'
```

```
assert compute_grade(15,15,11) == 'A-'
```

Late work

> **Note**
>
> You may visit office hours to discuss assignments that you did not complete on time to get feedback and check your own understanding, but they will not count toward skill demonstration.

Late assignments will not be graded. Every skill will be assessed through more than one assignment, so missing assignments occasionally not necessarily hurt 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 will not be able to earn the level 3 achievement in that skill. If you submit work that is not complete, however, it will be assessed and receive feedback. Submitting pseudocode or code with errors and comments about what you have tried could earn a level 1 achievement. Additionally, most 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.

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.

In this issue, include:

1. A new deadline proposal
2. What additional work you plan to add
3. Why the extension is important to your learning

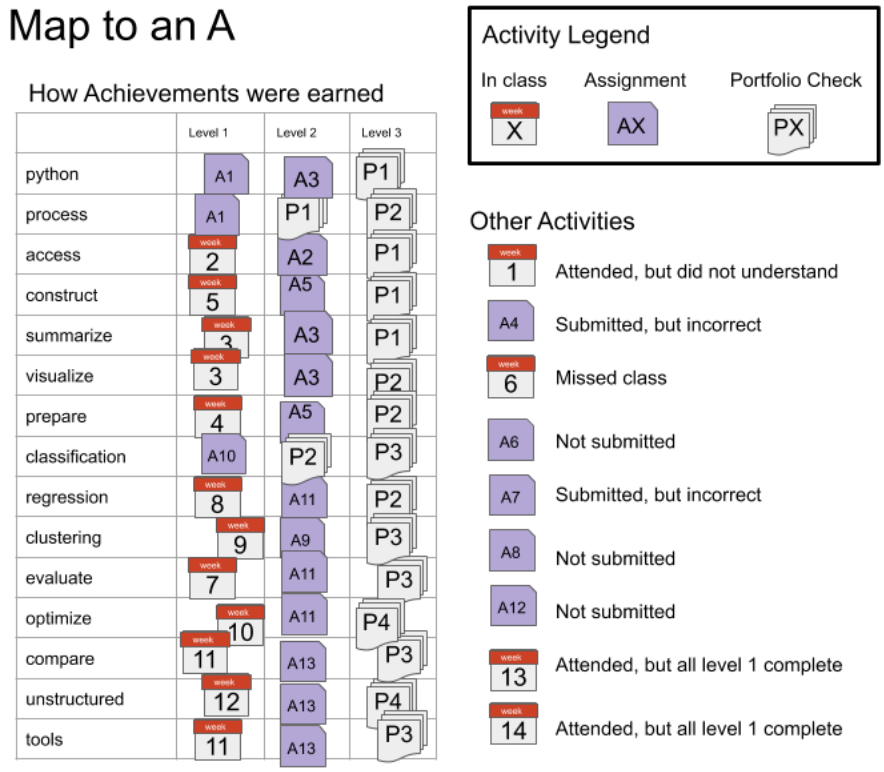
This request should be no more than 7 sentences.

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.

Grading Examples

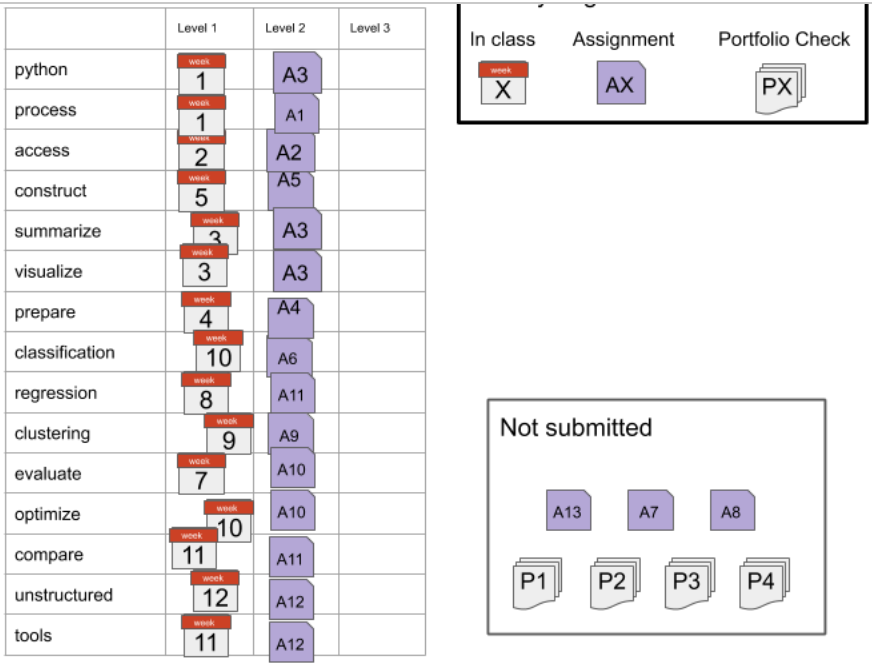
If you always attend and get everything correct, you will earn and A and you won't need to submit the 4th portfolio check.

Getting an A Without Perfection



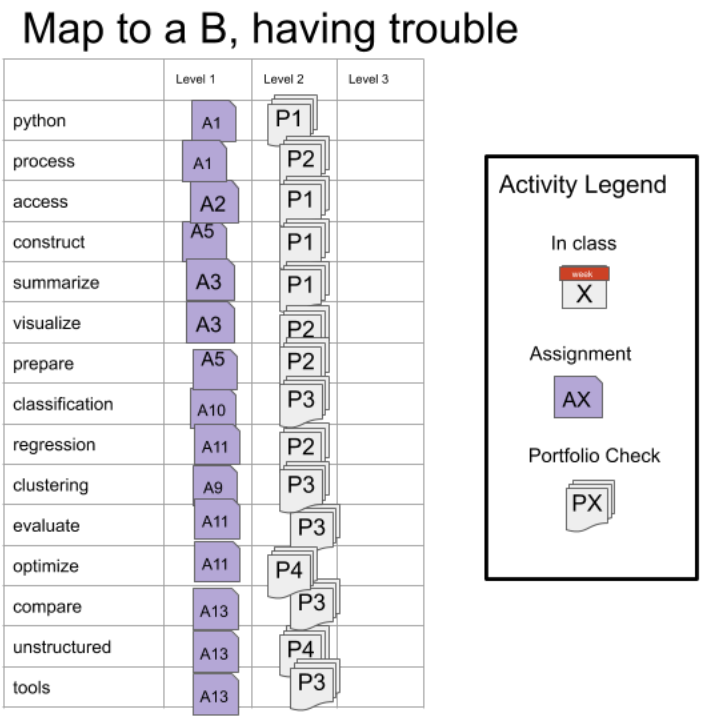
In this example the student made several mistakes, but still earned an A. This is the advantage to this grading scheme. For the `python`, `process`, and `classification` skills, the level 1 achievements were earned on assignments, not in class. For the `process` and `classification` skills, the level 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



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



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.

Academic Dishonesty

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

- In an assignment the level three achievements for the skills of focus in the assignment are ineligible, and the relevent level two for those skills requires meeting the standard for the level 3.

For example, if you are caught violating academic honesty 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 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+, beause level 3 in all of those skills becomes inelgible.

Grading Policies

Late Work

> **Note**
>
> You may visit office hours to discuss assignments that you did not complete on time to get feedback and check your own understanding, but they will not count toward skill demonstration.

Late assignments will not be graded. Every skill will be assessed through more than one assignment, so missing assignments occasionally not necessarily hurt 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 will not be able to earn the level 3 achievement in that skill. If you submit work that is not complete, however, it will be assessed and receive feedback. Submitting pseudocode or code with errors and comments about what you have tried could earn a level 1 achievement. Additionally, most 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.

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.

In this issue, include:

1. A new deadline proposal
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 assignment on time.

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.

Regrading

Re-request a review on your Feedback Pull request.

For general questions, post on the conversation tab of your Feedback PR with your request.

For specific questions, reply to a specifc comment.

If you think we missed *where* you did something, add a comment on that line (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 requestion and tag @brownsarahm

Support

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. Appointments and locations will be visible in the TutorTrac system on September 14th, 2020. 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-minute, 1 to 1 appointments that students can schedule on Starfish with Dr. David Hayes to address individual academic 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.

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

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 regarded as assurance that the work is the result of the student's own independent thought and study. Work 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
- Facilitating or aiding another's academic dishonesty
- Submitting the same paper for more than one course without prior approval from the instructors

URI COVID-19 Statement

The University is committed to delivering its educational mission while protecting the health and safety of our community. While the university has worked to create a healthy learning environment for all, it is up to all of us to ensure our campus stays that way.

As members of the URI community, students are required to comply with standards of conduct and take precautions to keep themselves and others safe. Visit web.uri.edu/coronavirus/ for the latest information about the URI COVID-19 response.

- Universal indoor masking is required by all community members, on all campuses, regardless of vaccination status. If the universal mask mandate is discontinued during the semester, students who have an approved exemption and are not fully vaccinated will need to continue to wear a mask indoors and maintain physical distance.
- Students who are experiencing symptoms of illness should not come to class. Please stay in your home/room and notify URI Health Services via phone at 401-874-2246.
- If you are already on campus and start to feel ill, go home/back to your room and self-isolate. Notify URI Health Services via phone immediately at 401-874-2246.

If you are unable to attend class, please notify me at brownsarahm@uri.edu. We will work together to ensure that course instruction and work is completed for the semester.

Course Communications

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

Mon	11am-1pm	Tyler 139 and zoom	Kyle
Wed	7-8:30pm	Zoom	Dr. Brown
Fri	3-6pm	Zoom	Kyle

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

To reach out, By usage

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[3], line 2
      1 df = df[['usage','platform','area','note']]
----> 2 display(HTML(df.style.hide()))

File /opt/hostedtoolcache/Python/3.8.16/x64/lib/python3.8/site-packages/IPython/core/display.py:430,
in HTML.__init__(self, data, url, filename, metadata)
    427     suffix = data[-10:].lower()
    428     return prefix.startswith("<iframe ") and suffix.endswith("</iframe>")
--> 430 if warn():
    431     warnings.warn("Consider using IPython.display.IFrame instead")
    432 super(HTML, self).__init__(data=data, url=url, filename=filename, metadata=metadata)

File /opt/hostedtoolcache/Python/3.8.16/x64/lib/python3.8/site-packages/IPython/core/display.py:426,
in HTML.__init__.<locals>.warn()
    420     return False
    422 #
    423 # Avoid calling lower() on the entire data, because it could be a
    424 # long string and we're only interested in its beginning and end.
    425 #
--> 426 prefix = data[:10].lower()
    427 suffix = data[-10:].lower()
    428 return prefix.startswith("<iframe ") and suffix.endswith("</iframe>")

TypeError: 'Styler' object is not subscriptable
```

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.

By Platform

```
/tmp/ipykernel_2084/2135006347.py:3: FutureWarning: this method is deprecated in favour of
`Styler.hide(axis="index")`
  display(HTML(data.drop(columns='platform').style.hide_index()._repr_html_()))
```

usage	area	note
matters that don't fit into another category	to brownsarahm@uri.edu	remember to include `[CSC310]` or `[DSP310]` (note `verbatim` no space)

Use github for


```
/tmp/ipykernel_2084/2135006347.py:3: FutureWarning: this method is deprecated in favour of
`Styler.hide(axis="index")`
  display(HTML(data.drop(columns='platform').style.hide_index()._repr_html_()))
```


usage	area	note
private questions to your assignment	issue on assignment repo	eg bugs in your code"
for general questions that can help others	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	discussion on community repo	include links in your portfolio

Use prismia for


```
/tmp/ipykernel_2084/2135006347.py:3: FutureWarning: this method is deprecated in favour of
`Styler.hide(axis="index")`
  display(HTML(data.drop(columns='platform').style.hide_index()._repr_html_()))
```


usage	area	note
in class	chat	outside of class time this is not monitored closely
any time	download transcript	use after class to get preliminary notes eg if you miss a class

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

- 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.

> Whether you use CSC or DSP does not matter.

1. Welcome and Introduction

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. How this class will work

Participatory Live Coding

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

Debugging is both technical and a soft skill

1.3. 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.

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

> **Warning**
>
> 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.

So programming for data science is more like *writing* it has a narrative flow and is made to be seen more than some other programming thaat you may have done.

1.4. Jupyter Notebooks

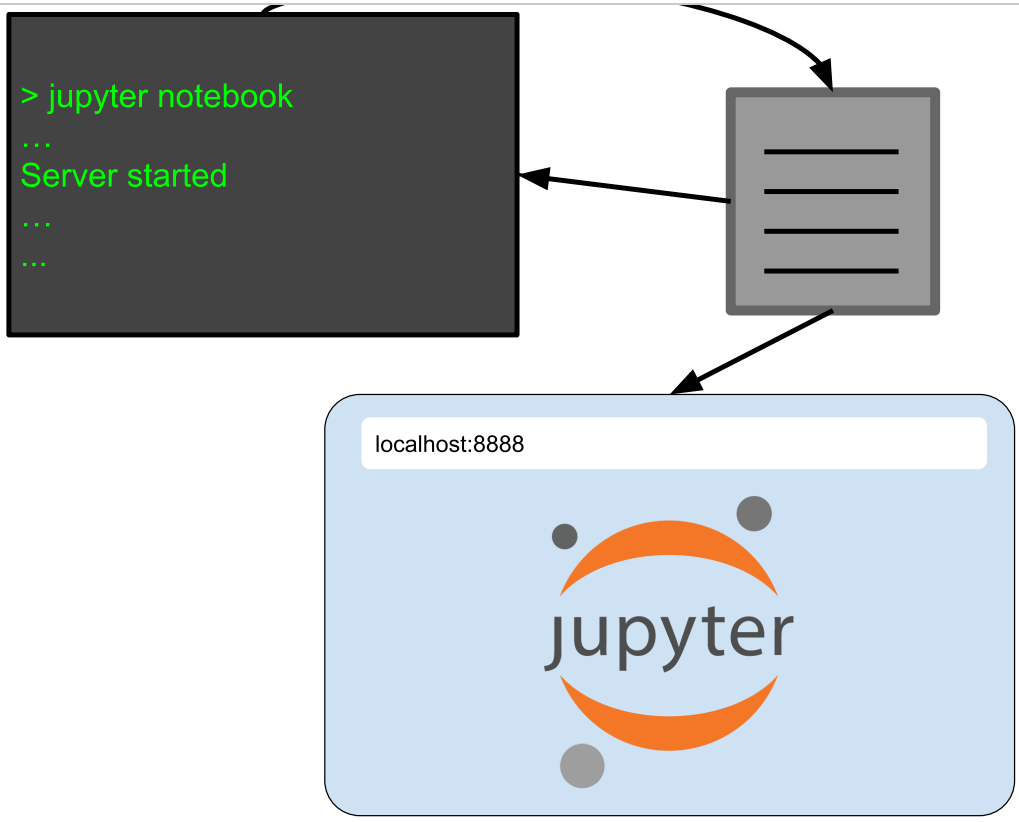
Launch a `jupyter notebook` server:

- on Windows, use anaconda terminal
- on Mac/Linux, use terminal

```
cd path/to/where/you/save/notes
jupyter notebook
```

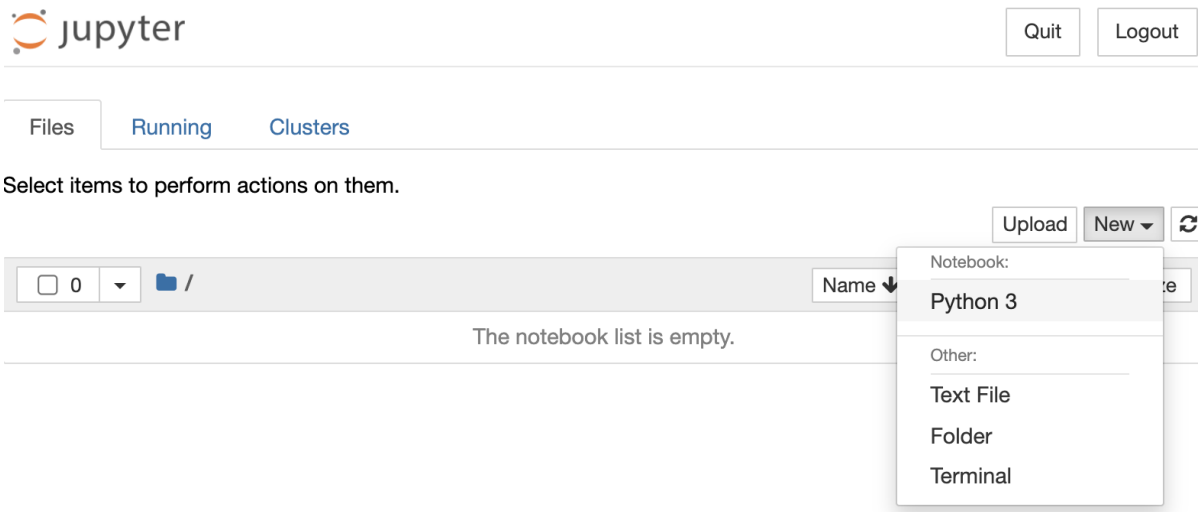
1.4.1. What just happened?

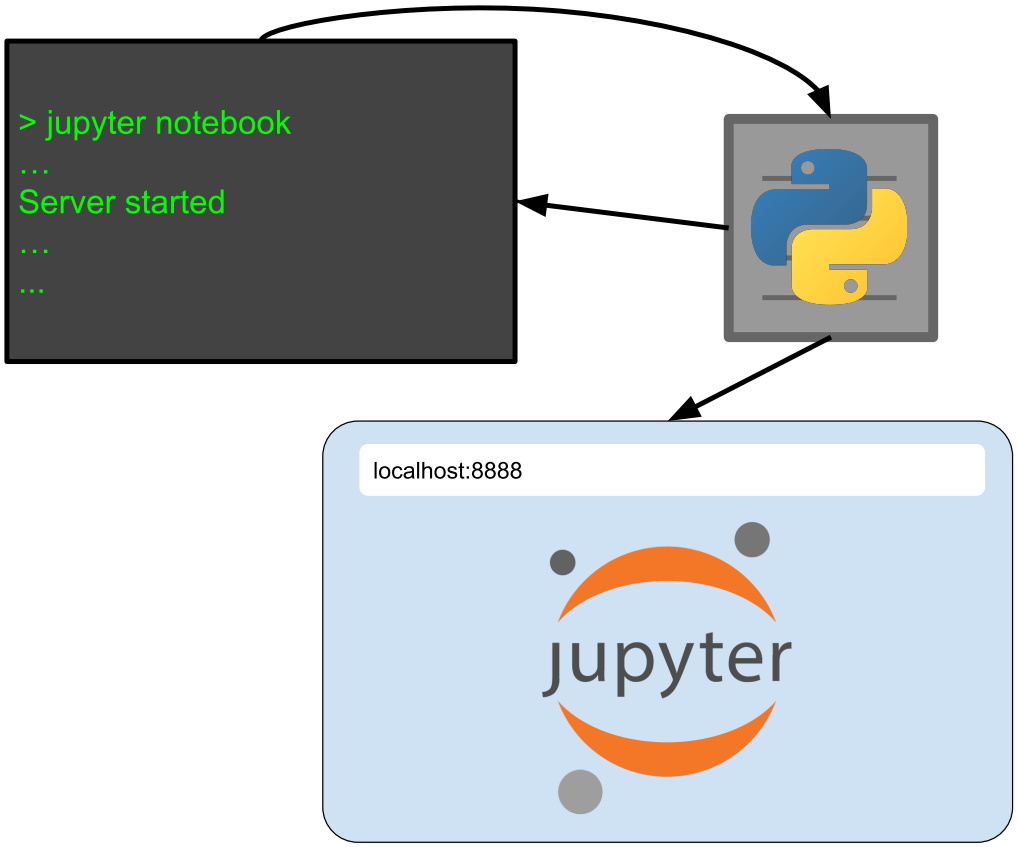
- launched a local web server
- opened a new browser tab pointed to it



1.4.2. Start a Notebook

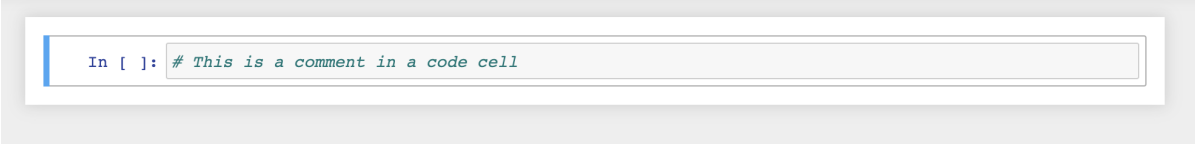
Go to the new menu in the top right and choose Python 3



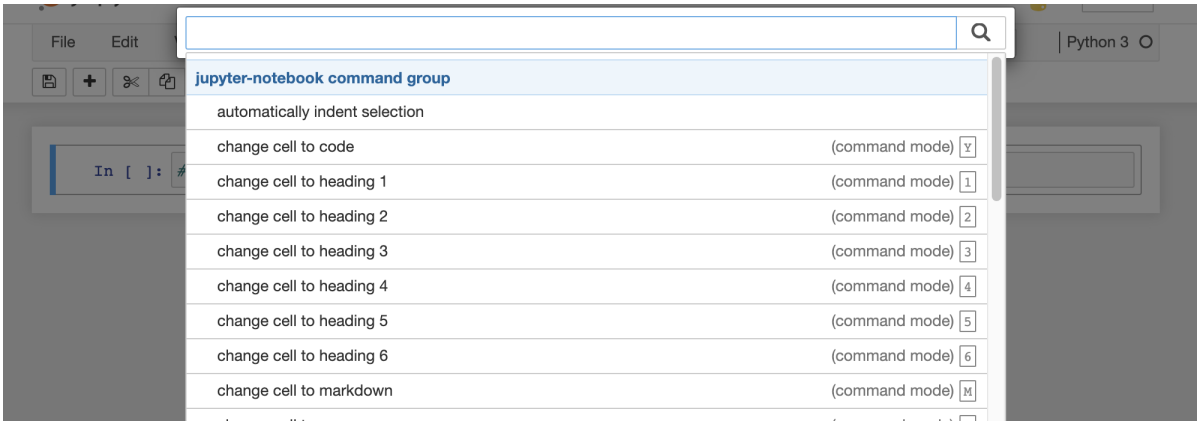


1.4.3. A jupyter notebook tour

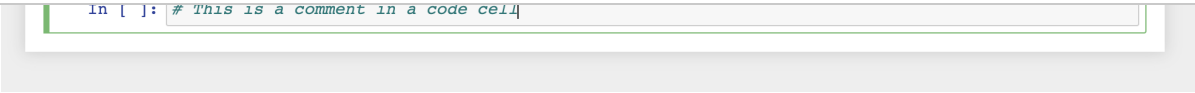
A Jupyter notebook has two modes. When you first open, it is in command mode. The border is blue in command mode.



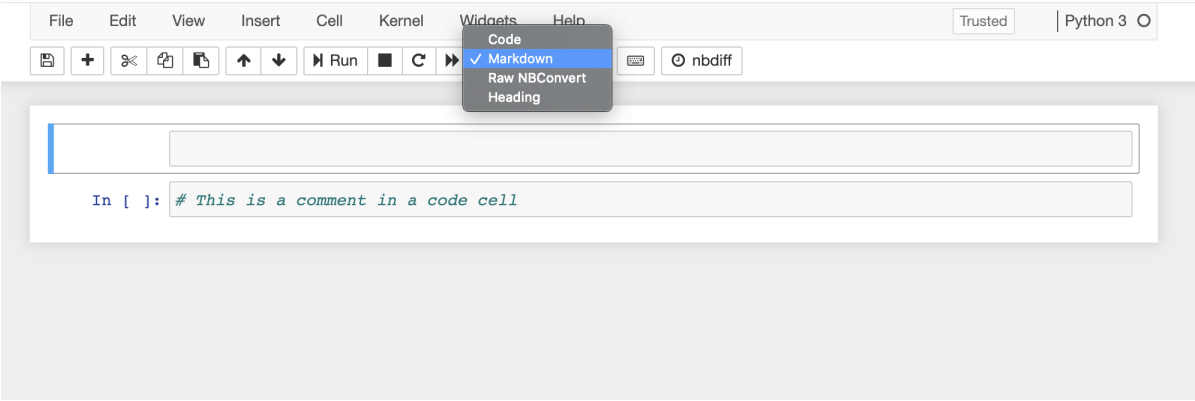
When you press a key in command mode it works like a shortcut. For example `p` shows the command search menu.



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. The border is green in 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

```
# this is a comment in a code cell , we can run this to see output
3+4
```

```
7
```

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

We set variables

```
name = 'sarah'
```

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

```
name
```

```
'sarah'
```

we can put a variable there tosee it

```
course = 'csc310'
course
name
```

Note that this version doesn't show use the value for `course`

```
name = 'Sarah'
```

Important

In class, we ran these cells out of order and noticed how the value does not update unless we run the new version

```
name
```

```
'Sarah'
```

```
course
```

```
'csc310'
```

1.4.4. Notebook Reminders

Blue border is command mode, green border is edit mode

use Escape to get to command mode

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

In code cells, we can use a python interpreter, for example as a calculator.

```
4+6
```

```
10
```

It prints out the last line of code that it ran, even though it executes all of them

```
name = 'sarah'
4+5
name *3
```

```
'sarahsarahsarah'
```

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.

We can print the docstring out, as a whole instead of using the shfit + tab to view it.

```
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 `value, ..., sep`

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.

```
print(name)
```

```
sarah
```

How do you use the `print` function to output: `Sarah_csc310`?

```
print(name,course,sep='_')
```

```
sarah_csc310
```

```
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.
```

We can put as many values as we want there. Thats what the `...` in the function signature means

```
print(name,course,'hello','bye',sep='_')
```

```
sarah_csc310_hello_bye
```

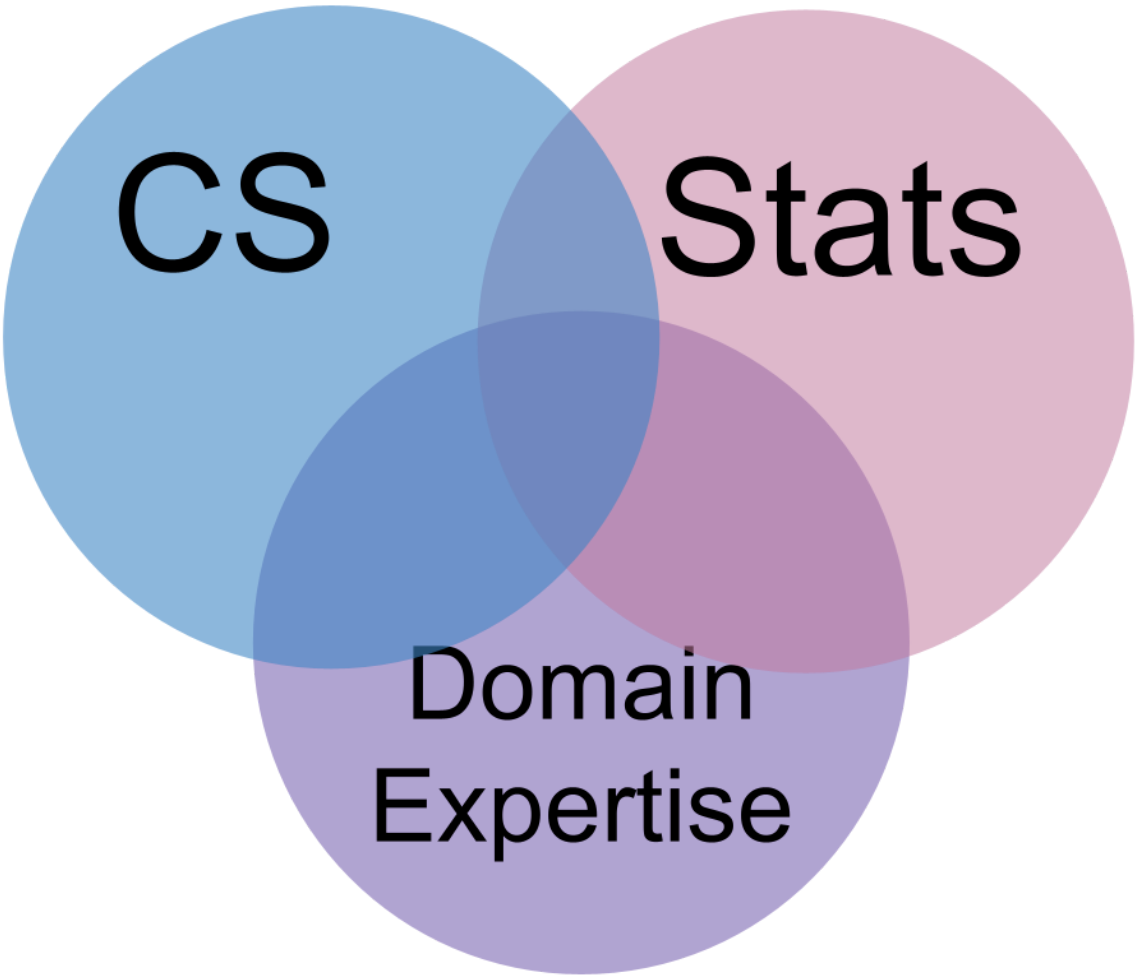
```
sarah
csc310
hello
bye
```

Important

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

1.6. What is Data Science?

Data Science is the combination of

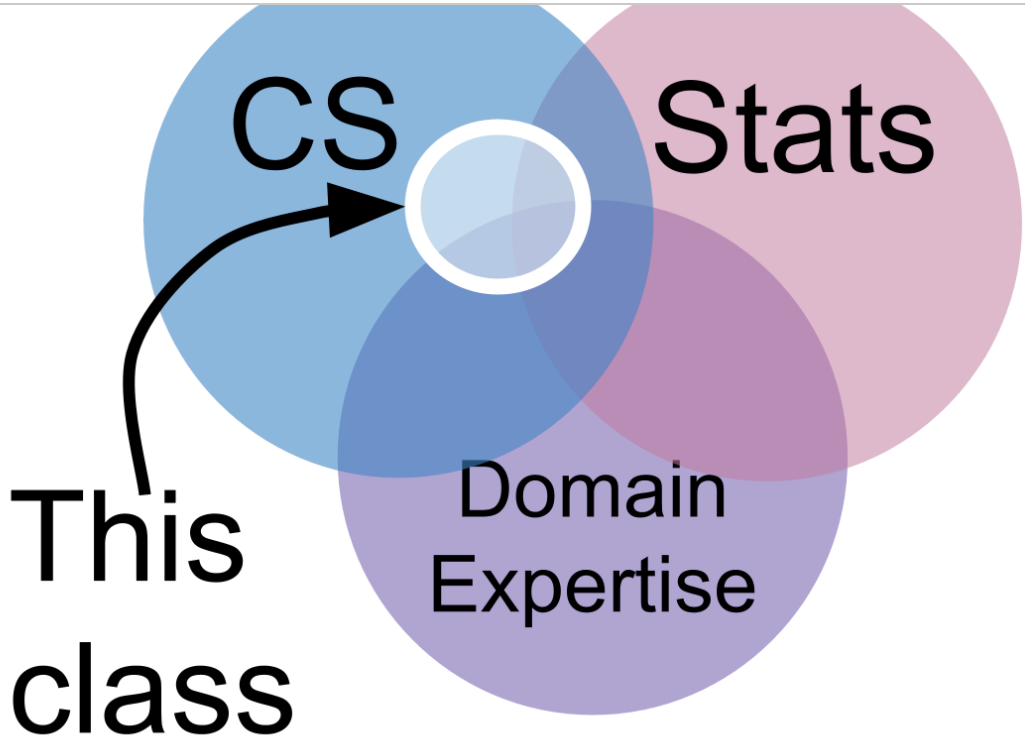


statistics is the type of math we use to make sense of data. Formally, a statistic is just a function of data.

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.

1.6.1. In this class,

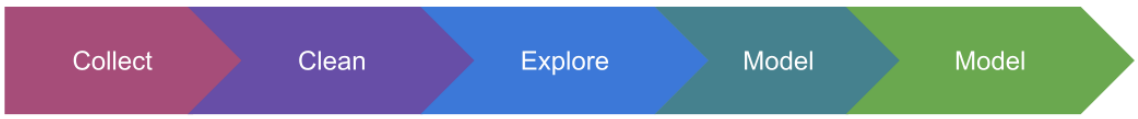


We'll focus on the programming as our main means of studying data science, but we will use bits of the other parts. In particular, you're encouraged to choose datasets that you have domain expertise about, or that you want to learn about.

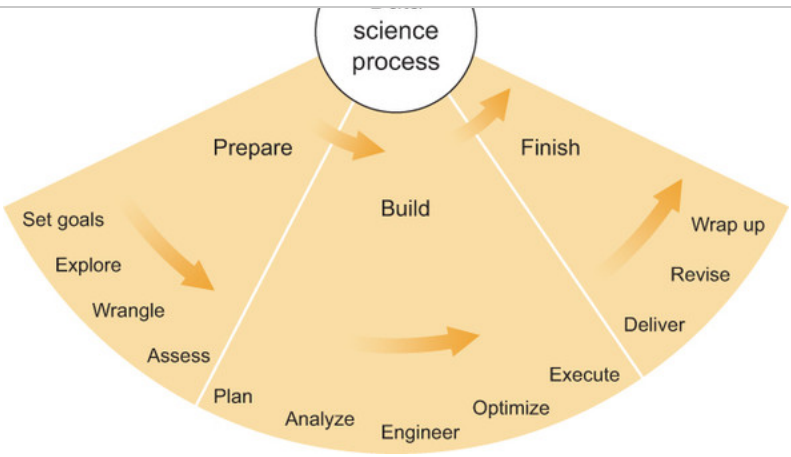
But there are many definitions. We'll use this one, but you may come across others.

1.6.2. How does data science happen?

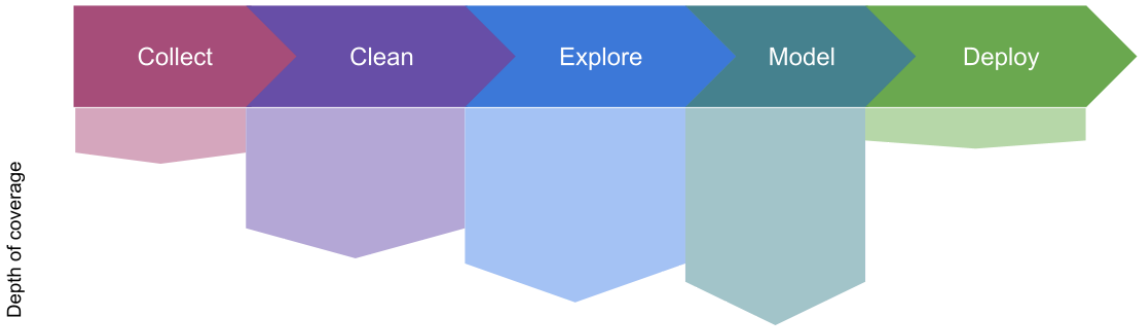
The most common way to think about what doing data science means is to think of this pipeline. It is in the perspective of the data, these are all of the things that happen to the data.



Another way to think about it

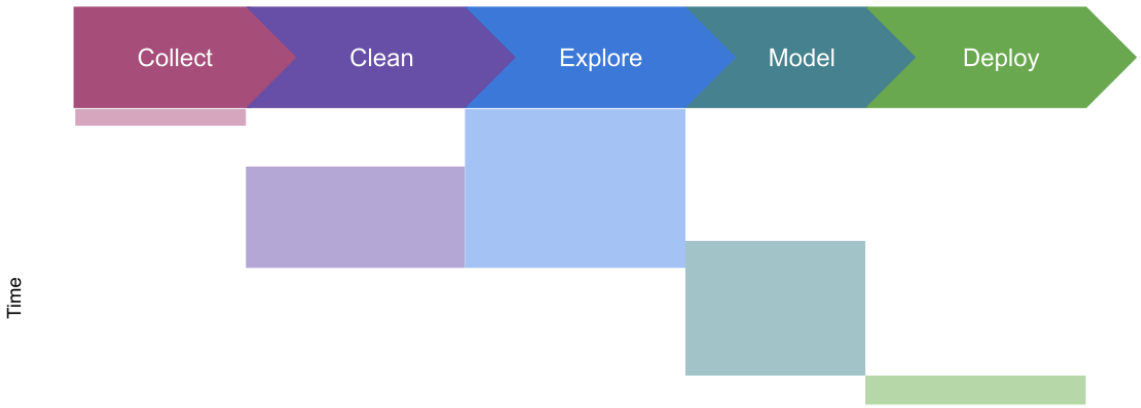


1.6.3. how we'll cover Data Science, in depth



- *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
- *deploy*: A little bit at the end, but a lot of preparation for decision making around deployment

1.6.4. 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.

- Read carefully the syllabus section of the course website
- skim the rest of the course website
- Bring questions about how the class will work to class on Thursday.
- Review Git & GitHub Fundamentals
- Bring git/github questions on Thursday.
- Begin reading chapter 1 of think like a data scientist (finish in time for it to help you with the assignment due Monday night)

On Thursday we will start with a review of the syllabus. You will answer an ungraded quiz to confirm that you understand and I'll answer all of your questions. Then we will do a little bit with Git/GitHub and start your first assignment in class.

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.

Warning

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.

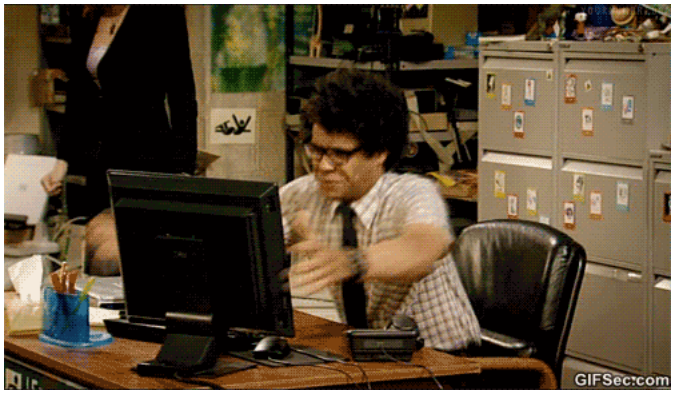
Tip

In chapter 1, focus most on sections 1.1, 1.3, and 1.7.

2. Syllabus and Python Review

2.1. Course Logistics

Class is designed to avoid this:



> 🔎Julia Evans🔍
> @b0rk · Follow
>
> we think about debugging as a technical skill (and it absolutely is!!) but a huge amount of it is managing your feelings so you don't get discouraged and being self-aware so you can recognize your incorrect assumptions
>
> 5:35 PM · Jun 11, 2021
>
> 4.1K Reply Copy link
>
> Read 81 replies

Read more about how I'm designing this course to help you learn on the how to learn page.

It's easy when reading something long to lose track of it. Your eyes can go over each word, without actually retaining the information, but it's important to understand the syllabus for the course.

You can find the answers to the following questions on the syllabus. If you've already read it, try answering them to check your understanding. If you haven't read it yet, use these to guide you to get familiar with finding key facts about the course on the syllabus.

1. What do you need to bring to class each day?
2. What is the basis of grading for this course?
3. How do you reference the course text?
4. What is the penalty for missing an assignment?

More information about the course is available throughout the site, the next few questions will help you self-check that you've found the important things. Remember, the goal is not necessarily to memorize all of this, but to be able to find it.

1. When & what are you expected to read for this class?

- [ ] read the text book before class
- [ ] review notes & documentation after class
- [ ] preview the notes & documentation before class
- [ ] read documentation and text book after class

1. Your assignment says to find a dataset that has variables of a specific type, which website can you use?
2. Your assignment says to find a dataset of any type about something you're interested in, which resource would you use?

2.3. Python Review

Official source on python:

- pep8 official style
- documentation note that you can change which version you are using

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

2.3.1. Functions

Syntax of a function in python:

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

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

A few things to note:

- the `def` keywords 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('sarah')
```

```
'hi sarah'
```

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

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

2.3.2. Conditionals

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

    Parameters
    ----------
    name : string
        the name of who to greet
    formal : bool
        if the greeting should be 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

```
greeting2('sarah',True)
```

```
'hello sarah'
```

```
greeting2('sarah',False)
```

```
'hi sarah'
```

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

```
greeting2('sarah')
```

```
help(greeting)
```

```
Help on function greeting in module __main__:

greeting(name)
    say hi to a person

    Parameters
    ----------
    name : string
        the name of who to greet
```

2.4. Questions After Class

2.4.1. Why is indentation important in python but not other languages like C++?

Python is a newer language than C and C++. Older languages had to contend with the fact that a space character uses the same amount of memory as any other character, so they were not used. However, whitespace is easy to read.

Python was started in 1989, compared to C in 1972, C++ was started in 1985ish, but stuck with a lot of things from C, so tht 1972 is strongly operative.

Python is designed to be easy. It is designed to make complex tasks easier to do. C is designed to be efficient and to compeile well, even if it is hard to learn and do.

2.4.2. Why is python so much slower as well?

Python is slower because it is an interpreted language. That means another program called the Python interpreter (which mostly are written in C) is actually running on your computer, that program parses the text of your source code and then executes the code. The interpreter cannot look ahead and change things in how you wrote your code while it runs.

In contrast, C++ (like C) is a compiled language. This means that a program called a compiler parses your code and translates it into assembly then to machine code. During this process it can optimize your code to make sure that it is fast.

2.4.3. Are portfolios simply whatever we submit, such as assignments, to Github or are there other things that need to be submitted to the portfolios for level 3?

Assignments are separate from the portfolio checks. It will become more clear what to do in your portfolio after you get feedback on assignment 2, and start working on assignment 3.

2.4.4. Will we know the specific criteria to fulfill a level 3 achievement when doing the portfolio?

The evaluation criteria are already listed on the Detailed Checklists.

2.4.5. when is the Assignment 1 due?

Monday, end of day. See the details: Assignment 1: Portfolio Setup, Data Science, and Python

2.4.6. how many large scale programs are we going to write in jupyter?

We won't actually write large scale programs, per se, but we will write some long analyses.

3. Grading review, Pandas, and Iterables

here is my solution

```
def compute_grade(num_level1,num_level2,num_level3):
    '''
    Computes a grade for CSC/DSP310 from numbers of achievements at each level

    Parameters:
    ------------
    num_level1 : int
      number of level 1 achievements earned
    num_level2 : int
      number of level 2 achievements earned
    num_level3 : int
      number of level 3 achievements earned

    Returns:
    --------
    letter_grade : string
      letter grade with modifier (+/-)
    '''
    if num_level1 == 15:
        if num_level2 == 15:
            if num_level3 == 15:
                grade = 'A'
            elif num_level3 >= 10:
                grade = 'A-'
            elif num_level3 >=5:
                grade = 'B+'
            else:
                grade = 'B'
        elif num_level2 >=10:
            grade = 'B-'
        elif num_level2 >=5:
            grade = 'C+'
        else:
            grade = 'C'
    elif num_level1 >= 10:
        grade = 'C-'
    elif num_level1 >= 5:
        grade = 'D+'
    elif num_level1 >=3:
        grade = 'D'
    else:
        grade = 'F'


    return grade
```

Note that we can verify it works using `assert`

```
assert compute_grade(15,15,15) =='A'
```

```
assert compute_grade(15,15,9) =='B+'
```

this also means we can assign the value out

```
my_grade = compute_grade(15,11,5)
```

```
my_grade
```

```
'B-'
```

Alternatively if we use a side effect instead, printing the value instead of returning it.

```
    Computes a grade for CSC/DSP310 from numbers of achievements at each level

    Parameters:
    ------------
    num_level1 : int
      number of level 1 achievements earned
    num_level2 : int
      number of level 2 achievements earned
    num_level3 : int
      number of level 3 achievements earned

    Returns:
    --------
    letter_grade : string
      letter grade with modifier (+/-)
    '''
    if num_level1 == 15:
        if num_level2 == 15:
            if num_level3 == 15:
                grade = 'A'
            elif num_level3 >= 10:
                grade = 'A-'
            elif num_level3 >=5:
                grade = 'B+'
            else:
                grade = 'B'
        elif num_level2 >=10:
            grade = 'B-'
        elif num_level2 >=5:
            grade = 'C+'
        else:
            grade = 'C'
    elif num_level1 >= 10:
        grade = 'C-'
    elif num_level1 >= 5:
        grade = 'D+'
    elif num_level1 >=3:
        grade = 'D'
    else:
        grade = 'F'


    print( grade)
```

Look this way it looks similar:

```
compute_grade_sideeffect(15,15,15)
```

```
A
```

```
compute_grade(15,15,15)
```

```
'A'
```

and python lets us assign something

```
my_grade = compute_grade_sideeffect(15,15,15)
```

```
A
```

but the output is nothing

```
type(my_grade)
```

```
NoneType
```

First we learned about the dataset then we can load it.

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

We will use pandas.

```
import pandas as pd
```

load the data

```
coffee_df = pd.read_csv(coffee_data_url,index_col=0)
```

```
type(coffee_df)
```

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

```
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')
```

```
coffee_df.columns[0]
```

```
'Species'
```

```
len(coffee_df.columns)
```

```
43
```

```
coffee_df.shape
```

```
(28, 43)
```

```
coffee_df.head()
```

1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	ankole coffee producers	0	ankc coff produce co
2	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethuraman estate	14/1148/2017/21	kaapi roya
3	Robusta	andrew hetzel	India	sethuraman estate	NaN	NaN	0000	sethuram esta
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 developme trust

5 rows × 43 columns

```
coffee_df.head
```

```
1   Robusta       ankole coffee producers coop            Uganda
2   Robusta                   nishant gurjer             India
3   Robusta                    andrew hetzel             India
4   Robusta                           ugacof            Uganda
5   Robusta      katuka development trust ltd            Uganda
6   Robusta                    andrew hetzel             India
7   Robusta                    andrew hetzel             India
8   Robusta                   nishant gurjer             India
9   Robusta                   nishant gurjer             India
10  Robusta                           ugacof            Uganda
11  Robusta                           ugacof            Uganda
12  Robusta                   nishant gurjer             India
13  Robusta                    andrew hetzel             India
14  Robusta  kasozi coffee farmers association            Uganda
15  Robusta      ankole coffee producers coop            Uganda
16  Robusta                    andrew hetzel             India
17  Robusta                    andrew hetzel             India
18  Robusta                kawacom uganda ltd            Uganda
19  Robusta                     nitubaasa ltd            Uganda
20  Robusta             mannya coffee project            Uganda
21  Robusta                    andrew hetzel             India
22  Robusta                    andrew hetzel             India
23  Robusta                    andrew hetzel     United States
24  Robusta                      luis robles           Ecuador
25  Robusta                      luis robles           Ecuador
26  Robusta                      james moore     United States
27  Robusta                    cafe politico             India
28  Robusta                    cafe politico           Vietnam

                         Farm.Name Lot.Number  \
1       kyangundu cooperative society        NaN
2      sethuraman estate kaapi royale         25
3                 sethuraman estate        NaN
4                ugacof project area        NaN
5   katikamu capca farmers association        NaN
6                                NaN        NaN
7                sethuraman estates        NaN
8      sethuraman estate kaapi royale          7
9                 sethuraman estate        RKR
10                            ishaka        NaN
11               ugacof project area        NaN
12     sethuraman estate kaapi royale      RC AB
13               sethuraman estates        NaN
14             kasozi coffee farmers        NaN
15            kyangundu coop society        NaN
16                sethuraman estate        NaN
17               sethuraman estates        NaN
18                         bushenyi        NaN
19   kigezi coffee farmers association        NaN
20             mannya coffee project        NaN
21                sethuraman estates        NaN
22                sethuraman estates        NaN
23                sethuraman estates        NaN
24                         robustasa   Lavado 1
25                         robustasa   Lavado 3
26                   fazenda cazengo        NaN
27                               NaN        NaN
28                               NaN        NaN

                                   Mill         ICO.Number  \
1                 ankole coffee producers                  0
2                       sethuraman estate    14/1148/2017/21
3                                    NaN               0000
4                                 ugacof                  0
5                katuka development trust                  0
6                                 (self)                NaN
7                                    NaN                NaN
8                       sethuraman estate    14/1148/2017/18
9                       sethuraman estate    14/1148/2016/17
10                           nsubuga umar                  0
11                                ugacof                  0
12                      sethuraman estate    14/1148/2016/12
13                                   NaN                NaN
14                                   NaN                  0
15  ankole coffee producers coop union ltd                  0
16                                   NaN               0000
17                      sethuraman estates                NaN
18                                kawacom                  0
19                               nitubaasa                 0
20                  mannya coffee project                  0
21                                   NaN                NaN
22                      sethuraman estates                NaN
23                      sethuraman estates                NaN
24                              our own lab               NaN
25                          own laboratory                NaN
```

```
28                                    NaN                  NaN

                           Company    Altitude  \
1       ankole coffee producers coop        1488
2                     kaapi royale        3170
3                sethuraman estate       1000m
4                     ugacof ltd         1212
5       katuka development trust ltd   1200-1300
6                   cafemakers, llc       3000'
7                        cafemakers        750m
8                     kaapi royale        3140
9                     kaapi royale        1000
10                      ugacof ltd    900-1300
11                      ugacof ltd        1095
12                    kaapi royale        1000
13                      cafemakers        750m
14  kasozi coffee farmers association        1367
15      ankole coffee producers coop        1488
16               sethuraman estate       1000m
17                 cafemakers, llc        750m
18              kawacom uganda ltd        1600
19                 nitubaasa ltd         1745
20           mannya coffee project         1200
21                      cafemakers        750m
22                 cafemakers, llc        750m
23                 cafemakers, llc       3000'
24                       robustasa         NaN
25                       robustasa          40
26           global opportunity fund  795 meters
27                    cafe politico         NaN
28                    cafe politico         NaN

                         Region  ...         Color  Category.Two.Defects  \
1           sheema south western  ...         Green                     2
2     chikmagalur karnataka indua  ...           NaN                     2
3                   chikmagalur  ...         Green                     0
4                       central  ...         Green                     7
5          luwero central region  ...         Green                     3
6                   chikmagalur  ...         Green                     0
7                   chikmagalur  ...         Green                     0
8     chikmagalur karnataka india  ...  Bluish-Green                     0
9          chikmagalur karnataka  ...         Green                     0
10                      western  ...         Green                     6
11      iganga namadrope eastern  ...         Green                     1
12         chikmagalur karnataka  ...         Green                     0
13                  chikmagalur  ...         Green                     1
14                      eastern  ...         Green                     7
15                south western  ...         Green                     2
16                  chikmagalur  ...         Green                     0
17                  chikmagalur  ...    Blue-Green                     0
18                      western  ...         Green                     1
19                      western  ...         Green                     2
20                     southern  ...         Green                     1
21                  chikmagalur  ...  Bluish-Green                     1
22                  chikmagalur  ...         Green                     0
23                  chikmagalur  ...         Green                     0
24              san juan, playas  ...    Blue-Green                     1
25              san juan, playas  ...    Blue-Green                     0
26  kwanza norte province, angola  ...           NaN                     6
27                          NaN  ...         Green                     1
28                          NaN  ...          None                     9

             Expiration                  Certification.Body  \
1       June 26th, 2015  Uganda Coffee Development Authority
2    October 31st, 2018         Specialty Coffee Association
3      April 29th, 2016         Specialty Coffee Association
4       July 14th, 2015  Uganda Coffee Development Authority
5       June 26th, 2015  Uganda Coffee Development Authority
6   February 28th, 2013         Specialty Coffee Association
7        May 15th, 2015         Specialty Coffee Association
8    October 25th, 2018         Specialty Coffee Association
9     August 17th, 2017         Specialty Coffee Association
10     August 5th, 2015  Uganda Coffee Development Authority
11      June 26th, 2015  Uganda Coffee Development Authority
12    August 23rd, 2017         Specialty Coffee Association
13       May 19th, 2015         Specialty Coffee Association
14      July 14th, 2015  Uganda Coffee Development Authority
15      July 14th, 2015  Uganda Coffee Development Authority
16     April 29th, 2016         Specialty Coffee Association
17       June 3rd, 2014         Specialty Coffee Association
18      June 27th, 2015  Uganda Coffee Development Authority
19      June 27th, 2015  Uganda Coffee Development Authority
20      June 27th, 2015  Uganda Coffee Development Authority
21       May 19th, 2015         Specialty Coffee Association
22      June 20th, 2014         Specialty Coffee Association
```

```
25   January 15th, 2017         Specialty Coffee Association
26  December 23rd, 2015         Specialty Coffee Association
27    August 25th, 2015         Specialty Coffee Association
28    August 25th, 2015         Specialty Coffee Association

                       Certification.Address  \
1   e36d0270932c3b657e96b7b0278dfd85dc0fe743
2   ff7c18ad303d4b603ac3f8cff7e611ffc735e720
3   ff7c18ad303d4b603ac3f8cff7e611ffc735e720
4   e36d0270932c3b657e96b7b0278dfd85dc0fe743
5   e36d0270932c3b657e96b7b0278dfd85dc0fe743
6   ff7c18ad303d4b603ac3f8cff7e611ffc735e720
7   ff7c18ad303d4b603ac3f8cff7e611ffc735e720
8   ff7c18ad303d4b603ac3f8cff7e611ffc735e720
9   ff7c18ad303d4b603ac3f8cff7e611ffc735e720
10  e36d0270932c3b657e96b7b0278dfd85dc0fe743
11  e36d0270932c3b657e96b7b0278dfd85dc0fe743
12  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
13  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
14  e36d0270932c3b657e96b7b0278dfd85dc0fe743
15  e36d0270932c3b657e96b7b0278dfd85dc0fe743
16  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
17  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
18  e36d0270932c3b657e96b7b0278dfd85dc0fe743
19  e36d0270932c3b657e96b7b0278dfd85dc0fe743
20  e36d0270932c3b657e96b7b0278dfd85dc0fe743
21  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
22  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
23  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
24  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
25  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
26  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
27  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
28  ff7c18ad303d4b603ac3f8cff7e611ffc735e720

                       Certification.Contact unit_of_measurement  \
1   03077a1c6bac60e6f514691634a7f6eb5c85aae8                    m
2   352d0cf7f3e9be14dad7df644ad65efc27605ae2                    m
3   352d0cf7f3e9be14dad7df644ad65efc27605ae2                    m
4   03077a1c6bac60e6f514691634a7f6eb5c85aae8                    m
5   03077a1c6bac60e6f514691634a7f6eb5c85aae8                    m
6   352d0cf7f3e9be14dad7df644ad65efc27605ae2                    m
7   352d0cf7f3e9be14dad7df644ad65efc27605ae2                    m
8   352d0cf7f3e9be14dad7df644ad65efc27605ae2                    m
9   352d0cf7f3e9be14dad7df644ad65efc27605ae2                    m
10  03077a1c6bac60e6f514691634a7f6eb5c85aae8                    m
11  03077a1c6bac60e6f514691634a7f6eb5c85aae8                    m
12  352d0cf7f3e9be14dad7df644ad65efc27605ae2                    m
13  352d0cf7f3e9be14dad7df644ad65efc27605ae2                    m
14  03077a1c6bac60e6f514691634a7f6eb5c85aae8                    m
15  03077a1c6bac60e6f514691634a7f6eb5c85aae8                    m
16  352d0cf7f3e9be14dad7df644ad65efc27605ae2                    m
17  352d0cf7f3e9be14dad7df644ad65efc27605ae2                    m
18  03077a1c6bac60e6f514691634a7f6eb5c85aae8                    m
19  03077a1c6bac60e6f514691634a7f6eb5c85aae8                    m
20  03077a1c6bac60e6f514691634a7f6eb5c85aae8                    m
21  352d0cf7f3e9be14dad7df644ad65efc27605ae2                    m
22  352d0cf7f3e9be14dad7df644ad65efc27605ae2                    m
23  352d0cf7f3e9be14dad7df644ad65efc27605ae2                    m
24  352d0cf7f3e9be14dad7df644ad65efc27605ae2                    m
25  352d0cf7f3e9be14dad7df644ad65efc27605ae2                    m
26  352d0cf7f3e9be14dad7df644ad65efc27605ae2                    m
27  352d0cf7f3e9be14dad7df644ad65efc27605ae2                    m
28  352d0cf7f3e9be14dad7df644ad65efc27605ae2                    m

   altitude_low_meters altitude_high_meters  altitude_mean_meters
1               1488.0               1488.0                1488.0
2               3170.0               3170.0                3170.0
3               1000.0               1000.0                1000.0
4               1212.0               1212.0                1212.0
5               1200.0               1300.0                1250.0
6               3000.0               3000.0                3000.0
7                750.0                750.0                 750.0
8               3140.0               3140.0                3140.0
9               1000.0               1000.0                1000.0
10               900.0               1300.0                1100.0
11              1095.0               1095.0                1095.0
12              1000.0               1000.0                1000.0
13               750.0                750.0                 750.0
14              1367.0               1367.0                1367.0
15              1488.0               1488.0                1488.0
16              1000.0               1000.0                1000.0
17               750.0                750.0                 750.0
18              1600.0               1600.0                1600.0
19              1745.0               1745.0                1745.0
```

```
22              750.0           750.0           750.0
23             3000.0          3000.0          3000.0
24                NaN             NaN             NaN
25               40.0            40.0            40.0
26              795.0           795.0           795.0
27                NaN             NaN             NaN
28                NaN             NaN             NaN

[28 rows x 43 columns]>
```

```
coffee_df.head(2)
```

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	A
1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	ankole coffee producers	0	ankole coffee producers coop	
2	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethuraman estate	14/1148/2017/21	kaapi royale	

2 rows × 43 columns

```
coffee_df.tail()
```

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude
24	Robusta	luis robles	Ecuador	robustasa	Lavado 1	our own lab	NaN	robustasa	NaN
25	Robusta	luis robles	Ecuador	robustasa	Lavado 3	own laboratory	NaN	robustasa	40
26	Robusta	james moore	United States	fazenda cazengo	NaN	cafe cazengo	NaN	global opportunity fund	795 meters
27	Robusta	cafe politico	India	NaN	NaN	NaN	14-1118-2014-0087	cafe politico	NaN
28	Robusta	cafe politico	Vietnam	NaN	NaN	NaN	NaN	cafe politico	NaN

5 rows × 43 columns

```
coffee_df.info()
```

```
Int64Index: 28 entries, 1 to 28
Data columns (total 43 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Species                28 non-null     object
 1   Owner                  28 non-null     object
 2   Country.of.Origin      28 non-null     object
 3   Farm.Name              25 non-null     object
 4   Lot.Number             6 non-null      object
 5   Mill                   20 non-null     object
 6   ICO.Number             17 non-null     object
 7   Company                28 non-null     object
 8   Altitude               25 non-null     object
 9   Region                 26 non-null     object
 10  Producer               26 non-null     object
 11  Number.of.Bags         28 non-null     int64
 12  Bag.Weight             28 non-null     object
 13  In.Country.Partner     28 non-null     object
 14  Harvest.Year           28 non-null     int64
 15  Grading.Date           28 non-null     object
 16  Owner.1                28 non-null     object
 17  Variety                3 non-null      object
 18  Processing.Method      10 non-null     object
 19  Fragrance...Aroma      28 non-null     float64
 20  Flavor                 28 non-null     float64
 21  Aftertaste             28 non-null     float64
 22  Salt...Acid            28 non-null     float64
 23  Bitter...Sweet         28 non-null     float64
 24  Mouthfeel              28 non-null     float64
 25  Uniform.Cup            28 non-null     float64
 26  Clean.Cup              28 non-null     float64
 27  Balance                28 non-null     float64
 28  Cupper.Points          28 non-null     float64
 29  Total.Cup.Points       28 non-null     float64
 30  Moisture               28 non-null     float64
 31  Category.One.Defects   28 non-null     int64
 32  Quakers                28 non-null     int64
 33  Color                  26 non-null     object
 34  Category.Two.Defects   28 non-null     int64
 35  Expiration             28 non-null     object
 36  Certification.Body     28 non-null     object
 37  Certification.Address  28 non-null     object
 38  Certification.Contact  28 non-null     object
 39  unit_of_measurement    28 non-null     object
 40  altitude_low_meters    25 non-null     float64
 41  altitude_high_meters   25 non-null     float64
 42  altitude_mean_meters   25 non-null     float64
dtypes: float64(15), int64(5), object(23)
memory usage: 9.6+ KB
```

```
coffee_df.columns[42]
```

```
'altitude_mean_meters'
```

```
col_types = coffee_df.dtypes
```

```
col_types[:5]
```

```
Species              object
Owner                object
Country.of.Origin    object
Farm.Name            object
Lot.Number           object
dtype: object
```

```
type(col_types[0])
```

```
numpy.dtype[object_]
```

```
    print(col_name[:3])
```

```
Spe
Own
Cou
Far
Lot
Mil
ICO
Com
Alt
Reg
Pro
Num
Bag
In.
Har
Gra
Own
Var
Pro
Fra
Fla
Aft
Sal
Bit
Mou
Uni
Cle
Bal
Cup
Tot
Moi
Cat
Qua
Col
Cat
Exp
Cer
Cer
Cer
uni
alt
alt
alt
```

```
my_list = ['honda','ford','nissan']
```

```
type(my_list)
```

```
list
```

```
my_list[-1]
```

```
'nissan'
```

```
short_names = [col_name[:3] for col_name in coffee_df.columns]
```

```
type(short_names)
```

```
list
```

```
short_names
```

```
 'Cou',
 'Far',
 'Lot',
 'Mil',
 'ICO',
 'Com',
 'Alt',
 'Reg',
 'Pro',
 'Num',
 'Bag',
 'In.',
 'Har',
 'Gra',
 'Own',
 'Var',
 'Pro',
 'Fra',
 'Fla',
 'Aft',
 'Sal',
 'Bit',
 'Mou',
 'Uni',
 'Cle',
 'Bal',
 'Cup',
 'Tot',
 'Moi',
 'Cat',
 'Qua',
 'Col',
 'Cat',
 'Exp',
 'Cer',
 'Cer',
 'Cer',
 'uni',
 'alt',
 'alt',
 'alt']
```

3.3. Questions After Class

3.3.1. • will we be gathering our own data or will it all be provided for the course?

3.3.2. Will you always give us an answer key for assignments?

No, but we will always give personalized feedback.

3.3.3. will we be cleaning data?

Yes, see the notes from the first class.

3.3.4. Will we do correlations later?

Yes

3.3.5. will we go over more imports like pandas?

Yes, we will use other libraries. In A2, you'll even import your own code.

3.3.6. How will datasets be used in conjunction with one another?

We will combine data in a few weeks.

They are both compelte languages so at some level, they can both do all the same things. Some libraries are easier/better for specific things in one language or the other.

3.3.8. Will we be working with databases at all in this class?

A little. We'll pull from a database into python.

3.3.9. Are we going going to be using pandas a lot in this class?

Yes. We will use pandas for almost every single remaining class session this semester.

3.3.10. how do you short-cut fill variable names in jupyter

Press tab to autocomplete

3.3.11. what is the most efficient way to get help on the homework ?

Accept the assignment and make an issue or go to office hours for general questions. For clarifying questions, you can post an issue on the course website.

If you are stuck with some progress made, upload (or push) your code first and then ask for help so we can see where you are.

3.3.12. is there a best or most common way to organize data for use

csv

3.3.13. how would you laod a csv file with python if i was using visual studio instead of jupyter?

All of the code we are writing will work in any python environment that has the libraries. the Editor you use (jupyter vs VSCode vs PyCharm) does not impact what you can do with python. Which interpreter you use can impact and jupyter does default to ipython instead of the core python kernel, but that does not change how to load data.

Remember that jupyter is not on a cloud service. YOu can use any file on your computer with a relative path.

4. Pandas and Indexing

4.1. Iterable types

```
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()
```

```
a
```

```
['a', 'b', 'c', 'd', 'e']
```

```
type(a)
```

```
list
```

```
{'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4}
```

```
type(b)
```

```
dict
```

```
c
```

```
('a', 'b', 'c', 'd', 'e')
```

```
type(c)
```

```
tuple
```

```
d
```

```
['a', 'b', 'c', 'd', 'e']
```

```
type(d)
```

```
list
```

4.2. Reading data other ways

```
import pandas as pd
```

```
course_comms_url = 'https://rhodyprog4ds.github.io/BrownSpring23/syllabus/communication.html'
```

THis reads in from the html directly.

```
pd.read_html(course_comms_url)
```

```
0  Mon  11am-1pm  Tyler 135 and Zoom       Kyle
1  Wed  7-8:30pm               Zoom  Dr. Brown
2  Fri     3-6pm               Zoom       Kyle,
                                      usage                    area  \
0  matters that don't fit into another category  to brownsarahm@uri.edu

                                                    note
0  remember to include `[CSC310]` or `[DSP310]` (...  ,
                                                   usage  \
0              private questions to your assignment
1        for general questions that can help others
2  to share resources or ask general questions in...

                          area  \
0     issue on assignment repo
1      issue on course website
2  discussion on community repo

                                                    note
0                            eg bugs in your code"
1  eg what the instructions of an assignment mean...
2                  include links in your portfolio  ,
      usage                area  \
0  in class                chat
1  any time  download transcript

                                                    note
0  outside of class time this is not monitored cl...
1  use after class to get preliminary notes eg if...  ]
```

```
html_list = pd.read_html(course_comms_url)
```

```
type(html_list)
```

```
list
```

```
type(html_list[0])
```

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

```
[type(h) for h in html_list]
```

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

```
achievements_url = 'https://rhodyprog4ds.github.io/BrownSpring23/syllabus/achievements.html'
```

get the tables

```
achievements_df_list = pd.read_html(achievements_url)
```

make a list means use a list comprehension

```
[ach.shape for ach in achievements_df_list]
```

```
[(14, 3), (15, 5), (15, 15), (15, 6)]
```

```
achievements_df_list.shape
```

```
AttributeError                            Traceback (most recent call last)
Cell In[20], line 1
----> 1 achievements_df_list.shape

AttributeError: 'list' object has no attribute 'shape'
```

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

```
coffee_df = pd.read_csv(coffee_data_url,index_col=0)
```

```
coffee_df.head(1)
```

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitud
1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	ankole coffee producers	0	ankole coffee producers coop	148

1 rows × 43 columns

```
coffee_df['Species'].head()
```

```
1    Robusta
2    Robusta
3    Robusta
4    Robusta
5    Robusta
Name: Species, dtype: object
```

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

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

```
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')
```

```
coffee_df['Number.of.Bags']
```

```
2     320
3     300
4     320
5       1
6     200
7     320
8     320
9     320
10    320
11    320
12    320
13    100
14      1
15    320
16    300
17    140
18      1
19     20
20      6
21    100
22    250
23    100
24      1
25      1
26      1
27      1
28      1
Name: Number.of.Bags, dtype: int64
```

```
new_values = {0:'<100',1:'100-199',2:'200-299',3:'300+'}
```

```
[new_values[int(num/100)] for num in coffee_df['Number.of.Bags']]
```

```
['300+',
 '300+',
 '300+',
 '300+',
 '<100',
 '200-299',
 '300+',
 '300+',
 '300+',
 '300+',
 '300+',
 '300+',
 '100-199',
 '<100',
 '300+',
 '300+',
 '100-199',
 '<100',
 '<100',
 '<100',
 '100-199',
 '200-299',
 '100-199',
 '<100',
 '<100',
 '<100',
 '<100',
 '<100']
```

```
bags_bin = lambda num: int(num/100)
[new_values[bags_bin(num)] for num in coffee_df['Number.of.Bags']]
```

```
 '300+',
 '300+',
 '<100',
 '200-299',
 '300+',
 '300+',
 '300+',
 '300+',
 '300+',
 '300+',
 '100-199',
 '<100',
 '300+',
 '300+',
 '100-199',
 '<100',
 '<100',
 '<100',
 '100-199',
 '200-299',
 '100-199',
 '<100',
 '<100',
 '<100',
 '<100',
 '<100']
```

```
type(pd.read_csv)
```

```
function
```

```
type(bags_bin)
```

```
function
```

4.3. Importing locally

If I make a file in the same folder as my notebook called `example.py` and then put

▶ Show code cell source

```
name = 'sarah'
```

in the file, we can use that file like:

```
from example import name
```

```
name
```

```
'sarah'
```

```
import example
```

```
example.name
```

```
'sarah'
```

4.4.1. why does casting the int over the (num/100) give you the right number? Is it because of floor division?

First let's look at an interim value, lets pick a value for `num`

```
num = 307
```

Then do the calculation without casting to int

```
num/100
```

```
3.07
```

Remember that `int` type is an integer or whole number, no fraction. So, casting drops the decimal part.

4.4.2. How would adding 2 DataFrametogether of separate types affect the type command?

It depends what "add" means. If addition it might error, but if it worked, then it would still be a DataFrame. If stacking with `pd.concat` it would also be a DatFrame.

If you make them into a list, then the would be a list.

4.4.3. what keys to use in the dictionaries?

In the assignment the instruction say

4.4.4. how to save as a local csv file?

pandas.DataFrame.to_csv

4.4.5. how to create a Dataframe?

Use the constructor

4.4.6. how to read using relative path?

A relative path can work just like a URL. read about them here

4.4.7. I would like to know about other common forms of data files.

The pandas documentation's I/O page is where I recommend starting

4.5. What other libraries do we end up using?

Next week we will use `seaborn` for plotting. Later in the semester we will use `sklearn` for machine learning. We will use a few other libaries for a few features, but these three are the main ones.

5. Exploratory Data Analysis (EDA)

Again, we import pandas as usual

and loaded the data in again

```
coffee_data_url = 'https://raw.githubusercontent.com/jldbc/coffee-quality-
database/master/data/robusta_data_cleaned.csv'
coffee_df = pd.read_csv(coffee_data_url,index_col=0)
```

5.1. 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
- Important in how you report your results
- 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

5.2. 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.

We can also get more structural information with the `info` method.

```
coffee_df.info()
```

```
Int64Index: 28 entries, 1 to 28
Data columns (total 43 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Species                28 non-null     object
 1   Owner                  28 non-null     object
 2   Country.of.Origin      28 non-null     object
 3   Farm.Name              25 non-null     object
 4   Lot.Number             6 non-null      object
 5   Mill                   20 non-null     object
 6   ICO.Number             17 non-null     object
 7   Company                28 non-null     object
 8   Altitude               25 non-null     object
 9   Region                 26 non-null     object
 10  Producer               26 non-null     object
 11  Number.of.Bags         28 non-null     int64
 12  Bag.Weight             28 non-null     object
 13  In.Country.Partner     28 non-null     object
 14  Harvest.Year           28 non-null     int64
 15  Grading.Date           28 non-null     object
 16  Owner.1                28 non-null     object
 17  Variety                3 non-null      object
 18  Processing.Method      10 non-null     object
 19  Fragrance...Aroma      28 non-null     float64
 20  Flavor                 28 non-null     float64
 21  Aftertaste             28 non-null     float64
 22  Salt...Acid            28 non-null     float64
 23  Bitter...Sweet         28 non-null     float64
 24  Mouthfeel              28 non-null     float64
 25  Uniform.Cup            28 non-null     float64
 26  Clean.Cup              28 non-null     float64
 27  Balance                28 non-null     float64
 28  Cupper.Points          28 non-null     float64
 29  Total.Cup.Points       28 non-null     float64
 30  Moisture               28 non-null     float64
 31  Category.One.Defects   28 non-null     int64
 32  Quakers                28 non-null     int64
 33  Color                  26 non-null     object
 34  Category.Two.Defects   28 non-null     int64
 35  Expiration             28 non-null     object
 36  Certification.Body     28 non-null     object
 37  Certification.Address  28 non-null     object
 38  Certification.Contact  28 non-null     object
 39  unit_of_measurement    28 non-null     object
 40  altitude_low_meters    25 non-null     float64
 41  altitude_high_meters   25 non-null     float64
 42  altitude_mean_meters   25 non-null     float64
dtypes: float64(15), int64(5), object(23)
memory usage: 9.6+ KB
```

Now, we can actually start to analyze the data itself.

The `describe` method provides us with a set of summary statistics that broadly describe the data overall.

```
coffee_df.describe()
```

	Number.of.Bags	Harvest.Year	Fragrance...Aroma	Flavor	Aftertaste	Salt...Acid	Bitter...Sweet	Mo
count	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000	28.
mean	168.000000	2013.964286	7.702500	7.630714	7.559643	7.657143	7.675714	7.
std	143.226317	1.346660	0.296156	0.303656	0.342469	0.261773	0.317063	0.
min	1.000000	2012.000000	6.750000	6.670000	6.500000	6.830000	6.670000	5.
25%	1.000000	2013.000000	7.580000	7.560000	7.397500	7.560000	7.580000	7.
50%	170.000000	2014.000000	7.670000	7.710000	7.670000	7.710000	7.750000	7.
75%	320.000000	2015.000000	7.920000	7.830000	7.770000	7.830000	7.830000	7.
max	320.000000	2017.000000	8.330000	8.080000	7.920000	8.000000	8.420000	8.

From this, we can draw several conclusions. FOr example straightforward ones like:

- the smallest number of bags rated is 1 and at least 25% of the coffees rates only had 1 bag
- the first ratings included were 2012 and last in 2017 (min & max)

- Category One defects are not very common (the 75th% is 0)

Or more nuanced ones that compare across variables like

- the raters scored coffee higher on Uniformity.Cup and Clean.Cup than other scores (mean score; only on the ones that seem to have a scale of up to 8/10)
- the coffee varied more in Mouthfeel and Balance that most other scores (the std; only on the ones that seem to have a scale of of up to 8/10)
- there are 3 ratings with no altitude (count of other variables is 28; alt is 25

> On the documentation page for describe the "ℹ See Also" shows the links to the documentation of most of the individual functions. This is a good way to learn about other things, or find something when you are not quite sure what it would be named. Go to a function thats similar to what you want and then look at the related functions.

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

We can use the descriptive statistics on individual columns as well.

5.2.1. Understanding Quantiles

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

We can use the descriptive statistics on individual columns as well.

```
coffee_df['Uniform.Cup'].describe()
```

```
count    28.000000
mean      9.904286
std       0.238753
min       9.330000
25%      10.000000
50%      10.000000
75%      10.000000
max      10.000000
Name: Uniform.Cup, dtype: float64
```

```
coffee_df[['Uniform.Cup','Mouthfeel']].describe()
```

	Uniform.Cup	Mouthfeel
count	28.000000	28.000000
mean	9.904286	7.506786
std	0.238753	0.725152
min	9.330000	5.080000
25%	10.000000	7.500000
50%	10.000000	7.670000
75%	10.000000	7.830000
max	10.000000	8.250000

5.3. Individual statistics

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

```
coffee_df.mean(numeric_only=True)
```

```
Harvest.Year              2013.964286
Fragrance...Aroma            7.702500
Flavor                       7.630714
Aftertaste                   7.559643
Salt...Acid                  7.657143
Bitter...Sweet               7.675714
Mouthfeel                    7.506786
Uniform.Cup                  9.904286
Clean.Cup                    9.928214
Balance                      7.541786
Cupper.Points                7.761429
Total.Cup.Points            80.868929
Moisture                     0.065714
Category.One.Defects         2.964286
Quakers                      0.000000
Category.Two.Defects         1.892857
altitude_low_meters       1367.600000
altitude_high_meters      1387.600000
altitude_mean_meters      1377.600000
dtype: float64
```

```
coffee_df['Flavor'].quantile(.8)
```

```
7.83
```

```
coffee_df['Aftertaste'].mean()
```

```
7.559642857142856
```

```
coffee_df.head(2)
```

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	A
1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	ankole coffee producers	0	ankole coffee producers coop	
2	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethuraman estate	14/1148/2017/21	kaapi royale	

2 rows × 43 columns

5.4. Working with categorical data

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

```
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')
```

```
coffee_df.describe().columns
```

```
       'Uniform.Cup', 'Clean.Cup', 'Balance', 'Cupper.Points',
       'Total.Cup.Points', 'Moisture', 'Category.One.Defects', 'Quakers',
       'Category.Two.Defects', 'altitude_low_meters', 'altitude_high_meters',
       'altitude_mean_meters'],
      dtype='object')
```

We can get the prevalence of each one with `value_counts`

```
coffee_df['Color'].value_counts()
```

```
Green           20
Blue-Green       3
Bluish-Green     2
None             1
Name: Color, dtype: int64
```

Try it Yourself

Note `value_counts` does not count the `NaN` values, but `count` counts all of the not missing values and the shape of the DataFrame is the total number of rows. How can you get the number of missing Colors?

Describe only operates on the numerical columns, but we might want to know about the others. We can get the number of each value with `value_counts`

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

```
India            13
Uganda           10
United States     2
Ecuador           2
Vietnam           1
Name: Country.of.Origin, dtype: int64
```

Value counts returns a pandas Series that has two parts: values and index

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

```
array([13, 10,  2,  2,  1])
```

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

```
Index(['India', 'Uganda', 'United States', 'Ecuador', 'Vietnam'], dtype='object')
```

The max takes the max of the values.

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

```
13
```

We can get the name of the most common country out of this Series using `idmax`

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

Or see only how many different values with the related:

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

```
5
```

5.5. Split-Apply-Combine

So, we can summarize data now, but the summaries we have done so far have treated each variable one at a time. The most interesting patterns are in often in how multiple variables interact. We'll do some modeling that looks at multivariate functions of data in a few weeks, but for now, we do a little more with summary statistics.

For example, how does the flavor ratings relate to the country?

```
coffee_df.groupby('Country.of.Origin')['Flavor'].describe()
```

	count	mean	std	min	25%	50%	75%	max
Country.of.Origin								
Ecuador	2.0	7.625000	0.063640	7.58	7.6025	7.625	7.6475	7.67
India	13.0	7.640769	0.279835	6.83	7.5800	7.750	7.7500	7.92
Uganda	10.0	7.758000	0.197754	7.42	7.6025	7.790	7.8975	8.08
United States	2.0	7.415000	0.120208	7.33	7.3725	7.415	7.4575	7.50
Vietnam	1.0	6.670000	NaN	6.67	6.6700	6.670	6.6700	6.67

Above we saw which country had the most ratings (remember one row is one rating), but what if we wanted to see the mean number of bags per country?

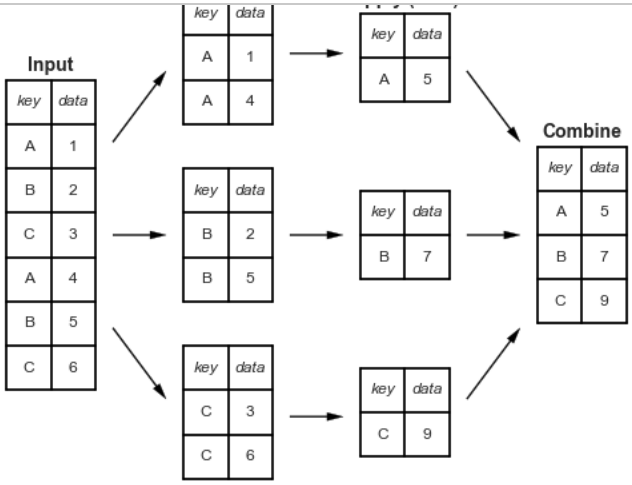
```
coffee_df.groupby('Country.of.Origin')['Number.of.Bags'].mean()
```

```
Country.of.Origin
Ecuador            1.000000
India            230.076923
Uganda           160.900000
United States     50.500000
Vietnam            1.000000
Name: Number.of.Bags, dtype: float64
```

Important

This data is only about coffee that was rated by a particular agency it is not economic data, so we cannot, for example conclude which country *produces* the amount of data. If we had economic dataset, a `Number.of.Bags` columns's mean would tell us exactly that, but the context of the dataset defines what a row means and therefore how we can interpret the **every single statistic** we calculate.

What just happened?



Groupby splits the whole dataframe into parts where each part has the same value for `Country.of.Origin` and then after that, we extracted the `Number.of.Bags` column, took the sum (within each separate group) and then put it all back together in one table (in this case, a `Series` becuase we picked one variable out)

5.5.1. How does Groupby Work?

Important

This is more details with code examples on how the groupby works. If you want to run this code for yourself, use the download icon at the top right to download these notes as a notebook.

We can view this by saving the groupby object as a variable and exploring it.

```
country_grouped = coffee_df.groupby('Country.of.Origin')

country_grouped
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7fb85251fe80>
```

Trying to look at it without applying additional functions, just tells us the type. But, it's iterable, so we can loop over.

```
for country,df in country_grouped:
    print(type(country), type(df))
```

```
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
```

We could manually compute things using the data structure, if needed, though using pandas functionality will usually do what we want. For example:

Note

I used this feature to build the separate view of the communication channels on this website. You can view that source using the github icon on that page.

```
for country,df in country_grouped:
    tot_bags =  df['Number.of.Bags'].sum()
    bag_total_dict[country] = tot_bags

pd.DataFrame.from_dict(bag_total_dict, orient='index',
                       columns =
['Number.of.Bags.Sum'])
```

I tried putting this dictionary into the dataframe for display purposes using the regular constructor and got an error, so I googled about making one from a dictionary to get the docs, which is how I learned about the `from_dict` method and its `orient` parameter which solved my problems.

	Number.of.Bags.Sum
Ecuador	2
India	2991
Uganda	1609
United States	101
Vietnam	1

is the same as what we did before

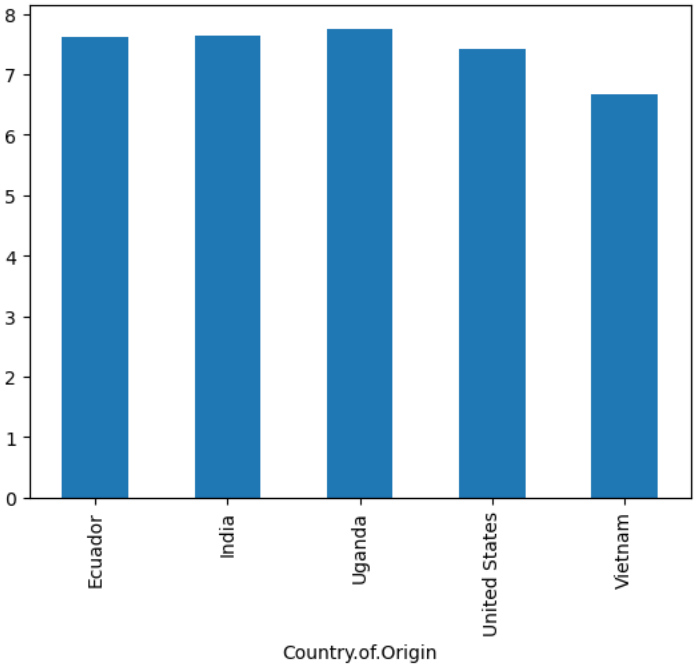
5.6. Plotting with Pandas

Pandas allows us to do basic plots on a `DataFrame` or `Series` with the `plot` method.

We want bars so we will use the `kind` parameter to switch it.

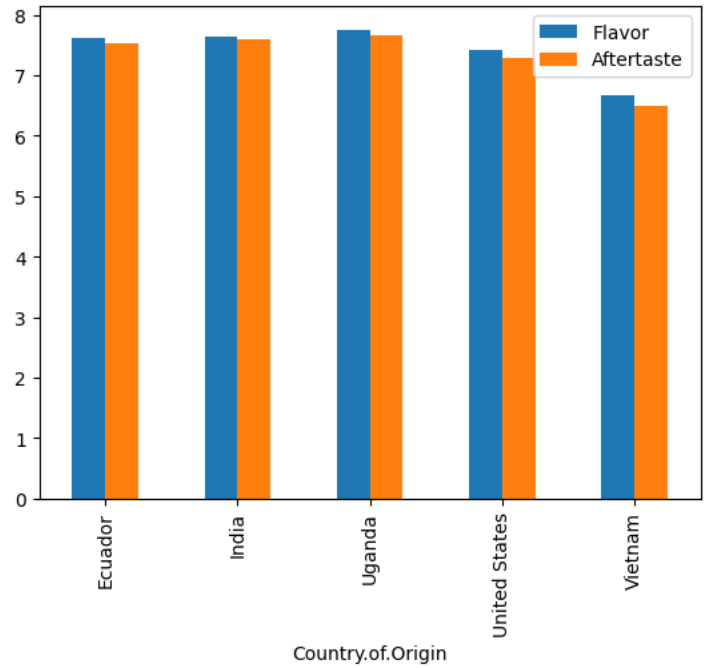
```
coffee_df.groupby('Country.of.Origin')['Flavor'].mean().plot(kind='bar')
```

```
<Axes: xlabel='Country.of.Origin'>
```



It can also be done on a dataframe like this

```
coffee_df.groupby('Country.of.Origin')[['Flavor','Aftertaste']].mean().plot(kind='bar')
```



Note that it adds a legend for us and uses two colors.

What is the default plot type

Try removing the `kind=bar` and see what it does

5.7. Questions after Class

5.7.1. Are there any calls for examples such as.plot or .describe that you do not want us to use?

Everything in pandas is welcome, unless it is deprecated or not recommended by pandas. Pandas will tell, like the FutureWarning that we saw today. Work will be accepted with warnings, most of the time but it is not best practice and some that I specifically tell you to avoid as we encounter them will not be accepted.

5.7.2. What keyboard key do you use to run the code so I don't have to use the mouse?

hold shift, press enter

5.7.3. How do we take the plots and save them to a separate file?

We will need an additional library to do this, we will do that on Thursday.

5.7.4. how do you display different types of charts

the `kind` attribute can change to differen types

5.7.5. When you give feedback on an assignment is there a way to fix it to get the points you said it did not meet?

No, you can attempt in the next assignment or portfolio check.

5.7.6. I want to know more about the limitations of pandas

5.7.7. Can Jupyter use other graphics software?

Jupyter notebooks are a file(roughly a json). You can edit it using any text editor. You can also convert to a plain text files using `jupytext` that is still runnable.

5.7.8. How do we generate different models as done in r, also is there a supernova function?

R is designed by and for statisticians. Most of the calculations can be done. They may be slightly more clunky in Python than R.

5.7.9. I had a question on the assignment, in the datasets.py file were we were supposed to save a function handle, should it be a function object or a string?

function object

5.7.10. Besides accepting the invite, is there any more setup we were supposed to do with the achievement tracker repository?

No, that's it.

6. Visualization

If your plots do not show, include this in any cell. The `%` signals that this is an ipython magic. This one controls matplotlib. Jupyter uses the IPython python kernel.

```
%matplotlib inline
```

Today's imports

```
import pandas as pd
import seaborn as sns
import matplotlib.pylab as plt
```

6.1. Summarizing Review

We will start with the same dataset we hvae been working with

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

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

Is the robust coffee's `Mouthfeel` or the `Aftertaste` more consistently scored in this dataset?

Why?

```
robusta_df[['Mouthfeel','Aftertaste']].describe()
```

count	28.000000	28.000000
mean	7.506786	7.559643
std	0.725152	0.342469
min	5.080000	6.500000
25%	7.500000	7.397500
50%	7.670000	7.670000
75%	7.830000	7.770000
max	8.250000	7.920000

from the lower `std` we can see that Aftertaste is more consistently rated.

We can also save this subset into a smaller dataframe to work with it more and plot it.

```
rob_ma_df = robusta_df[['Mouthfeel','Aftertaste']]
rob_ma_df.head(1)
```

	Mouthfeel	Aftertaste
0	8.25	7.75

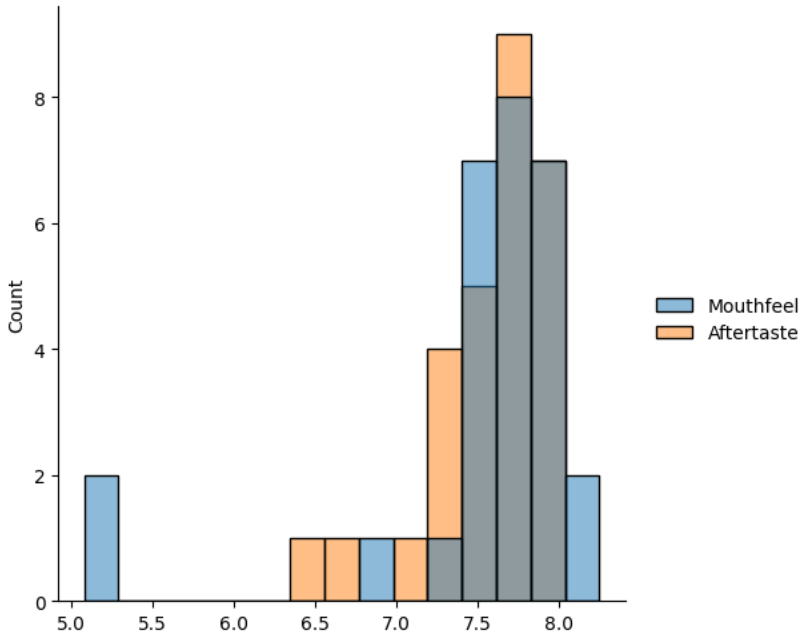
We will use `sns.displot` to look at how the data is distributed.

Important

For `seaborn` the online documentation is **immensely** valuable. Every function's page has basic documentation and lots of examples, so you can see how they use different paramters to modify plots visually. I **strongly recommend reading it often**. I recommend reading their tutorial too

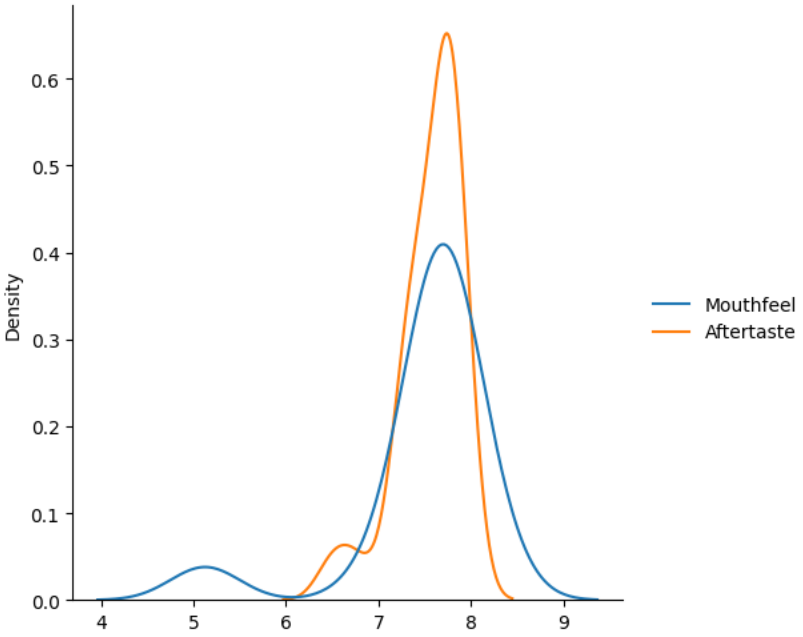
```
sns.displot(rob_ma_df)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f44a02f51f0>
```



We can change the kind, for example to a Kernel Density Estimate. This approximates the distribution of the data, you can think of it rougly like a smoothed out histogram.

```
<seaborn.axisgrid.FacetGrid at 0x7f44746f94c0>
```



This version makess it more visually clear that the the Aftertaste is more consistently, but it also helps us see that that might not be the whole story. Both have a second smaller bump, so the overall std might not be the best measure.

Question from class

Why do we need two sets of brackets?

It tries to use them to index in multiple ways instead.

```
robusta_df['Aftertaste','Mouthfeel']
```

```
KeyError                                  Traceback (most recent call last)
File /opt/hostedtoolcache/Python/3.8.16/x64/lib/python3.8/site-
packages/pandas/core/indexes/base.py:3802, in Index.get_loc(self, key, method, tolerance)
   3801 try:
-> 3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:

File /opt/hostedtoolcache/Python/3.8.16/x64/lib/python3.8/site-packages/pandas/_libs/index.pyx:138, in
pandas._libs.index.IndexEngine.get_loc()

File /opt/hostedtoolcache/Python/3.8.16/x64/lib/python3.8/site-packages/pandas/_libs/index.pyx:165, in
pandas._libs.index.IndexEngine.get_loc()

File pandas/_libs/hashtable_class_helper.pxi:5745, in
pandas._libs.hashtable.PyObjectHashTable.get_item()

File pandas/_libs/hashtable_class_helper.pxi:5753, in
pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: ('Aftertaste', 'Mouthfeel')

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[9], line 1
----> 1 robusta_df['Aftertaste','Mouthfeel']

File /opt/hostedtoolcache/Python/3.8.16/x64/lib/python3.8/site-packages/pandas/core/frame.py:3807, in
DataFrame.__getitem__(self, key)
   3805 if self.columns.nlevels > 1:
   3806     return self._getitem_multilevel(key)
-> 3807 indexer = self.columns.get_loc(key)
   3808 if is_integer(indexer):
   3809     indexer = [indexer]

File /opt/hostedtoolcache/Python/3.8.16/x64/lib/python3.8/site-
packages/pandas/core/indexes/base.py:3804, in Index.get_loc(self, key, method, tolerance)
   3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:
-> 3804     raise KeyError(key) from err
   3805 except TypeError:
   3806     # If we have a listlike key, _check_indexing_error will raise
   3807     #  InvalidIndexError. Otherwise we fall through and re-raise
   3808     #  the TypeError.
   3809     self._check_indexing_error(key)

KeyError: ('Aftertaste', 'Mouthfeel')
```

It tries to look for a multiindex, but we do not have one so it fails. THe second square brackets, makes it a list of names to use and pandas looks for them sequentially.

We will use a larger dataset for more interesting plots.

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

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

6.2. Plotting in Python

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

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.

Think Ahead

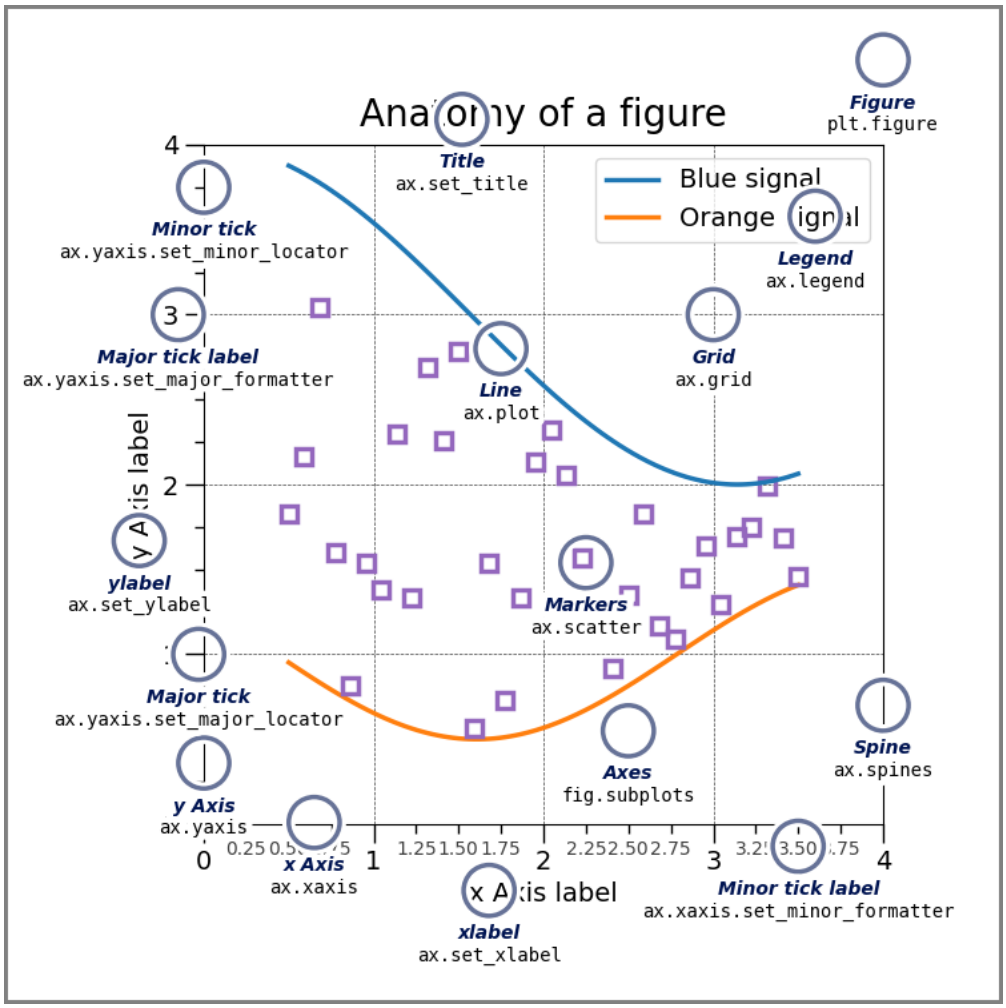
Learning ggplot is a way to earn level 3 for visualize

type of plot you might want and help you understand them to be able to customize plots.

[Seaborn's main goal is opinionated defaults and flexible customization]
(https://seaborn.pydata.org/tutorial/introduction.html#opinionated-defaults-and-flexible-customization

6.2.1. Anatomy of a figure

First is the matplotlib structure of a figure. BOth pandas and seaborn and other plotting libraries use matplotlib. Matplotlib was used in visualizing the first Black hole.



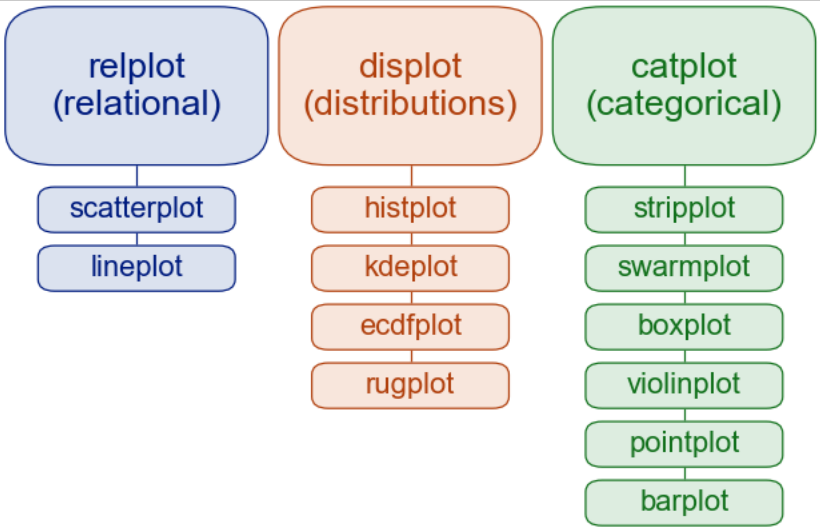
This is a lot of information, but these are good to know things. THe most important is the figure and the axes.

> **Try it Yourself**
>
> Make sure you can explain what is a figure and what are axes in your own words and why that distinction matters. Discuss in office hours if you are unsure.

that image was drawn with code and that page explains more.

6.2.2. Plotting Function types in Seaborn

Seaborn has two *levels* or groups of plotting functions. Figure and axes. Figure level fucntions can plot with subplots.



This is from thie overivew section of the official seaborn tutorial. It also includes a comparison of figure vs axes plotting.

The official introduction is also a good read.

6.2.3. More

The seaborn gallery and matplotlib gallery are nice to look at too.

6.2.4. Styling in Seaborn

Seaborn also lets us set a theme for visual styling This by default styles the plots to be more visually appealing

```
sns.set_theme(palette='colorblind')
```

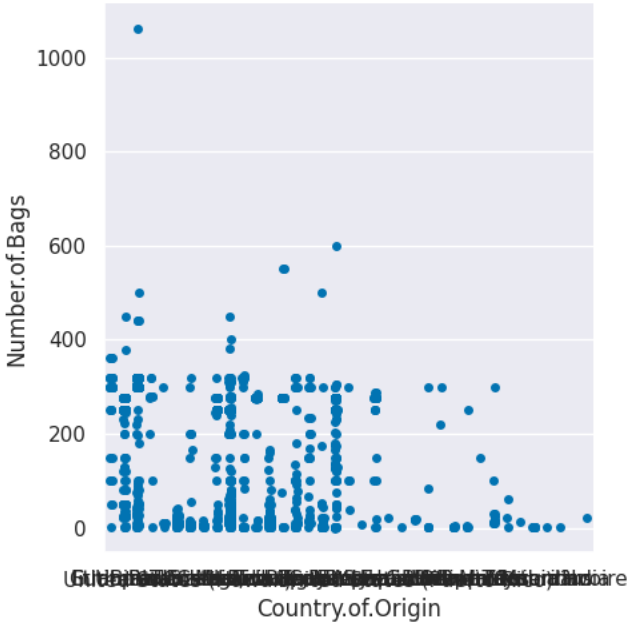
the colorblind palette is more distinguishable under a variety fo colorblindness types. for more. Colorblind is a good default, but you can choose others that you like more too.

more on colors

6.3. Bags by country

the `catplot` lets us plot vs categorical variables.

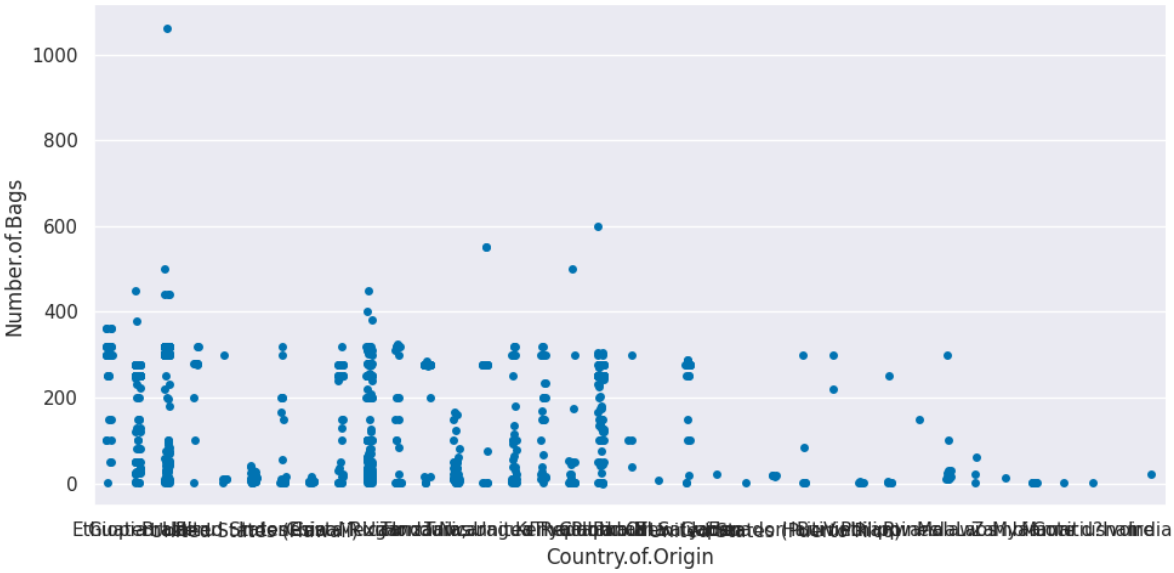
```
sns.catplot(data=coffee_df, y='Number.of.Bags',x='Country.of.Origin')
```



This is hard to read, we could try stretching it out to make it better

```
sns.catplot(data=coffee_df, y='Number.of.Bags',x='Country.of.Origin',aspect=2)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f446a4dfe80>
```



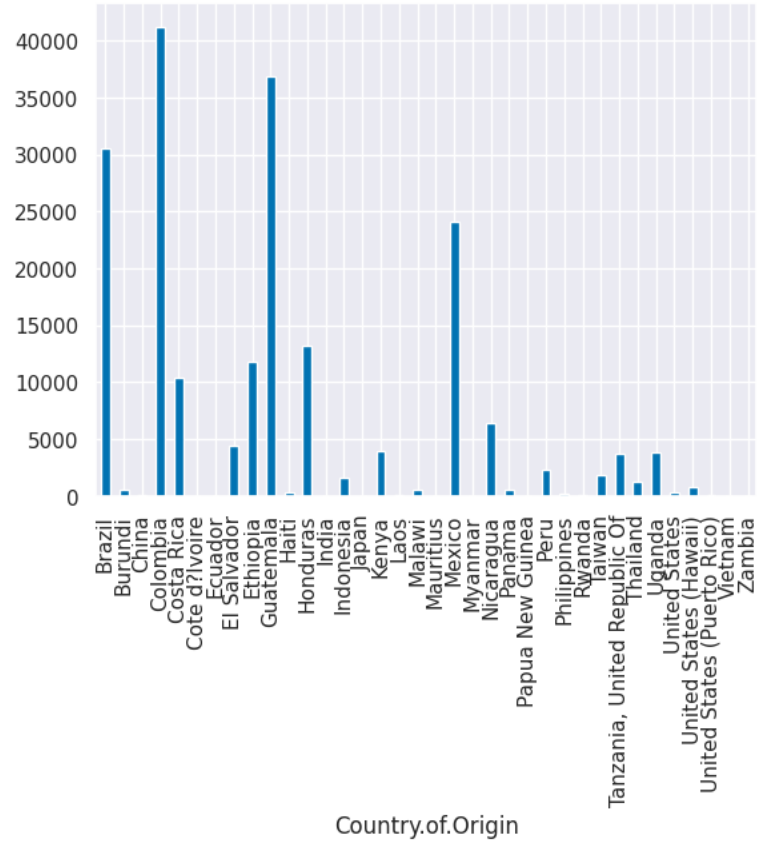
A better way might be to filter only the top countries. We'll find those by grouping by country then summing each smaller dataframe that groupby creates.

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

```
Country.of.Origin
Brazil        30534
Burundi         520
China            55
Colombia      41204
Costa Rica    10354
Name: Number.of.Bags, dtype: int64
```

```
tot_per_country.plot(kind='bar')
```

```
<Axes: xlabel='Country.of.Origin'>
```



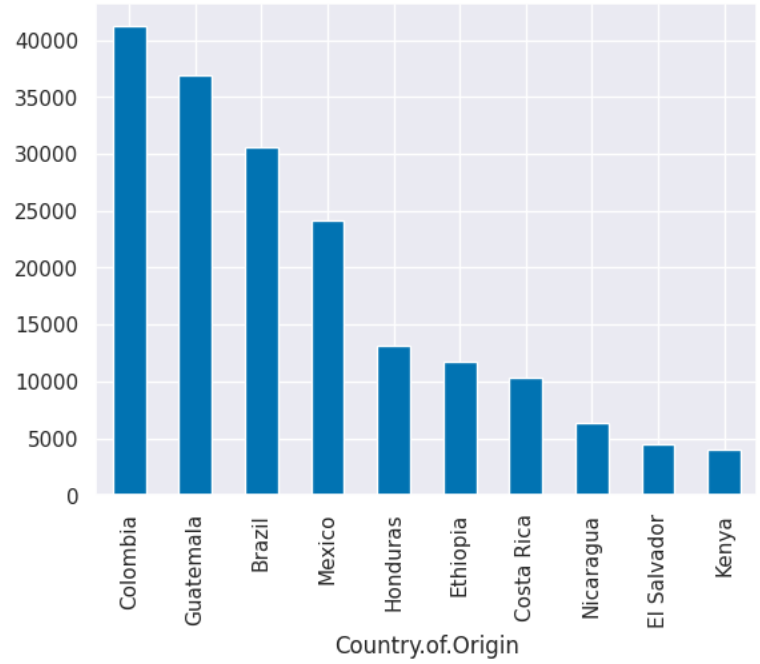
What if we take out only the top 10 countries? First we have to sort it. The default is to sort ascending so we use `ascending=False` to switch. pandas doesn'thave a plain `sort` method, we have to say if we want to sort by the values or the index. In this Series, the total number per for each country are the values and the country names are the index.

```
tot_per_country.sort_values(ascending=False)[:10]
```

```
Country.of.Origin
Colombia       41204
Guatemala      36868
Brazil         30534
Mexico         24140
Honduras       13167
Ethiopia       11761
Costa Rica     10354
Nicaragua       6406
El Salvador     4449
Kenya           3971
Name: Number.of.Bags, dtype: int64
```

We can alo plot this

```
tot_per_country.sort_values(ascending=False)[:10].plot(kind='bar')
```



6.4. Filtering a DataFrame

Now, we'll take just the country names out

```
top_countries = tot_per_country.sort_values(ascending=False)[:10].index
top_countries
```

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

and we can use that to filter the original `DataFrame`. To do this, we use `isin` to check each element in the `'Country.of.Origin'` column is in that list.

```
coffee_df['Country.of.Origin'].isin(top_countries)
```

```
0        True
1        True
2        True
3        True
4        True
        ...
1306     True
1307    False
1308     True
1309     True
1310     True
Name: Country.of.Origin, Length: 1311, dtype: bool
```

This is roughly equivalent to:

```
[country in top_countries for country in coffee_df['Country.of.Origin'] ]
```

```
True,
True,
True,
True,
True,
False,
True,
True,
True,
True,
False,
False,
False,
True,
False,
False,
True,
False,
True,
False,
True,
True,
False,
True,
True,
True,
True,
False,
True,
False,
True,
False,
True,
True,
True,
True,
False,
False,
True,
False,
False,
True,
True,
False,
True,
True,
False,
True,
False,
True,
False,
True,
True,
True,
True,
False,
False,
True,
True,
False,
False,
True,
True,
False,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
```

```
True,
True,
True,
False,
True,
False,
True,
False,
False,
True,
True,
True,
True,
True,
True,
True,
False,
True,
True,
True,
False,
False,
True,
True,
True,
True,
True,
False,
False,
False,
True,
True,
False,
True,
True,
True,
True,
True,
False,
True,
True,
True,
True,
True,
False,
False,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
False,
False,
True,
True,
False,
True,
True,
True,
True,
True,
True,
True,
True,
True,
False,
```

```
False,
False,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
False,
False,
True,
True,
True,
False,
True,
True,
True,
True,
False,
False,
True,
True,
False,
True,
True,
True,
True,
True,
True,
True,
True,
True,
False,
True,
True,
True,
True,
True,
True,
True,
True,
True,
False,
False,
True,
False,
True,
True,
True,
False,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
False,
True,
False,
False,
True,
True,
```

```
False,
True,
True,
False,
True,
True,
True,
False,
True,
True,
True,
True,
True,
True,
True,
True,
False,
True,
False,
True,
True,
True,
True,
True,
False,
True,
True,
True,
True,
True,
False,
False,
True,
True,
True,
True,
False,
True,
False,
True,
False,
False,
True,
True,
True,
True,
False,
True,
True,
True,
True,
True,
False,
True,
False,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
False,
True,
False,
True,
True,
True,
True,
False,
True,
True,
True,
False,
True,
True,
True,
False,
True,
True,
True,
False,
True,
True,
True,
False,
```

```
True,
True,
True,
False,
True,
True,
True,
True,
False,
False,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
False,
False,
True,
True,
True,
True,
False,
False,
False,
True,
True,
True,
False,
True,
True,
True,
True,
False,
True,
True,
False,
False,
True,
True,
True,
True,
True,
True,
False,
True,
True,
False,
True,
False,
False,
True,
True,
False,
False,
True,
False,
False,
True,
True,
True,
True,
False,
True,
True,
True,
False,
False,
True,
True,
True,
True,
False,
True,
True,
True,
```

```
True,
True,
True,
True,
True,
True,
False,
False,
False,
True,
True,
False,
False,
True,
True,
True,
True,
False,
False,
True,
False,
True,
True,
True,
True,
False,
False,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
False,
True,
True,
True,
False,
False,
False,
True,
False,
False,
True,
True,
False,
True,
True,
True,
True,
False,
False,
True,
False,
True,
True,
True,
True,
False,
False,
False,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
```

```
True,
True,
True,
False,
True,
False,
False,
True,
True,
False,
True,
True,
True,
False,
True,
False,
False,
False,
True,
True,
True,
True,
True,
True,
True,
True,
False,
True,
False,
True,
False,
True,
True,
True,
True,
False,
False,
True,
True,
True,
False,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
False,
True,
False,
True,
True,
True,
True,
False,
False,
True,
False,
True,
True,
False,
False,
True,
True,
True,
True,
True,
True,
False,
False,
False,
True,
```

```
True,
True,
True,
True,
True,
False,
False,
False,
True,
False,
False,
False,
True,
True,
True,
False,
True,
True,
True,
False,
True,
True,
True,
True,
False,
True,
False,
True,
False,
False,
False,
False,
False,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
False,
True,
True,
False,
False,
False,
False,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
False,
True,
True,
True,
True,
True,
True,
True,
False,
True,
False,
True,
True,
True,
True,
False,
True,
True,
True,
True,
True,
True,
True,
```

```
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
False,
True,
True,
True,
False,
True,
True,
True,
True,
False,
False,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
False,
True,
True,
True,
True,
False,
True,
False,
True,
False,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
False,
False,
True,
False,
True,
True,
True,
False,
True,
True,
False,
False,
True,
True,
True,
True,
True,
True,
False,
True,
True,
True,
True,
True,
```

```
True,
True,
False,
True,
True,
True,
False,
True,
False,
True,
True,
True,
True,
False,
False,
False,
False,
False,
False,
True,
True,
True,
False,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
False,
True,
False,
True,
False,
True,
True,
True,
True,
True,
True,
True,
True,
True,
True,
False,
True,
True,
True,
False,
True,
True,
True,
False,
False,
False,
True,
True,
True,
True,
False,
True,
True,
True,
True,
True,
True,
False,
True,
False,
False,
True,
True,
True,
True,
True,
True,
True,
```

```
True,
True,
True,
False,
True,
True,
True,
False,
False,
True,
False,
True,
True,
True,
False,
False,
True,
True,
True,
True,
True,
True,
True,
False,
True,
True,
False,
True,
False,
False,
False,
True,
True,
True,
True,
True,
False,
False,
True,
True,
True,
True,
True,
False,
False,
True,
False,
True,
False,
True,
False,
False,
True,
True,
True,
True,
True,
True,
False,
True,
True,
True,
True,
False,
True,
True,
True,
True,
False,
False,
True,
True,
True,
False,
True,
True,
True,
True,
True,
False,
True,
True,
True,
True,
True,
True,
```

```
True,
True,
True,
True,
False,
True,
True,
True,
True,
False,
False,
True,
True,
True,
False,
True,
True,
True,
False,
False,
False,
False,
True,
True,
True,
True,
True,
False,
True,
True,
True,
False,
False,
True,
False,
True,
True,
True,
True,
False,
True,
True,
...]
```

except this builds a list and the pandas way makes a `pd.Series` object. The Python `in` operator is really helpful to know and pandas offers us an `isin` method to get that type of pattern.

In a more basic programming format this process would be two separate loops worth of work.

```
c_in = []
# iterate over the country of each rating
for country in coffee_df['Country.of.Origin']:
    # make a false temp value
    cur_search = False
    # iterate over top countries
    for tc in top_countries:
        # flip the value if the current top & rating cofee match
        if tc==country:
            cur_search = True
    # save the result of the search
    c_in.append(cur_search)
```

Try it yourself

Run these versions and confirm for yourself that they are the same.

With that list of booleans, we can then mask the original DataFrame. This keeps only the value where the inner quantity is `True`

```
top_coffee_df = coffee_df[coffee_df['Country.of.Origin'].isin(top_countries)]
top_coffee_df.head(1)
```

0	1	Arabica	metad plc		Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc

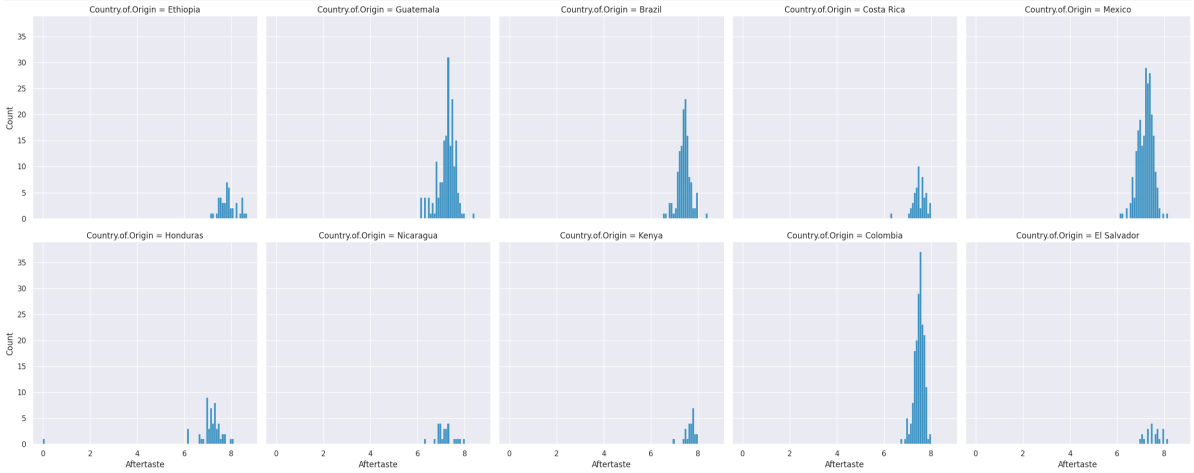
1 rows × 44 columns

```
top_coffee_df.shape, coffee_df.shape
```

```
((952, 44), (1311, 44))
```

```
sns.displot(data=top_coffee_df,x='Aftertaste', col='Country.of.Origin',col_wrap=5)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f4468d5f4f0>
```



6.5. 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.

6.6.1. Do we earn level 3's the same way level 1 and 2 are or are there more steps required?

You earn level 3s from your portfio. The portfolio makes more sense after you have completed assignment 3, so we will follow up on it next week after you all get a3 feedback.

6.6.2. How can I check what parameters can go into a method?

You can use the documentation online, or in jupyter, you can get help from the docstring. I usually use shift+tab to read the docstring but you can also use the `help()` function or the `?` in jupyter.

6.6.3. How do you know you can put kind = "bar" into the method?

I happen to reembmer this now, but to know what values you can read the docstring as above.

6.6.4. Do companies use things like "sns" for more in depth/graphical plots?

It depends on your role within the company. If you are a data scientist in a more reasearch role you might use seaborn more, but if you build customer facing visualizations, you might use something else.

For more interactive visualization, you could use plotly or bokeh that generate more javascript for you. Plotly as a company now also has a product called dash for building data dashboard apps.

6.6.5. Does "component disciplines" mean statistics, computer science and domain expertise, and does "phases" mean collect, clean, explore, model and deploy?

Yes.

> **Important**
>
> I updated the assignment text to clarify in response to some questions

7. Tidy Data and Reshaping Datasets

```
import pandas as pd
import seaborn as sns

sns.set_theme(palette='colorblind',font_scale=2)
```

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

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

```
list_of_df = [pd.read_csv(url_base + dataset,na_values='-') for dataset in datasets]
```

```
list_of_df[0]
```

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

	intervention	John Smith	Jane Doe	Mary Johnson
0	treatmenta	NaN	16	3
1	treatmentb	2.0	11	1

```
list_of_df[2]
```

	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

```
list_of_df[2].mean()
```

```
/tmp/ipykernel_1879/1880485675.py:1: FutureWarning: The default value of numeric_only in
DataFrame.mean is deprecated. In a future version, it will default to False. In addition, specifying
'numeric_only=None' is deprecated. Select only valid columns or specify the value of numeric_only to
silence this warning.
  list_of_df[2].mean()
```

```
result    6.6
dtype: float64
```

```
sum([16,3,2,11,1])/5
```

```
6.6
```

```
sum([16,3,2,11,1,0])/6
```

```
5.5
```

```
list_of_df[2].groupby('treatment').mean()
```

```
/tmp/ipykernel_1879/2310255724.py:1: FutureWarning: The default value of numeric_only in
DataFrameGroupBy.mean is deprecated. In a future version, numeric_only will default to False. Either
specify numeric_only or select only columns which should be valid for the function.
  list_of_df[2].groupby('treatment').mean()
```

	result
treatment	
a	9.500000
b	4.666667

```
list_of_df[2].groupby('person').mean()
```

```
DataFrameGroupBy.mean is deprecated. In a future version, numeric_only will default to False. Either
specify numeric_only or select only columns which should be valid for the function.
  list_of_df[2].groupby('person').mean()
```

person	result
Jane Doe	13.5
John Smith	2.0
Mary Johnson	2.0

```
dfa = list_of_df[0]
dfa
```

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

```
dfa.melt(id_vars=['name'],var_name='treatment',value_name='result')
```

	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

```
arabica_data_url = 'https://raw.githubusercontent.com/jldbc/coffee-quality-
database/master/data/arabica_data_cleaned.csv'
# load the data
coffee_df = pd.read_csv(arabica_data_url)
# get total bags per country
bags_per_country = coffee_df.groupby('Country.of.Origin')['Number.of.Bags'].sum()

# 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)]
```

```
bags_per_country
```

```
Brazil                          30354
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
Kenya                            3971
Laos                               81
Malawi                            557
Mauritius                           1
Mexico                          24140
Myanmar                            10
Nicaragua                        6406
Panama                            537
Papua New Guinea                    7
Peru                             2336
Philippines                       259
Rwanda                            150
Taiwan                           1914
Tanzania, United Republic Of     3760
Thailand                         1310
Uganda                           3868
United States                     361
United States (Hawaii)            833
United States (Puerto Rico)        71
Vietnam                            10
Zambia                             13
Name: Number.of.Bags, dtype: int64
```

```
top_bags_country_list
```

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

```
top_coffee_df.head(1)
```

	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 × 44 columns

```
coffee_df.head(1)
```

	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 × 44 columns

```
coffee_df.shape,top_coffee_df.shape
```

```
top_coffee_df.describe()
```

	Unnamed: 0	Number.of.Bags	Aroma	Flavor	Aftertaste	Acidity	Body	Balanc
count	952.000000	952.000000	952.000000	952.000000	952.000000	952.000000	952.000000	952.00000
mean	653.811975	192.073529	7.557468	7.513330	7.379338	7.533172	7.505662	7.51321
std	378.427772	120.682457	0.400004	0.418425	0.430553	0.403558	0.383316	0.43414
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
25%	323.750000	50.000000	7.420000	7.330000	7.170000	7.330000	7.330000	7.33000
50%	659.500000	250.000000	7.580000	7.580000	7.420000	7.500000	7.500000	7.50000
75%	972.500000	275.000000	7.750000	7.750000	7.580000	7.750000	7.670000	7.75000
max	1312.000000	1062.000000	8.750000	8.830000	8.670000	8.750000	8.580000	8.75000

```
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')
```

```
ratings_of_interest = ['Aroma', 'Flavor', 'Aftertaste', 'Acidity', 'Body',
       'Balance', ]
coffe_scores_df = top_coffee_df.melt(id_vars='Country.of.Origin',value_vars=ratings_of_interest,
                  var_name='rating',value_name='score')
coffe_scores_df.head(1)
```

	Country.of.Origin	rating	score
0	Ethiopia	Aroma	8.67

```
top_coffee_df.melt(id_vars='Country.of.Origin')['variable'].unique()
```

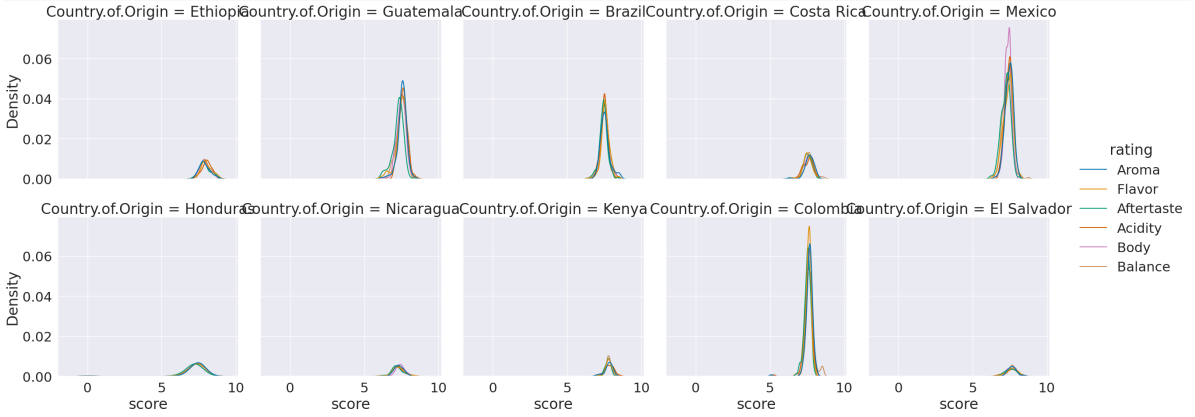
```
array(['Unnamed: 0', 'Species', 'Owner', 'Farm.Name', 'Lot.Number',
       'Mill', 'ICO.Number', 'Company', 'Altitude', 'Region', 'Producer',
       'Number.of.Bags', 'Bag.Weight', 'In.Country.Partner',
       'Harvest.Year', 'Grading.Date', 'Owner.1', 'Variety',
       'Processing.Method', 'Aroma', '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)
```

```
top_coffee_df.melt(id_vars='Country.of.Origin',value_vars=ratings_of_interest,)['variable'].unique()
```

```
array(['Aroma', 'Flavor', 'Aftertaste', 'Acidity', 'Body', 'Balance'],
      dtype=object)
```

```
sns.displot(data=coffe_scores_df, x='score',col='Country.of.Origin',
            hue = 'rating',col_wrap=5,kind='kde')
```

```
<seaborn.axisgrid.FacetGrid at 0x7effb84670a0>
```




```
sns.displot(data=coffe_scores_df, x='score',hue='Country.of.Origin',
            col = 'rating',col_wrap=3,kind='kde')
```

```
<seaborn.axisgrid.FacetGrid at 0x7eff882c5ca0>
```


```
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')
```

```
[ 'Country.of.Origin','Color'],value_vars=ratings_of_interest,
                   var_name='rating',value_name='score')
coffe_scores_df2.head(1)
```

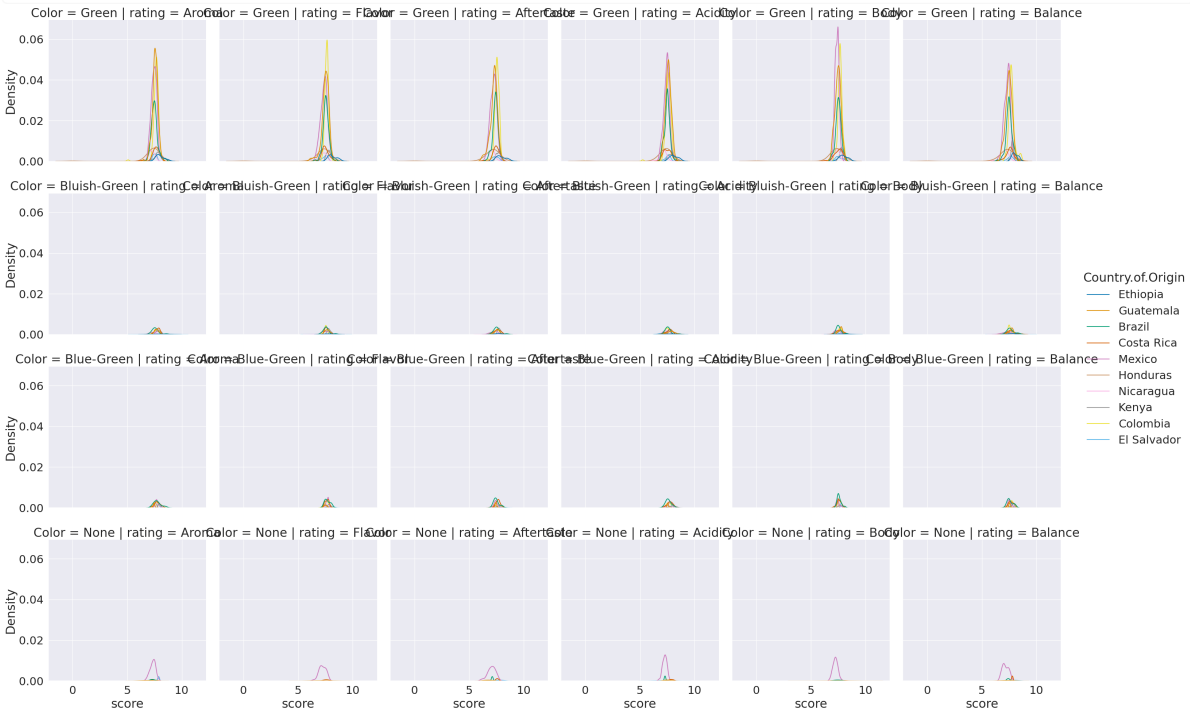
	Country.of.Origin	Color	rating	score
0	Ethiopia	Green	Aroma	8.67

```
sns.displot(data=coffe_scores_df2, x='score',hue='Country.of.Origin',
           col = 'rating',row='Color',kind='kde')
```

```
/tmp/ipykernel_1879/3482930274.py:1: UserWarning: Dataset has 0 variance; skipping density estimate.
Pass `warn_singular=False` to disable this warning.
  sns.displot(data=coffe_scores_df2, x='score',hue='Country.of.Origin',
/tmp/ipykernel_1879/3482930274.py:1: UserWarning: Dataset has 0 variance; skipping density estimate.
Pass `warn_singular=False` to disable this warning.
  sns.displot(data=coffe_scores_df2, x='score',hue='Country.of.Origin',
/tmp/ipykernel_1879/3482930274.py:1: UserWarning: Dataset has 0 variance; skipping density estimate.
Pass `warn_singular=False` to disable this warning.
  sns.displot(data=coffe_scores_df2, x='score',hue='Country.of.Origin',
/tmp/ipykernel_1879/3482930274.py:1: UserWarning: Dataset has 0 variance; skipping density estimate.
Pass `warn_singular=False` to disable this warning.
  sns.displot(data=coffe_scores_df2, x='score',hue='Country.of.Origin',
/tmp/ipykernel_1879/3482930274.py:1: UserWarning: Dataset has 0 variance; skipping density estimate.
Pass `warn_singular=False` to disable this warning.
  sns.displot(data=coffe_scores_df2, x='score',hue='Country.of.Origin',
/tmp/ipykernel_1879/3482930274.py:1: UserWarning: Dataset has 0 variance; skipping density estimate.
Pass `warn_singular=False` to disable this warning.
  sns.displot(data=coffe_scores_df2, x='score',hue='Country.of.Origin',
```

```
/tmp/ipykernel_1879/3482930274.py:1: UserWarning: Dataset has 0 variance; skipping density estimate.
Pass `warn_singular=False` to disable this warning.
  sns.displot(data=coffe_scores_df2, x='score',hue='Country.of.Origin',
/tmp/ipykernel_1879/3482930274.py:1: UserWarning: Dataset has 0 variance; skipping density estimate.
Pass `warn_singular=False` to disable this warning.
  sns.displot(data=coffe_scores_df2, x='score',hue='Country.of.Origin',
/tmp/ipykernel_1879/3482930274.py:1: UserWarning: Dataset has 0 variance; skipping density estimate.
Pass `warn_singular=False` to disable this warning.
  sns.displot(data=coffe_scores_df2, x='score',hue='Country.of.Origin',
```

```
<seaborn.axisgrid.FacetGrid at 0x7eff88267a00>
```



	Unnamed: 0	Number.of.Bags	Aroma	Flavor	Aftertaste	Acidity	Body	
count	1311.000000	1311.000000	1311.000000	1311.000000	1311.000000	1311.000000	1311.000000	131
mean	656.000763	153.887872	7.563806	7.518070	7.397696	7.533112	7.517727	
std	378.598733	129.733734	0.378666	0.399979	0.405119	0.381599	0.359213	
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	328.500000	14.500000	7.420000	7.330000	7.250000	7.330000	7.330000	
50%	656.000000	175.000000	7.580000	7.580000	7.420000	7.500000	7.500000	
75%	983.500000	275.000000	7.750000	7.750000	7.580000	7.750000	7.670000	
max	1312.000000	1062.000000	8.750000	8.830000	8.670000	8.750000	8.580000	

7.1. More manipulations

Here, we will make a tiny `DataFrame` from scratch to illustrate a couple of points

```
large_num_df = pd.DataFrame(data= [[730000000,392000000,580200000],
                                   [315040009,580000000,967290000]],
                            columns = ['a','b','c'])
large_num_df
```

	a	b	c
0	730000000	392000000	580200000
1	315040009	580000000	967290000

This dataet is not tidy, but making it this way was faster to set it up. We could make it tidy using melt as is.

```
large_num_df.melt()
```

	variable	value
0	a	730000000
1	a	315040009
2	b	392000000
3	b	580000000
4	c	580200000
5	c	967290000

However, I want an additional variable, so I wil reset the index, which adds an index column for the original index and adds a new index that is numerical. In this case they're the same.

```
large_num_df.reset_index()
```

	index	a	b	c
0	0	730000000	392000000	580200000
1	1	315040009	580000000	967290000

If I melt this one, using the index as the `id`, then I get a reasonable tidy DataFrame

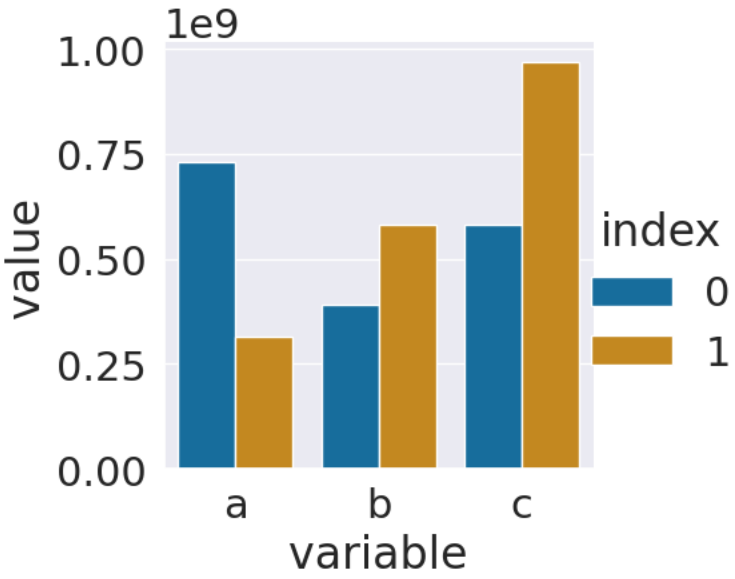
```
ls_tall_df
```

	index	variable	value
0	0	a	730000000
1	1	a	315040009
2	0	b	392000000
3	1	b	580000000
4	0	c	580200000
5	1	c	967290000

Now, we can plot.

```
sns.catplot(data = ls_tall_df,x='variable',y='value',
            hue='index',kind='bar')
```

```
<seaborn.axisgrid.FacetGrid at 0x7eff83b00550>
```



Since the numbers are so big, this might be hard to interpret. Displaying it with all the 0s would not be easier to read. The best thing to do is to add a new colum with adjusted values and a corresponding title.

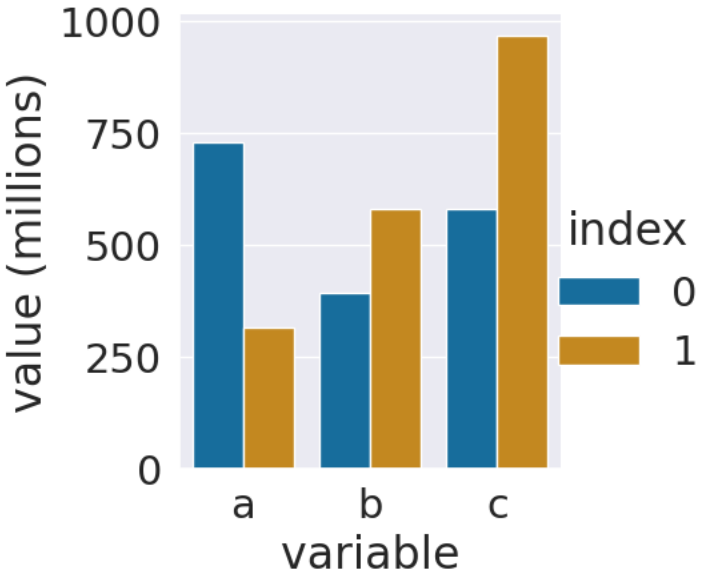
```
ls_tall_df['value (millions)'] = ls_tall_df['value']/1000000
ls_tall_df.head()
```

	index	variable	value	value (millions)
0	0	a	730000000	730.000000
1	1	a	315040009	315.040009
2	0	b	392000000	392.000000
3	1	b	580000000	580.000000
4	0	c	580200000	580.200000

Now we can plot again, with the smaller values and an updated axis label. Adding a column with the adjusted title is good practice because it does not lose any data and since we set the value and the title at the same time it keeps it clear what the values are.

```
hue= 'index',kind= 'bar' )
```

```
<seaborn.axisgrid.FacetGrid at 0x7eff839b3f10>
```



8. Reparing values

So 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 manipulations get used.
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?

```
import seaborn as sns
import numpy as np #
na_toy_df_np = pd.DataFrame(data = [[1,3,4,5],[2 ,6, np.nan]])
na_toy_df_pd = pd.DataFrame(data = [[1,3,4,5],[2 ,6, pd.NA]])

# make plots look nicer and increase font size
sns.set_theme(font_scale=2)
arabica_data_url = 'https://raw.githubusercontent.com/jldbc/coffee-quality-
database/master/data/arabica_data_cleaned.csv'

coffee_df = pd.read_csv(arabica_data_url,index_col=0)


rhodyprog4ds_gh_events_url = 'https://api.github.com/orgs/rhodyprog4ds/events'
course_gh_df = pd.read_json(rhodyprog4ds_gh_events_url)
```

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"

8.3. Fixing Column names

```
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', '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')
```

```
col_name_mapper = {col_name:col_name.lower().replace('.','_') for col_name in coffee_df.columns}
```

```
coffee_df.rename(columns=col_name_mapper).head(1)
```

	species	owner	country_of_origin	farm_name	lot_number	mill	ico_number	company	altitude	re
1	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	han

1 rows × 43 columns

```
coffee_df.head(1)
```

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	R
1	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	ha

1 rows × 43 columns

```
coffee_df_fixedcols.head(1)
```

	species	owner	country_of_origin	farm_name	lot_number	mill	ico_number	company	altitude	re
1	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	han

1 rows × 43 columns

```
coffee_df_fixedcols['unit_of_measurement'].value_counts()
```

```
m     1129
ft     182
Name: unit_of_measurement, dtype: int64
```

```
coffee_df_fixedcols['unit_of_measurement'].replace({'m':'meters','ft':'feet'})
```

```
1       meters
2       meters
3       meters
4       meters
5       meters
         ...
1307    meters
1308    meters
1309    meters
1310      feet
1312    meters
Name: unit_of_measurement, Length: 1311, dtype: object
```

```
coffee_df_fixedcols['unit_of_measurement_long'] = coffee_df_fixedcols['unit_of_measurement'].replace(
                                    {'m':'meters','ft':'feet'})
coffee_df_fixedcols.head(1)
```

	species	owner	country_of_origin	farm_name	lot_number	mill	ico_number	company	altitude	re
1	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	han

1 rows × 44 columns

8.4. Missing Values

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:

- fillna

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.

Filling can be good if you know how to fill reasonably, but don't have data to spare by dropping. For example

- 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

```
coffee_df_fixedcols.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1311 entries, 1 to 1312
Data columns (total 44 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   species                   1311 non-null   object 
 1   owner                     1304 non-null   object 
 2   country_of_origin         1310 non-null   object 
 3   farm_name                 955 non-null    object 
 4   lot_number                270 non-null    object 
 5   mill                      1001 non-null   object 
 6   ico_number                1165 non-null   object 
 7   company                   1102 non-null   object 
 8   altitude                  1088 non-null   object 
 9   region                    1254 non-null   object 
 10  producer                  1081 non-null   object 
 11  number_of_bags            1311 non-null   int64  
 12  bag_weight                1311 non-null   object 
 13  in_country_partner        1311 non-null   object 
 14  harvest_year              1264 non-null   object 
 15  grading_date              1311 non-null   object 
 16  owner_1                   1304 non-null   object 
 17  variety                   1110 non-null   object 
 18  processing_method         1159 non-null   object 
 19  aroma                     1311 non-null   float64
 20  flavor                    1311 non-null   float64
 21  aftertaste                1311 non-null   float64
 22  acidity                   1311 non-null   float64
 23  body                      1311 non-null   float64
 24  balance                   1311 non-null   float64
 25  uniformity                1311 non-null   float64
 26  clean_cup                 1311 non-null   float64
 27  sweetness                 1311 non-null   float64
 28  cupper_points             1311 non-null   float64
 29  total_cup_points          1311 non-null   float64
 30  moisture                  1311 non-null   float64
 31  category_one_defects      1311 non-null   int64  
 32  quakers                   1310 non-null   float64
 33  color                     1095 non-null   object 
 34  category_two_defects      1311 non-null   int64  
 35  expiration                1311 non-null   object 
 36  certification_body        1311 non-null   object 
 37  certification_address     1311 non-null   object 
 38  certification_contact     1311 non-null   object 
 39  unit_of_measurement       1311 non-null   object 
 40  altitude_low_meters       1084 non-null   float64
 41  altitude_high_meters      1084 non-null   float64
 42  altitude_mean_meters      1084 non-null   float64
 43  unit_of_measurement_long  1311 non-null   object 
dtypes: float64(16), int64(3), object(25)
memory usage: 460.9+ KB
```

8.4.1. Filling missing values

We can look at the type:

```
coffee_df_fixedcols['lot_number'].dtype
```

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

And we can look at the value counts.

```
coffee_df_fixedcols['lot_number'].value_counts()
```

```
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: lot_number, 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_fixedcols['lot_number'].fillna('1')
```

```
1                               1
2                               1
3                               1
4                               1
5                               1
                   ...
1307                            1
1308                            1
1309    017-053-0211/ 017-053-0212
1310                            1
1312                          103
Name: lot_number, Length: 1311, 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 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’ll use adding a column:

```
coffee_df_fixedcols['lot_number_clean'] = coffee_df_fixedcols['lot_number'].fillna('1')
```

```
coffee_df_fixedcols['lot_number_clean'].value_counts()
```

```
020/17                          3
019/17                          5
102                             3
103                             3
                              ...
3-59-2318                       1
8885                            1
5055                            1
MCCFWXA15/16                    1
017-053-0211/ 017-053-0212      1
Name: lot_number_clean, Length: 221, dtype: int64
```

8.4.2. Dropping missing values

To illustrate how `dropna` works, we'll use the `shape` method:

```
coffee_df_fixedcols.shape
```

```
(1311, 45)
```

```
coffee_df_fixedcols.dropna().shape
```

```
(130, 45)
```

By default, it drops any row with one or more `NaN` values.

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

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

```
(1084, 45)
```

```
coffee_alt_df = coffee_df_fixedcols.dropna(subset=['altitude_low_meters'])
```

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, n_cols = coffee_df_fixedcols.shape
coffee_df_fixedcols.dropna(thresh = .7*n_rows, axis=1).shape
```

```
(1311, 44)
```

This dataset is actually in pretty good shape, but if we use a more stringent threshold it drops more columns.

```
coffee_df_fixedcols.dropna(thresh = .85*n_rows, axis=1).shape
```

```
(1311, 34)
```

8.5. Inconsistent values

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_fixedcols['in_country_partner'].value_counts().sort_index()
```

```
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
Specialty Coffee Association                                               295
Specialty Coffee Association of Costa Rica                                  42
Specialty Coffee Association of Indonesia                                   10
Specialty Coffee Institute of Asia                                          16
Tanzanian Coffee Board                                                       6
Torch Coffee Lab Yunnan                                                      2
Uganda Coffee Development Authority                                         22
Yunnan Coffee Exchange                                                      12
Name: in_country_partner, dtype: int64
```

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_clean = coffee_df_fixedcols.replace(partner_corrections)
```

8.6. Example: Unpacking Jsons

```
rhodyprog4ds_gh_events_url
```

```
'https://api.github.com/orgs/rhodyprog4ds/events'
```

```
gh_df = pd.read_json(rhodyprog4ds_gh_events_url)
gh_df.head()
```

0	27534576446	CreateEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 592944632, 'name': 'rhodyprog4ds/BrownS...	{'ref': 'c12', 'ref_type': 'tag', 'master_bran...	True	2023-03-07 01:52:36+00:00	{'i 'rl
1	27534572112	PushEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 592944632, 'name': 'rhodyprog4ds/BrownS...	{'repository_id': 592944632, 'push_id': 128482...	True	2023-03-07 01:52:16+00:00	{'i 'rl
2	27423376182	PushEvent	{'id': 41898282, 'login': 'github-actions[bot]...	{'id': 592944632, 'name': 'rhodyprog4ds/BrownS...	{'repository_id': 592944632, 'push_id': 127911...	True	2023-03-01 18:18:49+00:00	{'i 'rl
3	27423307938	PushEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 592944632, 'name': 'rhodyprog4ds/BrownS...	{'repository_id': 592944632, 'push_id': 127911...	True	2023-03-01 18:15:43+00:00	{'i 'rl
4	27413353459	PushEvent	{'id': 41898282, 'login': 'github-actions[bot]...	{'id': 592944632, 'name': 'rhodyprog4ds/BrownS...	{'repository_id': 592944632, 'push_id': 127863...	True	2023-03-01 12:07:33+00:00	{'i 'rl

Some datasets have a nested structure

We want to transform each one of those from a dictionary like thing into a row in a data frame.

We can see each row is a Series type.

```
type(gh_df.loc[0])
```

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

```
a= '1'
type(a)
```

```
str
```

Recall, that base python types can be used as function, to cast an object from type to another.

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

```
int
```

This works with Pandas Series too

```
pd.Series(gh_df.loc[0]['actor'])
```

```
id                                             10656079
login                                       brownsarahm
display_login                               brownsarahm
gravatar_id
url                      https://api.github.com/users/brownsarahm
avatar_url       https://avatars.githubusercontent.com/u/10656079?
dtype: object
```

We can use pandas `apply` to do the same thing to every item in a dataset (over rows or columns as items) For example

	id	login	display_login	gravatar_id	url	
0	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.gith
1	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.gith
2	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.gith
3	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.gith
4	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.gith

compared to the original:

```
gh_df.head(1)
```

	id	type	actor	repo	payload	public	created_at	
0	27534576446	CreateEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 592944632, 'name': 'rhodyprog4ds/BrownS...	{'ref': 'c12', 'ref_type': 'tag', 'master_bran...	True	2023-03-07 01:52:36+00:00	{'i 'rh

We want to handle several columns this way, so we'll make alist of the names.

```
js_cols = ['actor','repo','payload','org']
```

`pd.concat` takes a list of dataframes and puts the together in one DataFrame.

```
pd.concat([gh_df[col].apply(pd.Series) for col in js_cols],axis=1).head()
```

	id	login	display_login	gravatar_id	url	
0	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.gith
1	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.gith
2	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.gith
3	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.gith
4	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.gith

5 rows × 28 columns

This is close, but a lot of columns have the same name. To fix this we will rename the new columns so that they have the original column name prepended to the new name.

pandas has a rename method for this.

and this is another job for lambdas.

```
pd.concat([gh_df[col].apply(pd.Series).rename(lambda c: '_'.join([c,col])) for col in
js_cols],axis=1).head()
```

```
TypeError                                 Traceback (most recent call last)
Cell In[35], line 1
----> 1 pd.concat([gh_df[col].apply(pd.Series).rename(lambda c: '_'.join([c,col])) for col in
js_cols],axis=1).head()

Cell In[35], line 1, in <listcomp>(.0)
----> 1 pd.concat([gh_df[col].apply(pd.Series).rename(lambda c: '_'.join([c,col])) for col in
js_cols],axis=1).head()

File /opt/hostedtoolcache/Python/3.8.16/x64/lib/python3.8/site-packages/pandas/core/frame.py:5573, in
DataFrame.rename(self, mapper, index, columns, axis, copy, inplace, level, errors)
   5454 def rename(
   5455     self,
   5456     mapper: Renamer | None = None,
   (...)
   5464     errors: IgnoreRaise = "ignore",
   5465 ) -> DataFrame | None:
   5466     """
   5467     Alter axes labels.
   5468
   (...)
   5571     4  3  6
   5572     """
-> 5573     return super()._rename(
   5574         mapper=mapper,
   5575         index=index,
   5576         columns=columns,
   5577         axis=axis,
   5578         copy=copy,
   5579         inplace=inplace,
   5580         level=level,
   5581         errors=errors,
   5582     )

File /opt/hostedtoolcache/Python/3.8.16/x64/lib/python3.8/site-packages/pandas/core/generic.py:1104,
in NDFrame._rename(self, mapper, index, columns, axis, copy, inplace, level, errors)
   1097         missing_labels = [
   1098             label
   1099             for index, label in enumerate(replacements)
   1100             if indexer[index] == -1
   1101         ]
   1102         raise KeyError(f"{missing_labels} not found in axis")
-> 1104 new_index = ax._transform_index(f, level=level)
   1105 result._set_axis_nocheck(new_index, axis=axis_no, inplace=True, copy=False)
   1106 result._clear_item_cache()

File /opt/hostedtoolcache/Python/3.8.16/x64/lib/python3.8/site-
packages/pandas/core/indexes/base.py:6416, in Index._transform_index(self, func, level)
   6414     return type(self).from_tuples(items, names=self.names)
   6415 else:
-> 6416     items = [func(x) for x in self]
   6417     return Index(items, name=self.name, tupleize_cols=False)

File /opt/hostedtoolcache/Python/3.8.16/x64/lib/python3.8/site-
packages/pandas/core/indexes/base.py:6416, in <listcomp>(.0)
   6414     return type(self).from_tuples(items, names=self.names)
   6415 else:
-> 6416     items = [func(x) for x in self]
   6417     return Index(items, name=self.name, tupleize_cols=False)

Cell In[35], line 1, in <lambda>(c)
----> 1 pd.concat([gh_df[col].apply(pd.Series).rename(lambda c: '_'.join([c,col])) for col in
js_cols],axis=1).head()

TypeError: sequence item 0: expected str instance, int found
```

```
gh_df['actor'].apply(pd.Series).rename(columns=lambda c: '_'.join([c,'actor']))
```

0	10656079	brownsarahm	brownsarahm	https://api.github.com/users/brownsarahm	htt
1	10656079	brownsarahm	brownsarahm	https://api.github.com/users/brownsarahm	htt
2	41898282	github-actions[bot]	github-actions	https://api.github.com/users/github-actions[bot]	htt
3	10656079	brownsarahm	brownsarahm	https://api.github.com/users/brownsarahm	htt
4	41898282	github-actions[bot]	github-actions	https://api.github.com/users/github-actions[bot]	htt
5	10656079	brownsarahm	brownsarahm	https://api.github.com/users/brownsarahm	htt
6	10656079	brownsarahm	brownsarahm	https://api.github.com/users/brownsarahm	htt
7	10656079	brownsarahm	brownsarahm	https://api.github.com/users/brownsarahm	htt
8	41898282	github-actions[bot]	github-actions	https://api.github.com/users/github-actions[bot]	htt
9	10656079	brownsarahm	brownsarahm	https://api.github.com/users/brownsarahm	htt
10	41898282	github-actions[bot]	github-actions	https://api.github.com/users/github-actions[bot]	htt
11	10656079	brownsarahm	brownsarahm	https://api.github.com/users/brownsarahm	htt
12	41898282	github-actions[bot]	github-actions	https://api.github.com/users/github-actions[bot]	htt
13	41898282	github-actions[bot]	github-actions	https://api.github.com/users/github-actions[bot]	htt
14	10656079	brownsarahm	brownsarahm	https://api.github.com/users/brownsarahm	htt
15	10656079	brownsarahm	brownsarahm	https://api.github.com/users/brownsarahm	htt
16	10656079	brownsarahm	brownsarahm	https://api.github.com/users/brownsarahm	htt
17	10656079	brownsarahm	brownsarahm	https://api.github.com/users/brownsarahm	htt
18	41898282	github-actions[bot]	github-actions	https://api.github.com/users/github-actions[bot]	htt
19	10656079	brownsarahm	brownsarahm	https://api.github.com/users/brownsarahm	htt
20	41898282	github-actions[bot]	github-actions	https://api.github.com/users/github-actions[bot]	htt
21	41898282	github-actions[bot]	github-actions	https://api.github.com/users/github-actions[bot]	htt
22	10656079	brownsarahm	brownsarahm	https://api.github.com/users/brownsarahm	htt
23	10656079	brownsarahm	brownsarahm	https://api.github.com/users/brownsarahm	htt
24	41898282	github-actions[bot]	github-actions	https://api.github.com/users/github-actions[bot]	htt
25	10656079	brownsarahm	brownsarahm	https://api.github.com/users/brownsarahm	htt
26	10656079	brownsarahm	brownsarahm	https://api.github.com/users/brownsarahm	htt
27	10656079	brownsarahm	brownsarahm	https://api.github.com/users/brownsarahm	htt
28	41898282	github-actions[bot]	github-actions	https://api.github.com/users/github-actions[bot]	htt
29	10656079	brownsarahm	brownsarahm	https://api.github.com/users/brownsarahm	htt

```
json_cols_df = pd.concat([gh_df[col].apply(pd.Series).rename(columns=lambda c: '_'.join([c,col])) for
col in js_cols],axis=1).head()
```

```
gh_df.columns
```

```
       dtype='object')
```

```
json_cols_df.columns
```

```
Index(['id_actor', 'login_actor', 'display_login_actor', 'gravatar_id_actor',
       'url_actor', 'avatar_url_actor', 'id_repo', 'name_repo', 'url_repo',
       'ref_payload', 'ref_type_payload', 'master_branch_payload',
       'description_payload', 'pusher_type_payload', 'repository_id_payload',
       'push_id_payload', 'size_payload', 'distinct_size_payload',
       'head_payload', 'before_payload', 'commits_payload', 'action_payload',
       'release_payload', 'id_org', 'login_org', 'gravatar_id_org', 'url_org',
       'avatar_url_org'],
      dtype='object')
```

Then we can put the two parts of the data together

```
pd.concat([gh_df[['id','type','public','created_at']],json_cols_df],)
```

0	2.753458e+10	CreateEvent	True	2023-03-07 01:52:36+00:00	NaN	NaN	NaN
1	2.753457e+10	PushEvent	True	2023-03-07 01:52:16+00:00	NaN	NaN	NaN
2	2.742338e+10	PushEvent	True	2023-03-01 18:18:49+00:00	NaN	NaN	NaN
3	2.742331e+10	PushEvent	True	2023-03-01 18:15:43+00:00	NaN	NaN	NaN
4	2.741335e+10	PushEvent	True	2023-03-01 12:07:33+00:00	NaN	NaN	NaN
5	2.741329e+10	CreateEvent	True	2023-03-01 12:05:02+00:00	NaN	NaN	NaN
6	2.741329e+10	ReleaseEvent	True	2023-03-01 12:05:01+00:00	NaN	NaN	NaN
7	2.741326e+10	PushEvent	True	2023-03-01 12:03:45+00:00	NaN	NaN	NaN
8	2.741324e+10	PushEvent	True	2023-03-01 12:02:58+00:00	NaN	NaN	NaN
9	2.741314e+10	PushEvent	True	2023-03-01 11:59:39+00:00	NaN	NaN	NaN
10	2.741294e+10	PushEvent	True	2023-03-01 11:51:18+00:00	NaN	NaN	NaN
11	2.741287e+10	PushEvent	True	2023-03-01 11:48:08+00:00	NaN	NaN	NaN
12	2.740257e+10	PushEvent	True	2023-03-01 02:35:34+00:00	NaN	NaN	NaN
13	2.740257e+10	PushEvent	True	2023-03-01 02:34:56+00:00	NaN	NaN	NaN
14	2.740256e+10	ReleaseEvent	True	2023-03-01 02:34:19+00:00	NaN	NaN	NaN
15	2.740254e+10	CreateEvent	True	2023-03-01 02:33:07+00:00	NaN	NaN	NaN
16	2.740254e+10	PushEvent	True	2023-03-01 02:32:40+00:00	NaN	NaN	NaN
17	2.740253e+10	PushEvent	True	2023-03-01 02:31:45+00:00	NaN	NaN	NaN
18	2.740034e+10	PushEvent	True	2023-02-28 23:48:59+00:00	NaN	NaN	NaN
19	2.740030e+10	PushEvent	True	2023-02-28 23:46:00+00:00	NaN	NaN	NaN
20	2.739872e+10	PushEvent	True	2023-02-28 22:03:16+00:00	NaN	NaN	NaN
21	2.739867e+10	PushEvent	True	2023-02-28 22:00:33+00:00	NaN	NaN	NaN
22	2.739865e+10	PushEvent	True	2023-02-28 21:59:44+00:00	NaN	NaN	NaN
23	2.739861e+10	PushEvent	True	2023-02-28 21:57:16+00:00	NaN	NaN	NaN
24	2.734234e+10	PushEvent	True	2023-02-26 23:06:26+00:00	NaN	NaN	NaN
25	2.734231e+10	PushEvent	True	2023-02-26 23:03:18+00:00	NaN	NaN	NaN
26	2.730273e+10	ReleaseEvent	True	2023-02-24 04:27:10+00:00	NaN	NaN	NaN
27	2.730272e+10	CreateEvent	True	2023-02-24 04:25:49+00:00	NaN	NaN	NaN

28	2.730143e+10	PushEvent	True	2023-02-24 02:31:47+00:00	NaN	NaN	NaN
29	2.730139e+10	PushEvent	True	2023-02-24 02:28:44+00:00	NaN	NaN	NaN
0	NaN	NaN	NaN	NaT	10656079.0	brownsarahm	brownsarahm
1	NaN	NaN	NaN	NaT	10656079.0	brownsarahm	brownsarahm
2	NaN	NaN	NaN	NaT	41898282.0	github-actions[bot]	github-actions
3	NaN	NaN	NaN	NaT	10656079.0	brownsarahm	brownsarahm
4	NaN	NaN	NaN	NaT	41898282.0	github-actions[bot]	github-actions

35 rows × 32 columns

and finally save this

```
gh_df_clean = pd.concat([gh_df[['id','type','public','created_at']],json_cols_df],axis=1)
gh_df_clean.head()
```

	id	type	public	created_at	id_actor	login_actor	display_login_actor	gravatar_
0	27534576446	CreateEvent	True	2023-03-07 01:52:36+00:00	10656079.0	brownsarahm	brownsarahm	
1	27534572112	PushEvent	True	2023-03-07 01:52:16+00:00	10656079.0	brownsarahm	brownsarahm	
2	27423376182	PushEvent	True	2023-03-01 18:18:49+00:00	41898282.0	github-actions[bot]	github-actions	
3	27423307938	PushEvent	True	2023-03-01 18:15:43+00:00	10656079.0	brownsarahm	brownsarahm	
4	27413353459	PushEvent	True	2023-03-01 12:07:33+00:00	41898282.0	github-actions[bot]	github-actions	

5 rows × 32 columns

If we want to analyze this data, this is a good place to save it to disk and start an analysis in separate notebook.

```
gh_df_clean.to_csv('gh_events_unpacked.csv')
```

8.7. Questions After Class

8.7.1. How the apply function works/use cases?

A4 will give you some examples, espeically the airline dataset. We will also keep seing it come up as we manipulate data more.

the apply docs have tiny examples that help illustrate what it does and some of how it works. The pandas faq has a section on apply and similar methods that gives some more use cases.

8.7.2. Is there a better way to see how many missing values?

There are lots of ways. All are fine. We used `info` in class because I was trying to use the way we had already seen. Info focuses on how many values are *present* instead of what is missing because it makes more sense in most cases. The more common question is: are there enough values to make decisions with?

`pandas.isna` not a Data Value method (which would be accessed like `pandas.Value.FunctionName`). This means we use it like

```
value_to_test = 4
pd.isna(value_to_test)
```

```
False
```

Try it Yourself

pass different values like: `False`, `np.nan` (also `import numpy as np`) and, `pd.NA`, `hello` to this function

```
help(pd.isna)
```

```
isna(obj: 'object') -> 'bool | npt.NDArray[np.bool_] | NDFrame'
    Detect missing values for an array-like object.

    This function takes a scalar or array-like object and indicates
    whether values are missing (``NaN`` in numeric arrays, ``None`` or ``NaN``
    in object arrays, ``NaT`` in datetimelike).

    Parameters
    ----------
    obj : scalar or array-like
        Object to check for null or missing values.

    Returns
    -------
    bool or array-like of bool
        For scalar input, returns a scalar boolean.
        For array input, returns an array of boolean indicating whether each
        corresponding element is missing.

    See Also
    --------
    notna : Boolean inverse of pandas.isna.
    Series.isna : Detect missing values in a Series.
    DataFrame.isna : Detect missing values in a DataFrame.
    Index.isna : Detect missing values in an Index.

    Examples
    --------
    Scalar arguments (including strings) result in a scalar boolean.

    >>> pd.isna('dog')
    False

    >>> pd.isna(pd.NA)
    True

    >>> pd.isna(np.nan)
    True

    ndarrays result in an ndarray of booleans.

    >>> array = np.array([[1, np.nan, 3], [4, 5, np.nan]])
    >>> array
    array([[ 1., nan,  3.],
           [ 4.,  5., nan]])
    >>> pd.isna(array)
    array([[False,  True, False],
           [False, False,  True]])

    For indexes, an ndarray of booleans is returned.

    >>> index = pd.DatetimeIndex(["2017-07-05", "2017-07-06", None,
    ...                           "2017-07-08"])
    >>> index
    DatetimeIndex(['2017-07-05', '2017-07-06', 'NaT', '2017-07-08'],
                  dtype='datetime64[ns]', freq=None)
    >>> pd.isna(index)
    array([False, False,  True, False])

    For Series and DataFrame, the same type is returned, containing booleans.

    >>> df = pd.DataFrame([['ant', 'bee', 'cat'], ['dog', None, 'fly']])
    >>> df
         0     1    2
    0  ant   bee  cat
    1  dog  None  fly
    >>> pd.isna(df)
           0      1      2
    0  False  False  False
    1  False   True  False

    >>> pd.isna(df[1])
    0    False
    1     True
    Name: 1, dtype: bool
```

The docstring says that it returns “bool or array-like of bool” but if we go to the website docs that have more examples, we can find out what that it will return a DataFrame if we pass it a DataFrame. Then we can use the `pandas.DataFrame.sum` method.

```
pd.isna(coffee_df_clean).sum()
```

```
country_of_origin             1
farm_name                   356
lot_number                 1041
mill                        310
ico_number                  146
company                     209
altitude                    223
region                       57
producer                    230
number_of_bags                0
bag_weight                    0
in_country_partner            0
harvest_year                 47
grading_date                  0
owner_1                       7
variety                     201
processing_method           152
aroma                         0
flavor                        0
aftertaste                    0
acidity                       0
body                          0
balance                       0
uniformity                    0
clean_cup                     0
sweetness                     0
cupper_points                 0
total_cup_points              0
moisture                      0
category_one_defects          0
quakers                       1
color                       216
category_two_defects          0
expiration                    0
certification_body            0
certification_address         0
certification_contact         0
unit_of_measurement           0
altitude_low_meters         227
altitude_high_meters        227
altitude_mean_meters        227
unit_of_measurement_long      0
lot_number_clean              0
dtype: int64
```

8.7.3. in `col_name_mapper = {col_name:col_name.lower().replace('.','_') for col_name in coffee_df.columns}` what is the `{}` for?

This is called a dictionary comphrehension. It is very similar to a list comprehension. It is one of the defined ways to build a `dict` type object

We also saw one when we looked at different types in a previous class.

```
{char:i for i,char in enumerate('abcde')}
```

```
{'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4}
```

`enumerate` is a built in function that iterates over items in an iterable type(list-like) and pops the each value paired with its index within the structure.

This way we get each character and it's position. We could use this as follows

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

```
  Cell In[47], line 2
    alpha_data = ['a','d','e','c','b',']
                                       ^
SyntaxError: EOL while scanning string literal
```

9.1. Announcements

Assignment 3 is graded, but only today, so A4 is extended until tomorrow so that you can incorporate your A3 feedback into A4 (in particular, extending you EDA if you did not earn level 2 for summarize and visualize in A3).

Your achievement trackers are updated.

Take a few minutes to think about the following questions and make a few notes for yourself whereever you need them to be (a planner, calendar, etc).

1. What achievements have you earned?
2. Does your tracking repo seem accurate to what you've done? If not, make an issue to ask about it.
3. Are you on track to earn the grade you want in this class?
4. If not, what will you need to do (respond more in class, submit more assignments, use your portfolio to catch up) to get back on track?
5. If you are on track and you want to earn above a B, take a minute to think about your portfolio. (tip: post an idea as an issue to get early feedback and help shaping your idea)

9.2. Merging Data

Focus this week is on how to programmatically combine sources of data

We will start by looking at combining multiple tabular data formats and see how to get data from other sources.

```
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 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()
```

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 `columns` is an attribute, so it does not need `()`

```
df_19.columns
```

```
Index(['PLAYER_NAME', 'TEAM_ID', 'PLAYER_ID', 'SEASON'], dtype='object')
```

Let's make note of the shape of each

```
df_18.shape, df_19.shape
```

```
((748, 3), (626, 4))
```

9.2.1. 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

```
pd.concat([df_18,df_19]).shape
```

```
(1374, 4)
```

Note that this has the maximum number of columns (because both had some overlapping columns) and the total number of rows.

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')
```

	TEAM_ID_x	PLAYER_ID	SEASON_x	PLAYER_NAME	TEAM_ID_y	SEASON_y
0	1610612761	202695	2018	Kawhi Leonard	1610612746	2019
1	1610612761	1627783	2018	Pascal Siakam	1610612761	2019
2	1610612761	201188	2018	Marc Gasol	1610612761	2019
3	1610612763	201188	2018	Marc Gasol	1610612761	2019
4	1610612761	201980	2018	Danny Green	1610612747	2019
...	...	...	...	...	...	...
533	1610612760	203583	2018	Abdul Gaddy	1610612760	2019
534	1610612760	203460	2018	Andre Roberson	1610612760	2019
535	1610612755	203658	2018	Norvel Pelle	1610612755	2019
536	1610612741	1627756	2018	Denzel Valentine	1610612741	2019
537	1610612754	203912	2018	C.J. Wilcox	1610612754	2019

538 rows × 6 columns

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.

```
df_1819_inner = pd.merge(df_18,df_19,on='PLAYER_ID',suffixes=('_2018','_2019'))
df_1819_inner.shape
```

```
(538, 6)
```

By default, this uses an *inner* merge, so we get the players that are in both datasets only. If we want to see differences, we need another type of merge.

Some players still appear twice, because they were in one of the datsets twice, this happens when a player plays for two team in one season.

9.2.3. 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.

```
df_1819_outer = pd.merge(df_18,df_19,how='outer',on='PLAYER_ID',suffixes=('_2018','_2019'),)
df_1819_outer
```

0	1.610613e+09	202695	2018.0	Kawhi Leonard	1.610613e+09	2019.0
1	1.610613e+09	1627783	2018.0	Pascal Siakam	1.610613e+09	2019.0
2	1.610613e+09	201188	2018.0	Marc Gasol	1.610613e+09	2019.0
3	1.610613e+09	201188	2018.0	Marc Gasol	1.610613e+09	2019.0
4	1.610613e+09	201980	2018.0	Danny Green	1.610613e+09	2019.0
...	...	...	...	...	...	...
922	NaN	1629097	NaN	Terry Larrier	1.610613e+09	2019.0
923	NaN	203461	NaN	Anthony Bennett	1.610613e+09	2019.0
924	NaN	203906	NaN	Devyn Marble	1.610613e+09	2019.0
925	NaN	1629755	NaN	Hassani Gravett	1.610613e+09	2019.0
926	NaN	1629721	NaN	JaKeenan Gant	1.610613e+09	2019.0

927 rows × 6 columns

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.

```
df_1819_inner.dtypes
```

```
TEAM_ID_2018     int64
PLAYER_ID        int64
SEASON_2018      int64
PLAYER_NAME     object
TEAM_ID_2019     int64
SEASON_2019      int64
dtype: object
```

```
df_1819_outer.dtypes
```

```
TEAM_ID_2018    float64
PLAYER_ID         int64
SEASON_2018     float64
PLAYER_NAME      object
TEAM_ID_2019    float64
SEASON_2019     float64
dtype: object
```

nan is a float

```
import numpy as np
type(np.nan)
```

```
float
```

Back the the question, we can also use a left merge. To pick out those rows:

```
df_1819left = pd.merge(df_18,df_19,how='left',on='PLAYER_ID',suffixes=('_2018','_2019'),)
df_18only = df_1819left[df_1819left['SEASON_2019'].isna()]
df_18only
```

9	1610612761	202391	2018	NaN	NaN	NaN
11	1610612761	201975	2018	NaN	NaN	NaN
18	1610612744	101106	2018	NaN	NaN	NaN
23	1610612744	2733	2018	NaN	NaN	NaN
24	1610612744	201973	2018	NaN	NaN	NaN
...	...	...	...	...	...	...
749	1610612752	1629246	2018	NaN	NaN	NaN
750	1610612748	1629159	2018	NaN	NaN	NaN
751	1610612762	1629163	2018	NaN	NaN	NaN
752	1610612743	1629150	2018	NaN	NaN	NaN
753	1610612738	1629167	2018	NaN	NaN	NaN

216 rows × 6 columns

```
len(df_18only['PLAYER_ID'].unique())
```

```
178
```

```
df_18only['PLAYER_ID'].value_counts()
```

```
1629150    4
202325     3
201160     3
1628393    3
202328     3
          ..
1628515    1
1627816    1
1628979    1
2736       1
1629167    1
Name: PLAYER_ID, Length: 178, dtype: int64
```

9.3. Getting Data from Databases

9.3.1. 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.

> 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

9.3.2. 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 type of database.

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 0x7f0a1cad2fa0>)
```

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 0x7f0a1ca65e30>
```

Then we use `fetchall` to get the the results of the query.

```
cursor.fetchall()
```

```
('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()
```

```
[]
```

We can run another query with execute then fetch that result. This query gives us the column names.

The schema of a database is the description of its setup and layout. The `*` means to get all.

```
cursor.execute("SELECT * FROM INFORMATION_SCHEMA.COLUMNS ")
```

```
---------------------------------------------------------------------------
OperationalError                          Traceback (most recent call last)
Cell In[25], line 1
----> 1 cursor.execute("SELECT * FROM INFORMATION_SCHEMA.COLUMNS ")

OperationalError: no such table: INFORMATION_SCHEMA.COLUMNS
```

Then we use `fetchall` to get the the results of the query.

```
cursor.fetchall()
```

```
[]
```

9.4. 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)
```

0	teams
1	conferences
2	playerGameStats2018
3	playerGameStats2019
4	teamGameStats2018
5	teamGameStats2019
6	playerTeams2018
7	playerTeams2019
8	teamDailyRankings2018
9	teamDailyRankings2019
10	playerNames

```
pd.read_sql('SELECT * FROM teams',conn).head(1)
```

	index	LEAGUE_ID	TEAM_ID	MIN_YEAR	MAX_YEAR	ABBREVIATION	NICKNAME	YEARFOUNDED	
0	0	0	1610612737	1949	2019	ATL	Hawks	1949	A

9.4.1. 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
dtype: int64
```

and we can get the player's name from the player name **remember our first query told us all the tables**

	PLAYER_NAME
0	Emanuel Terry

9.4.2. 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
```

Note

You can do the merging in SQL and pull only the merged table technically, but I use pandas more than sql so I did the minimum in SQL and the rest in pandas.

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
```

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.

```
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.unique).apply(len)
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)
```

Again 135 is more than 119.

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.

9.5. Questions After Class

9.5.1. Is there a max DB size?

Generally, no. In specific instances, yes. For example, MSFT SQL Server has a max size of 524,272 terabytes.

The most important limit here is realy that the cmoputer you are working on will have limits on how much data you can pull from the database into local RAM.

9.5.3. how do you learn more about different quieries you can use for sql?

Wizard zines has a good reference, but it is not free. I have some of their other work though and it is all high quality. this preview is especially helpful for me If the cost is prohibitive for you, but the preview of this looks like something you would like, send me an e-mail.

This cheatsheet is also good.

9.5.4. What other SQL 'keywords' in the queries are there? ex: SELECT, FROM, WHERE

quick reference

10. Web Scraping

```
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, then restart your kernel (Kernel menu, choose restart)

```
pip install beautifulsoup4
```

10.1. 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

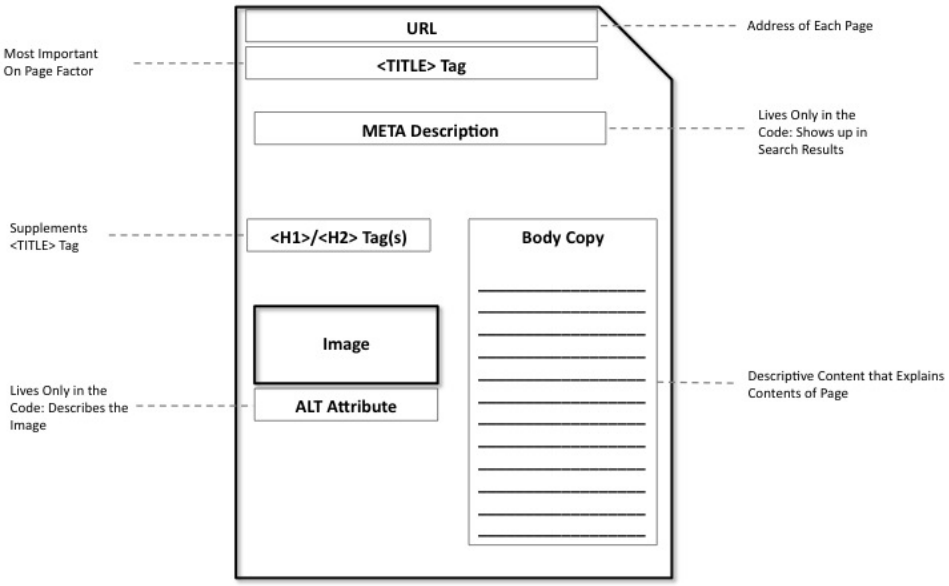
Note

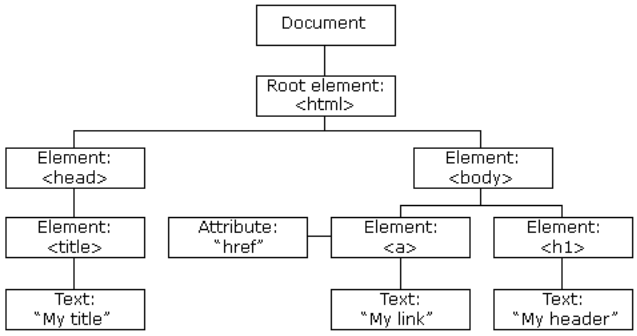
This has a long output, so I hid it by default, but you can view it

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.

10.2. Everything is Data

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





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

10.3. 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.

> **Note**
>
> the inspect link goes to instructions for different browsers

> **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

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">('
```

Note

here I suppressed th output in class by looking only at the first few characters

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

```
cs_people
```

▶ Show code cell output

10.3.1. Looking at tags

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

```
cs_people.a
```

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

```
cs_people.div
```

```
<a class="skip-link screen-reader-text" href="#content">Skip to content</a>
<div id="masthead">
<header class="site-header" id="brandbar" role="banner">
<div id="identity-print"><img alt="University of Rhode Island" src="https://web.uri.edu/cs/wp-
content/themes/uri-modern/images/logo-print.png" width="120px"/></div>
<div id="globalsearch" role="search">
<input aria-label="Toggle visibility of the search box." id="gsform-toggle" role="presentation"
type="checkbox"/>
<label for="gsform-toggle" id="gsform"><span>Search</span></label>
<form action="https://www.uri.edu/search" id="gs" method="get" name="global_general_search_form">
<input name="cx" type="hidden" value="016863979916529535900:17qai8akniu">
<input name="cof" type="hidden" value="FORID:11"/>
<label for="gs-query" id="gs-query-label">Searchbox</label>
<input id="gs-query" name="q" placeholder="Search" role="searchbox" type="text" value=""/>
<input class="searchsubmit" id="gs-submit" name="searchsubmit" type="submit" value="Search"/>
</input></form>
</div>
<div id="globalbanner-wrapper">
<div id="globalbanner">
<a href="https://www.uri.edu/" title="University of Rhode Island"><div id="identity">University of
Rhode Island</div></a>
<div id="gateways">
<input aria-label="Open the audience gateways menu when browsing on mobile" id="gateways-toggle"
role="presentation" type="checkbox"/>
<label for="gateways-toggle" id="gateways-label"><span>You</span></label>
<ul id="gateways-menu" role="menu">
<li><a href="https://www.uri.edu/gateway/future-students" role="menuitem">Future Students</a></li>
<li><a href="https://www.uri.edu/gateway/students" role="menuitem">Students</a></li>
<li><a href="https://www.uri.edu/gateway/faculty" role="menuitem">Faculty</a></li>
<li><a href="https://www.uri.edu/gateway/staff" role="menuitem">Staff</a></li>
<li><a href="https://www.uri.edu/gateway/families" role="menuitem">Parents and Families</a></li>
<li><a href="https://www.uri.edu/gateway/alumni" role="menuitem">Alumni</a></li>
<li><a href="https://www.uri.edu/gateway/community" role="menuitem">Community</a></li>
</ul>
</div>
</div>
</div>
</header><!-- #brandbar -->
<header id="siteheader">
<div class="light" id="sitebanner">
<div id="sb-backdrop">
<div id="sb-background-image" style="background-image:url(https://web.uri.edu/wp-
content/uploads/sites/1531/cropped-Big-Data.jpg)"></div>
<div id="sb-screen"></div>
</div>
<div id="sitebranding">
<div id="siteidentity">
<h1 class="site-title">
<a href="https://web.uri.edu/cs/" rel="home">
                        Department of Computer Science and Statistics           </a>
</h1>
<h2 class="site-description">College of Arts and Sciences</h2>
</div>
<div id="sitesocial">
</div>
</div><!-- #sitebranding -->
</div><!-- #sitebanner -->
<div class="content-width" id="navigation">
<nav aria-label="Breadcrumb" id="breadcrumbs">
<ol><li><a href="https://www.uri.edu/">URI</a></li><li><a href="https://web.uri.edu/artsci">Arts and
Sciences</a></li><li><a href="https://web.uri.edu/cs">Department of Computer Science and
Statistics</a></li><li aria-current="page">People</li></ol></nav>
<div id="localnav">
<section class="cl-wrapper cl-menu-wrapper"><div class="cl-menu" data-name="Site Menu" data-show-
title="0" id="cl-localnav"><ul class="cl-menu-list cl-menu-list-no-js" id="menu-navigation"><li
class="menu-item menu-item-type-custom menu-item-object-custom menu-item-home menu-item-8243"
id="menu-item-8243"><a href="https://web.uri.edu/cs" title="
">Home</a></li>
<li class="menu-item menu-item-type-post_type menu-item-object-page menu-item-8255" id="menu-item-
8255"><a href="https://web.uri.edu/cs/about/" title="
">About</a></li>
<li class="menu-item menu-item-type-post_type menu-item-object-page menu-item-8806" id="menu-item-
8806"><a href="https://web.uri.edu/cs/academics/">Academics</a></li>
<li class="menu-item menu-item-type-post_type menu-item-object-page current-menu-item page_item page-
item-629 current_page_item menu-item-8257" id="menu-item-8257"><a aria-current="page"
href="https://web.uri.edu/cs/people/" title="                                                    ">People</a>
</li>
<li class="menu-item menu-item-type-post_type menu-item-object-page menu-item-9661" id="menu-item-
9661"><a href="https://web.uri.edu/cs/research/">Research</a></li>
<li class="menu-item menu-item-type-post_type menu-item-object-page menu-item-10871" id="menu-item-
10871"><a href="https://web.uri.edu/cs/news-and-events/">News and Events</a></li>
<li class="menu-item menu-item-type-post_type menu-item-object-page menu-item-8267" id="menu-item-
8267"><a href="https://web.uri.edu/cs/contact/" title="
">Contact</a></li>
</ul></div></section></div>
```

```
</div>
<div class="site-content" id="content">
<main class="site-main" id="main" role="main">
<article class="post-629 page type-page status-publish hentry" id="post-629">
<div class="entry-content">
<h1>People</h1>
<section class="cl-wrapper cl-menu-wrapper"><div class="cl-menu" data-name="people" data-show-
title="0" id=""><ul class="cl-menu-list cl-menu-list-no-js" id="menu-people"><li class="menu-item
menu-item-type-post_type menu-item-object-page current-menu-item page_item page-item-629
current_page_item menu-item-8737" id="menu-item-8737"><a aria-current="page"
href="https://web.uri.edu/cs/people/">Faculty</a></li>
<li class="menu-item menu-item-type-post_type menu-item-object-page menu-item-8746" id="menu-item-
8746"><a href="https://web.uri.edu/cs/people/staff/">Staff</a></li>
<li class="menu-item menu-item-type-post_type menu-item-object-page menu-item-11844" id="menu-item-
11844"><a href="https://web.uri.edu/cs/people/faculty-emeriti/">Faculty Emeriti</a></li>
</ul></div></section>
<h2>Full-time Faculty</h2>

<div class="uri-people-tool cl-tiles halves"><div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>
<a href="https://web.uri.edu/cs/meet/marco-alvarez/"><img alt="" class="u-photo wp-post-image"
height="120" loading="lazy" src="https://web.uri.edu/cs/wp-content/uploads/sites/1531/marco-
alvarez.png" width="120"/></a>
</figure>
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/marco-alvarez/">Marco Alvarez</a></h3>
</div>
</header>
<div class="inside">
<p class="people-title p-job-title">Associate Professor | Director of Graduate Studies</p>
<p class="people-department">Computer Science</p>
<p class="people-misc"><span class="p-tel">401.874.5009</span> – <a class="u-email"
href="mailto:malvarez@uri.edu">malvarez@uri.edu</a></p>
<div style="clear:both;"></div>
</div>
</div>
<div class="peopleitem h-card">
<header>
<div class="header">
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/samantha-armenti/">Samantha Armenti</a></h3>
</div>
</header>
<div class="inside">
<p class="people-title p-job-title">Assistant Teaching Professor</p>
<p class="people-department">Computer Science</p>
<p class="people-misc"><a class="u-email" href="mailto:sarmenti@uri.edu ">sarmenti@uri.edu </a></p>
<div style="clear:both;"></div>
</div>
</div>
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>
<a href="https://web.uri.edu/cs/meet/sarah-brown/"><img alt="" class="u-photo wp-post-image"
height="300" loading="lazy" sizes="(max-width: 300px) 100vw, 300px" src="https://web.uri.edu/cs/wp-
content/uploads/sites/1531/Sarah-Brown.png" srcset="https://web.uri.edu/cs/wp-
content/uploads/sites/1531/Sarah-Brown.png 300w, https://web.uri.edu/cs/wp-
content/uploads/sites/1531/Sarah-Brown-150x150.png 150w" width="300"/></a>
</figure>
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/sarah-brown/">Sarah Brown</a></h3>
</div>
</header>
<div class="inside">
<p class="people-title p-job-title">Assistant Professor</p>
<p class="people-department">Computer Science</p>
<p class="people-misc"><a class="u-email" href="mailto:brownsarahm@uri.edu">brownsarahm@uri.edu</a>
</p>
<div style="clear:both;"></div>
</div>
</div>
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>
<a href="https://web.uri.edu/cs/meet/michael-conti/"><img alt="" class="u-photo wp-post-image"
height="2475" loading="lazy" sizes="(max-width: 2560px) 100vw, 2560px" src="https://web.uri.edu/cs/wp-
content/uploads/sites/1531/mike-conti-scaled.jpg" srcset="https://web.uri.edu/cs/wp-
content/uploads/sites/1531/mike-conti-scaled.jpg 2560w, https://web.uri.edu/cs/wp-
content/uploads/sites/1531/mike-conti-300x290.jpg 300w, https://web.uri.edu/cs/wp-
content/uploads/sites/1531/mike-conti-1024x990.jpg 1024w, https://web.uri.edu/cs/wp-
content/uploads/sites/1531/mike-conti-768x743.jpg 768w, https://web.uri.edu/cs/wp-
content/uploads/sites/1531/mike-conti-1536x1485.jpg 1536w, https://web.uri.edu/cs/wp-
content/uploads/sites/1531/mike-conti-2048x1980.jpg 2048w, https://web.uri.edu/cs/wp-
content/uploads/sites/1531/mike-conti-364x352.jpg 364w, https://web.uri.edu/cs/wp-
```

content/uploads/sites/1531/mike-conti-1280x1280.jpg 1280w, https://web.uri.edu/cs/wp-content/uploads/sites/1531/mike-conti-2000x1934.jpg 2000w" width="2560"/>
</figure>
<h3 class="p-name">Michael Conti</h3>

</header>

Assistant Teaching Professor
Computer Science
michaelconti@uri.edu

<header>

<figure>
<img alt="" class="u-photo wp-post-image"
height="219" loading="lazy" sizes="(max-width: 219px) 100vw, 219px" src="https://web.uri.edu/cs/wp-content/uploads/sites/1531/noah-daniels-web.png" srcset="https://web.uri.edu/cs/wp-content/uploads/sites/1531/noah-daniels-web.png 219w, https://web.uri.edu/cs/wp-content/uploads/sites/1531/noah-daniels-web-150x150.png 150w" width="219"/>
</figure>
<h3 class="p-name">Noah Daniels</h3>

</header>

Assistant Professor
Computer Science
noah_daniels@uri.edu

<header>

<figure>
<img alt="" class="u-photo wp-post-image"
height="126" loading="lazy" src="https://web.uri.edu/cs/wp-content/uploads/sites/1531/lisa-dipippo-web.jpg" width="125"/>
</figure>
<h3 class="p-name">Lisa DiPippo</h3>

</header>

Professor | Chair
Computer Science
ldipippo@uri.edu

<header>

<figure>
<img alt="" class="u-photo wp-post-image"
height="125" loading="lazy" src="https://web.uri.edu/cs/wp-content/uploads/sites/1531/victor-fay-wolfe-web.jpg" width="125"/>
</figure>
<h3 class="p-name">Victor Fay-Wolfe</h3>

</header>

Professor
Computer Science
vfaywolfe@uri.edu

<header>

<h3 class="p-name">Lutz Hamel</h3>

</header>

Associate Professor
Computer Science
lutzhamel@uri.edu

```
<figure>
<a href="https://web.uri.edu/cs/meet/abdeltawab-hendawi/"><img alt="" class="u-photo wp-post-image"
height="885" loading="lazy" sizes="(max-width: 1000px) 100vw, 1000px" src="https://web.uri.edu/cs/wp-
content/uploads/sites/1531/0-1.jpeg" srcset="https://web.uri.edu/cs/wp-content/uploads/sites/1531/0-
1.jpeg 1000w, https://web.uri.edu/cs/wp-content/uploads/sites/1531/0-1-300x266.jpeg 300w,
https://web.uri.edu/cs/wp-content/uploads/sites/1531/0-1-768x680.jpeg 768w, https://web.uri.edu/cs/wp-
content/uploads/sites/1531/0-1-364x322.jpeg 364w, https://web.uri.edu/cs/wp-
content/uploads/sites/1531/0-1-500x443.jpeg 500w" width="1000"/></a>
</figure>
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/abdeltawab-hendawi/">Abdeltawab Hendawi</a>
</h3>
</div>
</header>
<div class="inside">
<p class="people-title p-job-title">Assistant Professor</p>
<p class="people-department">Data Science | Computer Science</p>
<p class="people-misc"><span class="p-tel">401.874.5738</span> – <a class="u-email"
href="mailto:hendawi@uri.edu">hendawi@uri.edu</a></p>
<div style="clear:both;"></div>
</div>
</div>
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>
<a href="https://web.uri.edu/cs/meet/jean-yves-herve/"><img alt="" class="u-photo wp-post-image"
height="299" loading="lazy" sizes="(max-width: 300px) 100vw, 300px" src="https://web.uri.edu/cs/wp-
content/uploads/sites/1531/jean-yves-herve-web.jpg" srcset="https://web.uri.edu/cs/wp-
content/uploads/sites/1531/jean-yves-herve-web.jpg 300w, https://web.uri.edu/cs/wp-
content/uploads/sites/1531/jean-yves-herve-web-150x150.jpg 150w" width="300"/></a>
</figure>
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/jean-yves-herve/">Jean-Yves Hervé</a></h3>
</div>
</header>
<div class="inside">
<p class="people-title p-job-title">Associate Professor</p>
<p class="people-department">Computer Science</p>
<p class="people-misc"><a class="u-email" href="mailto:jyh@cs.uri.edu">jyh@cs.uri.edu</a></p>
<div style="clear:both;"></div>
</div>
</div>
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>
<a href="https://web.uri.edu/cs/meet/natallia-katenka/"><img alt="" class="u-photo wp-post-image"
height="200" loading="lazy" sizes="(max-width: 200px) 100vw, 200px" src="https://web.uri.edu/cs/wp-
content/uploads/sites/1531/natalia-katenka-web.jpg" srcset="https://web.uri.edu/cs/wp-
content/uploads/sites/1531/natalia-katenka-web.jpg 200w, https://web.uri.edu/cs/wp-
content/uploads/sites/1531/natalia-katenka-web-150x150.jpg 150w" width="200"/></a>
</figure>
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/natallia-katenka/">Natallia Katenka</a></h3>
</div>
</header>
<div class="inside">
<p class="people-title p-job-title">Associate Professor | Director of Undergraduate Studies</p>
<p class="people-department">Statistics</p>
<p class="people-misc"><a class="u-email" href="mailto:nkatenka@uri.edu">nkatenka@uri.edu</a></p>
<div style="clear:both;"></div>
</div>
</div>
<div class="peopleitem h-card">
<header>
<div class="header">
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/soheyb-kouider/">Soheyb Kouider</a></h3>
</div>
</header>
<div class="inside">
<p class="people-title p-job-title">Associate Teaching Professor</p>
<p class="people-department">Statistics</p>
<p class="people-misc"><span class="p-tel">401.874.2562</span> – <a class="u-email"
href="mailto:soheyb@uri.edu">soheyb@uri.edu</a></p>
<div style="clear:both;"></div>
</div>
</div>
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>
<a href="https://web.uri.edu/cs/meet/edmund-lamagna/"><img alt="" class="u-photo wp-post-image"
height="150" loading="lazy" src="https://web.uri.edu/cs/wp-content/uploads/sites/1531/edmund-lamagna-
web.jpg" width="150"/></a>
</figure>
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/edmund-lamagna/">Edmund Lamagna</a></h3>
</div>
```

```
<p class="people-title p-job-title">Professor</p>
<p class="people-department">Computer Science</p>
<p class="people-misc"><a class="u-email" href="mailto:eal@cs.uri.edu">eal@cs.uri.edu</a></p>
<div style="clear:both;"></div>
</div>
</div>
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>
<a href="https://web.uri.edu/cs/meet/indrani-mandal/"><img alt="" class="u-photo wp-post-image"
height="280" loading="lazy" sizes="(max-width: 278px) 100vw, 278px" src="https://web.uri.edu/cs/wp-
content/uploads/sites/1531/0.jpeg" srcset="https://web.uri.edu/cs/wp-content/uploads/sites/1531/0.jpeg
278w, https://web.uri.edu/cs/wp-content/uploads/sites/1531/0-150x150.jpeg 150w" width="278"/></a>
</figure>
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/indrani-mandal/">Indrani Mandal</a></h3>
</div>
</header>
<div class="inside">
<p class="people-title p-job-title">Assistant Teaching Professor</p>
<p class="people-department">Computer Science</p>
<p class="people-misc"><a class="u-email" href="mailto:indrani_mandal@uri.edu ">indrani_mandal@uri.edu
</a></p>
<div style="clear:both;"></div>
</div>
</div>
<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"
height="1600" loading="lazy" sizes="(max-width: 1600px) 100vw, 1600px" src="https://web.uri.edu/cs/wp-
content/uploads/sites/1531/Gavino_headshot1-e1610666088381.jpg" srcset="https://web.uri.edu/cs/wp-
content/uploads/sites/1531/Gavino_headshot1-e1610666088381.jpg 1600w, https://web.uri.edu/cs/wp-
content/uploads/sites/1531/Gavino_headshot1-e1610666088381-300x300.jpg 300w,
https://web.uri.edu/cs/wp-content/uploads/sites/1531/Gavino_headshot1-e1610666088381-1024x1024.jpg
1024w, https://web.uri.edu/cs/wp-content/uploads/sites/1531/Gavino_headshot1-e1610666088381-
150x150.jpg 150w, https://web.uri.edu/cs/wp-content/uploads/sites/1531/Gavino_headshot1-
e1610666088381-768x768.jpg 768w, https://web.uri.edu/cs/wp-
content/uploads/sites/1531/Gavino_headshot1-e1610666088381-1536x1536.jpg 1536w,
https://web.uri.edu/cs/wp-content/uploads/sites/1531/Gavino_headshot1-e1610666088381-364x364.jpg 364w,
https://web.uri.edu/cs/wp-content/uploads/sites/1531/Gavino_headshot1-e1610666088381-500x500.jpg 500w,
https://web.uri.edu/cs/wp-content/uploads/sites/1531/Gavino_headshot1-e1610666088381-1000x1000.jpg
1000w, https://web.uri.edu/cs/wp-content/uploads/sites/1531/Gavino_headshot1-e1610666088381-
1280x1280.jpg 1280w" width="1600"/></a>
</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 |  Statistics Section Head | Director of
Graduate Studies</p>
<p class="people-department">Statistics</p>
<p class="people-misc"><span class="p-tel">401.874.4388</span> – <a class="u-email"
href="mailto:gpuggioni@uri.edu">gpuggioni@uri.edu</a></p>
<div style="clear:both;"></div>
</div>
</div>
<div class="peopleitem h-card">
<header>
<div class="header">
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/jonathan-schrader/">Jonathan Schrader</a></h3>
</div>
</header>
<div class="inside">
<p class="people-title p-job-title">Assistant Teaching Professor</p>
<p class="people-department">Computer Science</p>
<p class="people-misc"><a class="u-email"
href="mailto:jonathan.schrader@uri.edu">jonathan.schrader@uri.edu</a></p>
<div style="clear:both;"></div>
</div>
</div>
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>
<a href="https://web.uri.edu/cs/meet/krishna-venkatasubramanian/"><img alt="" class="u-photo wp-post-
image" height="2703" loading="lazy" sizes="(max-width: 2437px) 100vw, 2437px"
src="https://web.uri.edu/cs/wp-content/uploads/sites/1531/KV_biopic_2.jpg"
srcset="https://web.uri.edu/cs/wp-content/uploads/sites/1531/KV_biopic_2.jpg 2437w,
https://web.uri.edu/cs/wp-content/uploads/sites/1531/KV_biopic_2-270x300.jpg 270w,
https://web.uri.edu/cs/wp-content/uploads/sites/1531/KV_biopic_2-768x852.jpg 768w,
https://web.uri.edu/cs/wp-content/uploads/sites/1531/KV_biopic_2-923x1024.jpg 923w,
https://web.uri.edu/cs/wp-content/uploads/sites/1531/KV_biopic_2-364x404.jpg 364w,
https://web.uri.edu/cs/wp-content/uploads/sites/1531/KV_biopic_2-500x555.jpg 500w,
```

https://web.uri.edu/cs/wp-content/uploads/sites/1531/KV_bioPic_2-2000x2218.jpg 2000w" width="2437" />

</figure>
<h3 class="p-name">Krishna Venkatasubramanian</h3>

</header>

Assistant Professor
Computer Science
krish@uri.edu

<header>

<figure>

</figure>
<h3 class="p-name">Jing Wu</h3>

</header>

Associate Professor
Statistics
401.874.4504 – jing_wu@uri.edu

<header>

<figure>

</figure>
<h3 class="p-name">Yichi Zhang</h3>

</header>

Assistant Professor
Statistics
yichizhang@uri.edu

<header>

<figure>

</figure>
<h3 class="p-name">Guangyu Zhu</h3>

</header>

Assistant Professor
Statistics
guangyuzhu@uri.edu

<h2>Adjunct Faculty and Limited Join Appointments</h2>

<header>

<figure>
<img alt="" class="u-photo wp-post-image" height="966" loading="lazy" sizes="(max-width: 966px) 100vw, 966px" src="https://web.uri.edu/cs/wp-

content/uploads/sites/1531/Ashley-Buchanan-500x500.jpg 500w, https://web.uri.edu/cs/wp-content/uploads/sites/1531/Ashley-Buchanan-150x150.jpg 150w, https://web.uri.edu/cs/wp-content/uploads/sites/1531/Ashley-Buchanan-768x768.jpg 768w, https://web.uri.edu/cs/wp-content/uploads/sites/1531/Ashley-Buchanan-364x364.jpg 364w, https://web.uri.edu/cs/wp-content/uploads/sites/1531/Ashley-Buchanan-500x500.jpg 500w" width="966"/>
</figure>
<h3 class="p-name">Ashley Buchanan</h3>

</header>

Limited Joint Appointment
Biostatistics
401.874.4739 – buchanan@uri.edu

<header>

<figure>

</figure>
<h3 class="p-name">Nina Kajiji</h3>

</header>

Adjunct Associate Professor
Computer Science
nina@uri.edu

<header>

<figure>

</figure>
<h3 class="p-name">Rachel Schwartz</h3>

</header>

Assistant Professor – Limited Joint Appointment
Biological Sciences
401.874.5404 – rsschwartz@uri.edu

<header>

<figure>

</figure>
<h3 class="p-name">Ying Zhang</h3>

</header>

Assistant Professor – Limited Joint Appointment
Computer Science
401.874.4915 – yingzhang@uri.edu

<!-- .entry-content -->
</article><!-- #post-## -->
</main><!-- #main -->
<!-- #content -->

<a href="https://www.uri.edu/connect/?utm_campaign=request-info&utm_source=action-bar&utm_medium=web" id="action-connect" role="menuitem"><span role="presentation" title="Learn

```
id="action-tour" role="menuitem"><span role="presentation"></span>Tour</a><a
href="https://www.uri.edu/give" id="action-give" role="menuitem"><span role="presentation">
</span>Give</a> </div>
</div><!-- #actionbar-wrapper -->
<footer id="globalfooter">
<div id="basement">
<div id="storagebins">
<div id="sb-university">
<input checked="" id="sb-university-toggle" name="storagebin" role="presentation" type="radio"
value="university"/>
<label aria-label="Open the University footer menu when browsing on mobile." for="sb-university-
toggle"><span>University</span></label>
<ul aria-label="The University footer menu." role="menu">
<li><a href="https://www.uri.edu/about/leadership/" role="menuitem">Leadership</a></li>
<li><a href="https://web.uri.edu/diversity/" role="menuitem">Diversity and Inclusion</a></li>
<li><a href="https://web.uri.edu/global/" role="menuitem">Global</a></li>
<li><a href="https://www.uri.edu/about/campuses/" role="menuitem">Campuses</a></li>
<li><a href="https://www.uri.edu/safety/" role="menuitem">Safety</a></li>
</ul>
</div>
<div id="sb-campus-life">
<input id="sb-campus-life-toggle" name="storagebin" role="presentation" type="radio" value="campus-
life"/>
<label aria-label="Open the Campus Life footer menu when browsing on mobile." for="sb-campus-life-
toggle"><span>Campus Life</span></label>
<ul aria-label="The Campus Life footer menu." role="menu">
<li><a href="https://web.uri.edu/housing/" role="menuitem">Housing</a></li>
<li><a href="https://web.uri.edu/dining/" role="menuitem">Dining</a></li>
<li><a href="https://www.uri.edu/athletics/" role="menuitem">Athletics and Recreation</a></li>
<li><a href="https://www.uri.edu/campus-life/health-and-wellness/" role="menuitem">Health and
Wellness</a></li>
<li><a href="https://events.uri.edu" role="menuitem">Events</a></li>
</ul>
</div>
<div id="sb-academics">
<input id="sb-academics-toggle" name="storagebin" role="presentation" type="radio" value="academics"/>
<label aria-label="Open the Academics footer menu when browsing on mobile." for="sb-academics-toggle">
<span>Academics</span></label>
<ul aria-label="The Academics footer menu." role="menu">
<li><a href="https://www.uri.edu/academics/" role="menuitem">Undergraduate</a></li>
<li><a href="https://web.uri.edu/graduate-school/" role="menuitem">Graduate</a></li>
<li><a href="https://web.uri.edu/advising/" role="menuitem">Advising</a></li>
<li><a href="https://web.uri.edu/library/" role="menuitem">Libraries</a></li>
<li><a href="https://web.uri.edu/career/students/" role="menuitem">Internships</a></li>
</ul>
</div>
</div>
<div id="gimmicks">
<!-- Tides Widget -->
<div class="uri-tides-widget darkmode" data-height="22"><span class="status"></span></div>
<hr/>
<!-- Social Media Component -->
<aside class="cl-wrapper cl-social-wrapper"><ul class="cl-social light"><li><a class="cl-social-
facebook" href="https://www.facebook.com/universityofri" title="Facebook">Facebook</a></li><li><a
class="cl-social-instagram" href="https://www.instagram.com/universityofri/"
title="Instagram">Instagram</a></li><li><a class="cl-social-twitter"
href="https://twitter.com/universityofri" title="Twitter">Twitter</a></li><li><a class="cl-social-
youtube" href="https://www.youtube.com/user/UniversityOfRI" title="YouTube">YouTube</a></li></ul>
</aside> </div>
</div>
<div id="tagline"></div>
<div id="legal">
<p>Copyright © <a class="subtle" href="http://www.uri.edu/">University of Rhode Island</a> |
University of Rhode Island, Kingston, RI 02881, USA | 1.401.874.1000</p>
<p>URI is an equal opportunity employer committed to the principles of affirmative action.  <a
class="jobs" href="https://jobs.uri.edu/">Work at URI</a></p>
</div>
</footer><!-- #globalfooter -->
</div>
```

`cs_people.h3`

```
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/marco-alvarez/">Marco Alvarez</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

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

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

▶ Show code cell output

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

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

```
24
```

Important

answer to questions about searching from the docs

```
type(people_items)
```

```
bs4.element.ResultSet
```

We can also look at only the first instance

```
people_items[0]
```

```
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>
<a href="https://web.uri.edu/cs/meet/marco-alvarez/"><img alt="" class="u-photo wp-post-image"
height="120" loading="lazy" src="https://web.uri.edu/cs/wp-content/uploads/sites/1531/marco-
alvarez.png" width="120"/></a>
</figure>
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/marco-alvarez/">Marco Alvarez</a></h3>
</div>
</header>
<div class="inside">
<p class="people-title p-job-title">Associate Professor | Director of Graduate Studies</p>
<p class="people-department">Computer Science</p>
<p class="people-misc"><span class="p-tel">401.874.5009</span> – <a class="u-email"
href="mailto:malvarez@uri.edu">malvarez@uri.edu</a></p>
<div style="clear:both;"></div>
</div>
</div>
```

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').a.string
```

```
'Marco Alvarez'
```

Then we can use a list comprehension to build alist of them.

```
names = [name.a.string for name in cs_people.find_all('h3','p-name')]
names
```

```
 'Sarah Brown',
 'Michael Conti',
 'Noah Daniels',
 'Lisa DiPippo',
 'Victor Fay-Wolfe',
 'Lutz Hamel',
 'Abdeltawab Hendawi',
 'Jean-Yves Hervé',
 'Natallia Katenka',
 'Soheyb Kouider',
 'Edmund Lamagna',
 'Indrani Mandal',
 'Gavino Puggioni',
 'Jonathan Schrader',
 'Krishna Venkatasubramanian',
 'Jing Wu',
 'Yichi Zhang',
 'Guangyu Zhu',
 'Ashley Buchanan',
 'Nina Kajiji',
 'Rachel Schwartz',
 'Ying Zhang']
```

```
people_items[0]
```

```
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>
<a href="https://web.uri.edu/cs/meet/marco-alvarez/"><img alt="" class="u-photo wp-post-image"
height="120" loading="lazy" src="https://web.uri.edu/cs/wp-content/uploads/sites/1531/marco-
alvarez.png" width="120"/></a>
</figure>
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/marco-alvarez/">Marco Alvarez</a></h3>
</div>
</header>
<div class="inside">
<p class="people-title p-job-title">Associate Professor | Director of Graduate Studies</p>
<p class="people-department">Computer Science</p>
<p class="people-misc"><span class="p-tel">401.874.5009</span> – <a class="u-email"
href="mailto:malvarez@uri.edu">malvarez@uri.edu</a></p>
<div style="clear:both;"></div>
</div>
</div>
```

```
people_items[0].find('p','people-title').string
```

```
'Associate Professor | Director of Graduate Studies'
```

How to pull out the titles for each person (eg Assitatn Teaching Professor, Associate Professor)

```
titles = [t.string for t in cs_people.find_all("p","people-title")]
```

```
titles
```

```
 'Assistant Professor',
 'Assistant Teaching Professor',
 'Assistant Professor',
 'Professor | Chair',
 'Professor',
 'Associate Professor ',
 'Assistant Professor',
 'Associate Professor',
 'Associate Professor | Director of Undergraduate Studies',
 'Associate Teaching Professor',
 'Professor',
 'Assistant Teaching Professor',
 'Associate Professor |  Statistics Section Head | Director of Graduate Studies',
 'Assistant Teaching Professor',
 'Assistant Professor',
 'Associate Professor',
 'Assistant Professor ',
 'Assistant Professor',
 'Limited Joint Appointment',
 'Adjunct Associate Professor',
 'Assistant Professor – Limited Joint Appointment',
 'Assistant Professor – Limited Joint Appointment']
```

on one item, the `p` tag seems to work, but that is because the tag gives only the first instance,

```
people_items[0].find('p')
```

```
<p class="people-title p-job-title">Associate Professor | Director of Graduate Studies</p>
```

but we see if we ust this for all, it is way more informaiton than we were looking for.

```
[t.string for t in cs_people.find_all("p")]
```

```
 'Computer Science',
 None,
 'Assistant Teaching Professor',
 'Computer Science',
 'sarmenti@uri.edu ',
 'Assistant Professor',
 'Computer Science',
 'brownsarahm@uri.edu',
 'Assistant Teaching Professor',
 'Computer Science',
 'michaelconti@uri.edu ',
 'Assistant Professor',
 'Computer Science',
 'noah_daniels@uri.edu',
 'Professor | Chair',
 'Computer Science',
 'ldipippo@uri.edu',
 'Professor',
 'Computer Science',
 'vfaywolfe@uri.edu',
 'Associate Professor ',
 'Computer Science',
 'lutzhamel@uri.edu',
 'Assistant Professor',
 'Data Science | Computer Science',
 None,
 'Associate Professor',
 'Computer Science',
 'jyh@cs.uri.edu',
 'Associate Professor | Director of Undergraduate Studies',
 'Statistics',
 'nkatenka@uri.edu',
 'Associate Teaching Professor',
 'Statistics',
 None,
 'Professor',
 'Computer Science',
 'eal@cs.uri.edu',
 'Assistant Teaching Professor',
 'Computer Science',
 'indrani_mandal@uri.edu ',
 'Associate Professor |  Statistics Section Head | Director of Graduate Studies',
 'Statistics',
 None,
 'Assistant Teaching Professor',
 'Computer Science',
 'jonathan.schrader@uri.edu',
 'Assistant Professor',
 'Computer Science',
 'krish@uri.edu',
 'Associate Professor',
 'Statistics',
 None,
 'Assistant Professor ',
 'Statistics',
 'yichizhang@uri.edu',
 'Assistant Professor',
 'Statistics',
 'guangyuzhu@uri.edu',
 None,
 'Limited Joint Appointment',
 'Biostatistics',
 None,
 'Adjunct Associate Professor',
 'Computer Science',
 'nina@uri.edu',
 'Assistant Professor – Limited Joint Appointment',
 'Biological Sciences',
 None,
 'Assistant Professor – Limited Joint Appointment',
 'Computer Science',
 None,
 '\n',
 None,
 None]
```

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')]
```

```
css_df.head()
```

	name	title	e-mails	discipline
0	Marco Alvarez	Associate Professor | Director of Graduate Stu...	malvarez@uri.edu	Computer Science
1	Samantha Armenti	Assistant Teaching Professor	sarmenti@uri.edu	Computer Science
2	Sarah Brown	Assistant Professor	brownsarahm@uri.edu	Computer Science
3	Michael Conti	Assistant Teaching Professor	michaelconti@uri.edu	Computer Science
4	Noah Daniels	Assistant Professor	noah_daniels@uri.edu	Computer Science

10.4. 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/marco-alvarez/">Marco Alvarez</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/marco-alvarez/'}
```

It's a dictionary and the attribute we want is the key we want. Nowe, we do the same thing we did above, request, pull the content from the response and then use the parser.

```
alvarez_url = people_items[0].find('h3','p-name').a.attrs['href']
alvarez_html = requests.get(alvarez_url).content
alvarez_info = BeautifulSoup(alvarez_html,'html.parser')
```

then we find the tag and class we need from inspecting and pull that.

```
alvarez_info.find_all('li','people-location')
```

```
[<li class="people-location"><strong>Office Location:</strong> Tyler 255</li>]
```

it's an interable, so we pull the item out

```
alvarez_info.find_all('li','people-location')[0]
```

```
<li class="people-location"><strong>Office Location:</strong> Tyler 255</li>
```

Then we get the content.

```
alvarez_info.find_all('li','people-location')[0].string
```

```
alvarez_info.find_all('li','people-location')[0].__dict__
```


```
{'parser_class': bs4.BeautifulSoup,
 'name': 'li',
 'namespace': None,
 '_namespaces': {},
 'prefix': None,
 'sourceline': 371,
 'sourcepos': 329,
 'known_xml': False,
 'attrs': {'class': ['people-location']},
 'contents': [<strong>Office Location:</strong>, ' Tyler 255'],
 'parent': <ul class="people-list">
 <li class="people-title">Associate Professor | Director of Graduate Studies</li> <li class="people-
department">Computer Science</li> <li class="people-phone"><strong>Phone:</strong> 401.874.5009</li>
<li class="people-email"><strong>Email:</strong> <a
href="mailto:malvarez@uri.edu">malvarez@uri.edu</a></li> <li class="people-location"><strong>Office
Location:</strong> Tyler 255</li> <li class="people-url"><strong>Website:</strong> <a
href="http://homepage.cs.uri.edu/~malvarez/">http://homepage.cs.uri.edu/~malvarez/</a> </li></ul>,
 'previous_element': ' ',
 'next_element': <strong>Office Location:</strong>,
 'next_sibling': ' ',
 '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)}
```


it's the second elment of content

```
alvarez_info.find_all('li','people-location')[0].contents[1]
```

```
' Tyler 255'
```

Now tht we know how to do it, we can put it in a loop.

```
offices = []
for name_link in cs_people.find_all('h3','p-name'):
    url = name_link.a.attrs['href']
    person_html = requests.get(url).content
    person_info = BeautifulSoup(person_html,'html.parser')
    try:
        offices.append(person_info.find_all('li','people-location')[0].contents[1])
    except:
        offices.append(pd.NA)


css_df['office'] = offices
```

We got an error at first, so we added the `try` and `except` to handle when there is no office location.

```
css_df.head()
```

0	Marco Alvarez	Associate Professor | Director of Graduate Stu...	malvarez@uri.edu	Computer Science	Tyler 255
1	Samantha Armenti	Assistant Teaching Professor	sarmenti@uri.edu	Computer Science	Tyler 129
2	Sarah Brown	Assistant Professor	brownsarahm@uri.edu	Computer Science	Tyler 134
3	Michael Conti	Assistant Teaching Professor	michaelconti@uri.edu	Computer Science	Tyler 137
4	Noah Daniels	Assistant Professor	noah_daniels@uri.edu	Computer Science	Tyler 250

```
css_df.to_csv('css_faculty.csv')
```

10.5. Questions after class

10.5.1. what does .a do?

it gives the first instance of the `<a>` tag

10.5.2. 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.

10.5.3. In theory, you could parse images and potentially their metadata with this method?

This method could be a way to download images and the text that is around them, yes. This is how a lot of image datasets are built for machine learning.

10.5.4. Is API the website's way of specify what information it will allow for you have?

What we did today did not use any API. An API call would use the request library, and similar patterns to what we did, espeically the end of class. However and API call would typically respond with json, not html.

10.5.5. In the web-scraping of the offices, there were two strings, 'CBLS 377', and 'CBLS Building 487' how would we use pandas to normalize things like this?"

We could use the `replace` method that we used last week.

11. Evaluating ML Algorithms

This week we are going to start learning about machine learning.

We are going to do this by looking at how to tell if machine learning has worked.

This is because:

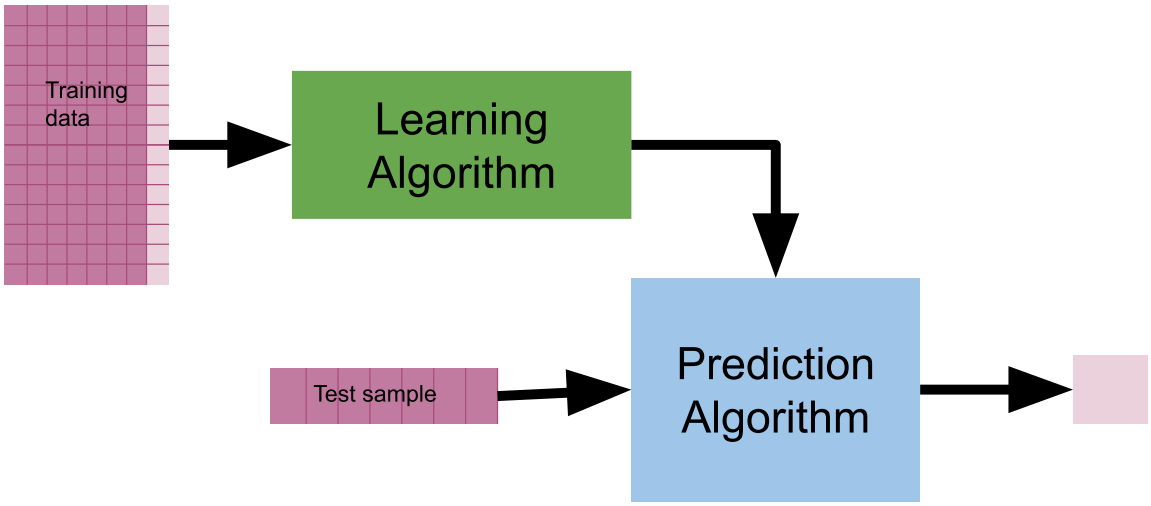
- you have to check if one worked after you build one
- if you do not check carefully, it might only sometimes work
- gives you a chance to learn *only* evaluation instead of evaluation + an ML task

First, what is an Algorithm?

> An algorithm is a set of ordered steps to complete a task.

Note that when people outside of CS talk about algorithms that impact people's lives these are often *not written directly by people* anymore. THey are often the result of machine learning.

In machine learning, people write an algorithm for how to write an algorithm based on data. This often comes in the form of a statitistical model of some sort.



When we *do* machine learning, this can also be called:

- data mining
- pattern recognition
- modeling

because we are looking for patterns in the data and typically then planning to use those patterns to make predictions or automate a task.

Each of these terms does have slightly different meanings and usage, but sometimes they're used close to exchangeably.

11.2. How can we tell if ML is working?

We measure the performance of the prediction algorithm, to determine if the learning algorithm worked.

11.3. Replicating the COMPAS Audit

We are going to replicate the audit from ProPublica Machine Bias

Propublica started the COMPAS Debate with the article Machine Bias. With their article, they also released details of their methodology and their data and code. This presents a real data set that can be used for research on how data is used in a criminal justice setting without researchers having to perform their own requests for information, so it has been used and reused a lot of times.

11.3.2. Propublica COMPAS Data

The dataset consists of COMPAS scores assigned to defendants over two years 2013-2014 in Broward County, Florida, it was released by Propublica in a GitHub Repository. These scores are determined by a proprietary algorithm designed to evaluate a persons recidivism risk - the likelihood that they will reoffend. Risk scoring algorithms are widely used by judges to inform their sentencing and bail decisions in the criminal justice system in the United States.

The journalists collected, for each person arreste din 2013 and 2014:

- basic demographics
- details about what they were charged with and priors
- the COMPAS score assigned to them
- if they had actually been re-arrested within 2 years of their arrest

This means that we have what the COMPAS algorithm predicted (in the form of a score from 1-10) and what actually happened (re-arrested or not). We can then measure how well the algorithm worked, in practice, in the real world.

```
import pandas as pd
from sklearn import metrics
import seaborn as sns
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[1], line 2
      1 import pandas as pd
----> 2 from sklearn import metrics
      3 import seaborn as sns

ModuleNotFoundError: No module named 'sklearn'
```

We're going to work with a cleaned copy of the data released by Propublica that also has a minimal subset of features.

- `age`: defendant's age
- `c_charge_degree`: degree charged (Misdemeanor of Felony)
- `race`: defendant's race
- `age_cat`: defendant's age quantized in "less than 25", "25-45", or "over 45"
- `score_text`: COMPAS score: 'low'(1 to 5), 'medium' (5 to 7), and 'high' (8 to 10).
- `sex`: defendant's gender
- `priors_count`: number of prior charges
- `days_b_screening_arrest`: number of days between charge date and arrest where defendant was screened for compas score
- `decile_score`: COMPAS score from 1 to 10 (low risk to high risk)
- `is_recid`: if the defendant recidivized
- `two_year_recid`: if the defendant within two years
- `c_jail_in`: date defendant was imprisoned
- `c_jail_out`: date defendant was released from jail
- `length_of_stay`: length of jail stay

```
compas_clean_url = 'https://raw.githubusercontent.com/ml4sts/outreach-compas/main/data/compas_c.csv'
compas_df = pd.read_csv(compas_clean_url)
compas_df.head()
```

0	3	34	F	African-American	25 - 45	Low	Male	0	-1.(
1	4	24	F	African-American	Less than 25	Low	Male	4	-1.(
2	8	41	F	Caucasian	25 - 45	Medium	Male	14	-1.(
3	10	39	M	Caucasian	25 - 45	Low	Female	0	-1.(
4	14	27	F	Caucasian	25 - 45	Low	Male	0	-1.(

11.4. One-hot Encoding

We will audit first to see how good the algorithm is by treating the predictions as either high or not high. One way we can get to that point is to transform the `score_text` column from one column with three values, to 3 binary columns.

```
pd.get_dummies(compas_df['score_text'])
```

	High	Low	Medium
0	0	1	0
1	0	1	0
2	0	0	1
3	0	1	0
4	0	1	0
...	...	...	...
5273	0	1	0
5274	1	0	0
5275	0	0	1
5276	0	1	0
5277	0	1	0

5278 rows × 3 columns

```
compas_onehot = pd.concat([compas_df,pd.get_dummies(compas_df['score_text'])],axis=1)
```

We could have done the above line in one neater step, but in class I for this was an option.

```
compas_df_onehot = pd.get_dummies(compas_df,columns=['score_text'])
```

Next lets look at the thresholds that were used so that we know what the mean

```
compas_onehot.groupby('score_text')['decile_score'].agg(['min','max'])
```

score_text		
High	8	10
Low	1	4
Medium	5	7

We will also audit with respect to second threshold.

```
compas_onehot['MedHigh'] = compas_onehot['High'] + compas_onehot['Medium']
```

11.5. Sklearn Performance metrics

The first thing we usually check is the accuracy: the percentage of all samples that are correct.

```
metrics.accuracy_score(compas_onehot['two_year_recid'],compas_onehot['High'])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[8], line 1
----> 1 metrics.accuracy_score(compas_onehot['two_year_recid'],compas_onehot['High'])

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

However this does not tell us anything about what *types* of mistakes the algorithm made. The type of mistake often matters in terms of how we trust or deploy an algorithm. We use a confusion matrix to describe the performance in more detail.

> **Note**
>
> the wikipedia page for confusion matrix is a really good reference

A confusion matrix counts the number of samples of each *true* category that wre predicted to be in each category. In this case we have a binary prediction problem: people either are re-arrested (truth) or not and were given a high score or not(prediction). In binary problems we adopt a common language of labeling one outcome/predicted value positive and the other negative. We do this not based on the social value of the outcome, but on the numerical encoding.

In this data, being re-arrested is indicated by a 1 in the `two_year_recid` column, so this is the *positive class* and not being re-arrested is 0, so the *negative class*. Similarly a high score is 1, so that's the *positive prediction* and not high is 0, so that is the a *negative prediction*.

```
metrics.accuracy_score(compas_onehot['two_year_recid'],compas_onehot['MedHigh'])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[9], line 1
----> 1 metrics.accuracy_score(compas_onehot['two_year_recid'],compas_onehot['MedHigh'])

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

docs

```
metrics.confusion_matrix(compas_onehot['two_year_recid'],compas_onehot['MedHigh'])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[10], line 1
----> 1 metrics.confusion_matrix(compas_onehot['two_year_recid'],compas_onehot['MedHigh'])

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

> these terms can be used in any sort of detection problem, whether machine learning is used or not

`sklearn.metrics` provides a `[confusion matrix](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html)` function that we can use.

Since this is binary problem we have 4 possible outcomes:

- true negatives($C_{0,0}$): did not get a high score and were not re-arrested
- false negatives($C_{1,0}$):: did not get a high score and were re-arrested
- false positives($C_{0,1}$):: got a high score and were not re-arrested
- true positives($C_{1,1}$):: got a high score and were re-arrested

With these we can revisit accuracy:

$$A = \frac{C_{0,0} + C_{1,1}}{C_{0,0} + C_{1,0} + C_{0,1} + C_{1,1}}$$

and we can define new scores. Two common ones in CS are recall and precision.

Recall is:

$$R = \frac{C_{1,1}}{C_{1,0} + C_{1,1}}$$

```
metrics.recall_score(compas_df['two_year_recid'],compas_df['score_text_High'])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[11], line 1
----> 1 metrics.recall_score(compas_df['two_year_recid'],compas_df['score_text_High'])

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

That is, among the truly positive class how many were correctly predicted? In COMPAS, it's the percentage of the re-arrested people who got a high score.

Precision is $$P = \frac{C_{1,1}}{C_{0,1}+C_{1,1}}$$

```
metrics.recall_score(compas_onehot['two_year_recid'],compas_onehot['MedHigh'])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[12], line 1
----> 1 metrics.recall_score(compas_onehot['two_year_recid'],compas_onehot['MedHigh'])

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

```
metrics.precision_score(compas_onehot['two_year_recid'],compas_onehot['MedHigh'])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[13], line 1
----> 1 metrics.precision_score(compas_onehot['two_year_recid'],compas_onehot['MedHigh'])

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

To groupby and then do the score, we can use a lambda again, with apply

```
acc_fx  = lambda d: metrics.accuracy_score(d['two_year_recid'],d['MedHigh'])
compas_onehot.groupby('race').apply(acc_fx)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[14], line 2
      1 acc_fx  = lambda d: metrics.accuracy_score(d['two_year_recid'],d['MedHigh'])
----> 2 compas_onehot.groupby('race').apply(acc_fx)

File /opt/hostedtoolcache/Python/3.8.16/x64/lib/python3.8/site-
packages/pandas/core/groupby/groupby.py:1567, in GroupBy.apply(self, func, *args, **kwargs)
   1559     new_msg = (
   1560         f"The operation {orig_func} failed on a column. If any error is "
   1561         f"raised, this will raise an exception in a future version "
   1562         f"of pandas. Drop these columns to avoid this warning."
   1563     )
   1564     with rewrite_warning(
   1565         old_msg, FutureWarning, new_msg
   1566     ) if is_np_func else nullcontext():
-> 1567         result = self._python_apply_general(f, self._selected_obj)
   1568 except TypeError:
   1569     # gh-20949
   1570     # try again, with .apply acting as a filtering
   (...)
   1574     # fails on *some* columns, e.g. a numeric operation
   1575     # on a string grouper column
   1577     with self._group_selection_context():
   1578         # GH#50538

File /opt/hostedtoolcache/Python/3.8.16/x64/lib/python3.8/site-
packages/pandas/core/groupby/groupby.py:1629, in GroupBy._python_apply_general(self, f, data,
not_indexed_same, is_transform, is_agg)
   1592 @final
   1593 def _python_apply_general(
   1594     self,
   (...)
   1599     is_agg: bool = False,
   1600 ) -> NDFrameT:
   1601     """
   1602     Apply function f in python space
   1603
   (...)
   1627         data after applying f
   1628     """
-> 1629     values, mutated = self.grouper.apply(f, data, self.axis)
   1630     if not_indexed_same is None:
   1631         not_indexed_same = mutated or self.mutated

File /opt/hostedtoolcache/Python/3.8.16/x64/lib/python3.8/site-
packages/pandas/core/groupby/ops.py:839, in BaseGrouper.apply(self, f, data, axis)
    837 # group might be modified
    838 group_axes = group.axes
--> 839 res = f(group)
    840 if not mutated and not _is_indexed_like(res, group_axes, axis):
    841     mutated = True

Cell In[14], line 1, in <lambda>(d)
----> 1 acc_fx  = lambda d: metrics.accuracy_score(d['two_year_recid'],d['MedHigh'])
      2 compas_onehot.groupby('race').apply(acc_fx)

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

That lambda + apply is equivalent to:

```
race_acc = []
for race, rdf in compas_race:
    acc = skmetrics.accuracy_score(rdf['two_year_recid'],
             rdf['score_text_MedHigh'])
    race_acc.append([race,acc])

pd.DataFrame(race_acc, columns =['race','accuracy'])
```

```
Cell In[15], line 2
      1 race_acc = []
----> 2 for race, rdf in compas_race:
      3     acc = skmetrics.accuracy_score(rdf['two_year_recid'],
      4               rdf['score_text_MedHigh'])
      5     race_acc.append([race,acc])

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

```
recall_fx  = lambda d: metrics.recall_score(d['two_year_recid'],d['MedHigh'])
compas_onehot.groupby('race').apply(recall_fx)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[16], line 2
      1 recall_fx  = lambda d: metrics.recall_score(d['two_year_recid'],d['MedHigh'])
----> 2 compas_onehot.groupby('race').apply(recall_fx)

File /opt/hostedtoolcache/Python/3.8.16/x64/lib/python3.8/site-
packages/pandas/core/groupby/groupby.py:1567, in GroupBy.apply(self, func, *args, **kwargs)
   1559     new_msg = (
   1560         f"The operation {orig_func} failed on a column. If any error is "
   1561         f"raised, this will raise an exception in a future version "
   1562         f"of pandas. Drop these columns to avoid this warning."
   1563     )
   1564     with rewrite_warning(
   1565         old_msg, FutureWarning, new_msg
   1566     ) if is_np_func else nullcontext():
-> 1567         result = self._python_apply_general(f, self._selected_obj)
   1568 except TypeError:
   1569     # gh-20949
   1570     # try again, with .apply acting as a filtering
   (...)
   1574     # fails on *some* columns, e.g. a numeric operation
   1575     # on a string grouper column
   1577     with self._group_selection_context():
   1578         # GH#50538

File /opt/hostedtoolcache/Python/3.8.16/x64/lib/python3.8/site-
packages/pandas/core/groupby/groupby.py:1629, in GroupBy._python_apply_general(self, f, data,
not_indexed_same, is_transform, is_agg)
   1592 @final
   1593 def _python_apply_general(
   1594     self,
   (...)
   1599     is_agg: bool = False,
   1600 ) -> NDFrameT:
   1601     """
   1602     Apply function f in python space
   1603
   (...)
   1627         data after applying f
   1628     """
-> 1629     values, mutated = self.grouper.apply(f, data, self.axis)
   1630     if not_indexed_same is None:
   1631         not_indexed_same = mutated or self.mutated

File /opt/hostedtoolcache/Python/3.8.16/x64/lib/python3.8/site-
packages/pandas/core/groupby/ops.py:839, in BaseGrouper.apply(self, f, data, axis)
    837 # group might be modified
    838 group_axes = group.axes
--> 839 res = f(group)
    840 if not mutated and not _is_indexed_like(res, group_axes, axis):
    841     mutated = True

Cell In[16], line 1, in <lambda>(d)
----> 1 recall_fx  = lambda d: metrics.recall_score(d['two_year_recid'],d['MedHigh'])
      2 compas_onehot.groupby('race').apply(recall_fx)

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

```
precision_fx  = lambda d: metrics.precision_score(d['two_year_recid'],d['MedHigh'])
compas_onehot.groupby('race').apply(precision_fx)
```

```
NameError                                 Traceback (most recent call last)
Cell In[17], line 2
      1 precision_fx  = lambda d: metrics.precision_score(d['two_year_recid'],d['MedHigh'])
----> 2 compas_onehot.groupby('race').apply(precision_fx)

File /opt/hostedtoolcache/Python/3.8.16/x64/lib/python3.8/site-
packages/pandas/core/groupby/groupby.py:1567, in GroupBy.apply(self, func, *args, **kwargs)
   1559     new_msg = (
   1560         f"The operation {orig_func} failed on a column. If any error is "
   1561         f"raised, this will raise an exception in a future version "
   1562         f"of pandas. Drop these columns to avoid this warning."
   1563     )
   1564     with rewrite_warning(
   1565         old_msg, FutureWarning, new_msg
   1566     ) if is_np_func else nullcontext():
-> 1567         result = self._python_apply_general(f, self._selected_obj)
   1568 except TypeError:
   1569     # gh-20949
   1570     # try again, with .apply acting as a filtering
   (...)
   1574     # fails on *some* columns, e.g. a numeric operation
   1575     # on a string grouper column
   1577     with self._group_selection_context():
   1578         # GH#50538

File /opt/hostedtoolcache/Python/3.8.16/x64/lib/python3.8/site-
packages/pandas/core/groupby/groupby.py:1629, in GroupBy._python_apply_general(self, f, data,
not_indexed_same, is_transform, is_agg)
   1592 @final
   1593 def _python_apply_general(
   1594     self,
   (...)
   1599     is_agg: bool = False,
   1600 ) -> NDFrameT:
   1601     """
   1602     Apply function f in python space
   1603
   (...)
   1627         data after applying f
   1628     """
-> 1629     values, mutated = self.grouper.apply(f, data, self.axis)
   1630     if not_indexed_same is None:
   1631         not_indexed_same = mutated or self.mutated

File /opt/hostedtoolcache/Python/3.8.16/x64/lib/python3.8/site-
packages/pandas/core/groupby/ops.py:839, in BaseGrouper.apply(self, f, data, axis)
    837 # group might be modified
    838 group_axes = group.axes
--> 839 res = f(group)
    840 if not mutated and not _is_indexed_like(res, group_axes, axis):
    841     mutated = True

Cell In[17], line 1, in <lambda>(d)
----> 1 precision_fx  = lambda d: metrics.precision_score(d['two_year_recid'],d['MedHigh'])
      2 compas_onehot.groupby('race').apply(precision_fx)

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

The recall tells us that the model has very different impact on people. On the other hand the precision tells us the scores mean about the same thing for Black and White people.

Researchers established that these are mutually exclusive, provably. We cannot have both, so it is very important to think about what the performance metrics mean and how your algorithm will be used in order to choose how to prepare a model. We will train models starting next week, but knowing these goals in advance is essential.

Importantly, this is not a statistical, computational choice that data can answer for us. This is about *human* values (and to some extent the law; certain domains have legal protections that require a specific condition).

The Fair Machine Learning book's classificaiton Chapter has a section on relationships between criteria with the proofs.

Important

We used ProPublica's COMPAS dataset to replicate (parts of, with different tools) their analysis. That is, they collected the dataset in order to audit the COMPAS algorithm and we used it for the same purpose (and to learn model evaluation). This dataset is not designed for *training* models, even though it has been used as such many times. This is not the best way to use this dataset and for future assignments I do not recommend using this dataset.

install aif360

> **Note**
>
> If you are interested in fairness in ML, that is what my research is. Reach out to me if you want to know more!

11.8. Portfolio

Audience is not *me*, but a generally knowledgable person. For example:

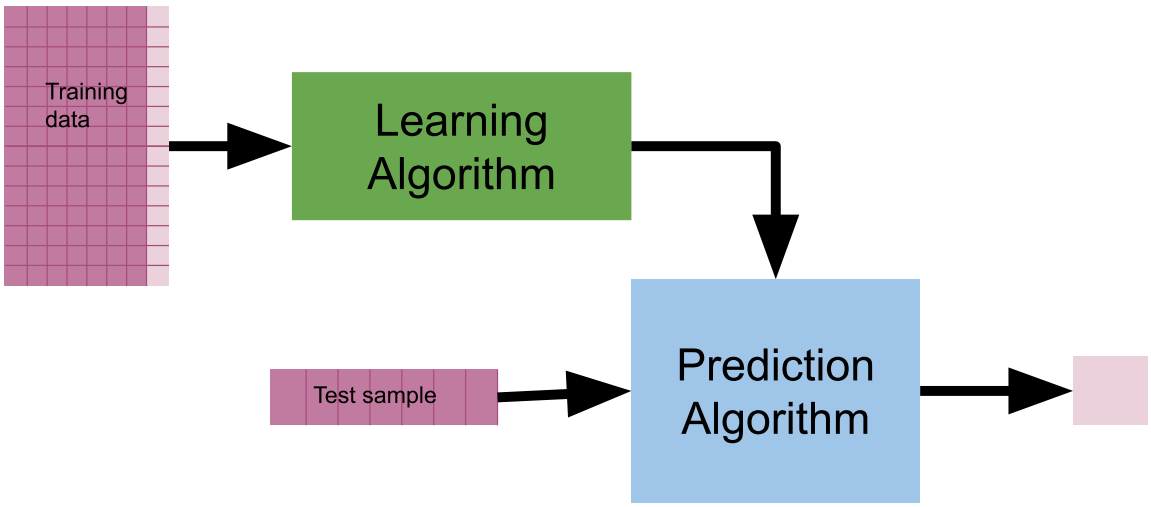
- a student deciding if they want to take this course or not. They know how to code, but not datascience.
- a person familiar with the domain your data is from (eg a sports fan if sports data)
- a future employer who wants to know about your skills

any of these people know big ideas, but not exactly what happened in class. You can specify which audience you're targetting in the introduction (which is the one piece that I'm the audience for)

Goal is to show what you understand and are able to do not only what you *can* do, because you can do a lot of simple things by finding answers online. we want you to understand enough that when you start seeing new, real problems, you're able to do these things on your own

- level 1: you can follow a conversation
- level 2: you can do it if someone gives you a rough plan
- level 3: you can do it, given only an end goal Think of this more like a report with code as the figures than a coding assignment. To see what you've learned we should be able to read through on piece of text, not compare two files so it could be like:

12. What is ML?



```
from sklearn import metrics as skmetrics
from aif360 import metrics as fairmetrics
from aif360.datasets import BinaryLabelDataset
import seaborn as sns

compas_clean_url = 'https://raw.githubusercontent.com/ml4sts/outreach-compas/main/data/compas_c.csv'
compas_df = pd.read_csv(compas_clean_url,index_col = 'id')

compas_df = pd.get_dummies(compas_df,columns=['score_text'],)
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[1], line 2
      1 import pandas as pd
----> 2 from sklearn import metrics as skmetrics
      3 from aif360 import metrics as fairmetrics
      4 from aif360.datasets import BinaryLabelDataset

ModuleNotFoundError: No module named 'sklearn'
```

We may get a warning which is **okay**. If you run the cell again it will go away.

12.1. The COMPAS data

We are going to continue with the ProPublica COMPAS audit data. Remember it contains:

- `age`: defendant's age
- `c_charge_degree`: degree charged (Misdemeanor of Felony)
- `race`: defendant's race
- `age_cat`: defendant's age quantized in "less than 25", "25-45", or "over 45"
- `score_text`: COMPAS score: 'low'(1 to 5), 'medium' (5 to 7), and 'high' (8 to 10).
- `sex`: defendant's gender
- `priors_count`: number of prior charges
- `days_b_screening_arrest`: number of days between charge date and arrest where defendant was screened for compas score
- `decile_score`: COMPAS score from 1 to 10 (low risk to high risk)
- `is_recid`: if the defendant recidivized
- `two_year_recid`: if the defendant within two years
- `c_jail_in`: date defendant was imprisoned
- `c_jail_out`: date defendant was released from jail
- `length_of_stay`: length of jail stay

First, we will look at it

```
compas_df.head()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[2], line 1
----> 1 compas_df.head()

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

Notice the last three columns. When we use `pd.getdummies` with its `columns` parameter, then we can append the columns all at once and they get the original column name prepended to the value in the new column name.

We use the `two_year_recid` as the basis of our audit because it is the real outcome that the designers of COMPAS were hoping to predict. Sicne the COMPAS score is on a scale of 1-10, we transform to a binary variable by thresholding it (eg all above t are 1, below are 0). We use the `score_text` instead of `decile_score` in our thresholding so that we use a recommended threshold.

let's do it by inverting here

```
int_not = lambda a:int(not(a))
compas_df['score_text_MedHigh'] = compas_df['score_text_Low'].apply(int_not)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[3], line 2
      1 int_not = lambda a:int(not(a))
----> 2 compas_df['score_text_MedHigh'] = compas_df['score_text_Low'].apply(int_not)

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

Let's review computing the accruacy with sklearn:

```
skmetrics.accuracy_score(compas_df['two_year_recid'],
                         compas_df['score_text_High'])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[4], line 1
----> 1 skmetrics.accuracy_score(compas_df['two_year_recid'],
      2                          compas_df['score_text_High'])

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

```
skmetrics.accuracy_score(compas_df['two_year_recid'],
                         compas_df['score_text_MedHigh'])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[5], line 1
----> 1 skmetrics.accuracy_score(compas_df['two_year_recid'],
      2                          compas_df['score_text_MedHigh'])

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

12.2. What about breaking it down by race?

Recall, we used groupby to get the per race score by creating a `lambda` function that we could apply to the groupby object.

```
compas_race = compas_df.groupby('race')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[6], line 1
----> 1 compas_race = compas_df.groupby('race')

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

We can apply our method to each part of the groupby object with `apply`

```
acc_fx = lambda d: skmetrics.accuracy_score(d['two_year_recid'],
                         d['score_text_MedHigh'])

compas_race.apply(acc_fx,).reset_index().rename(columns={0:'accuracy'})
```

```
Cell In[7], line 4
      1 acc_fx = lambda d: skmetrics.accuracy_score(d['two_year_recid'],
      2                         d['score_text_MedHigh'])
----> 4 compas_race.apply(acc_fx,).reset_index().rename(columns={0:'accuracy'})

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

12.3. ML Notation

We use standard notation in machine learning, and in fair machine learning speicfically.

This is important because we want to be able to communicate,like we call the horizontal and vertical axes of a plot the `x` and `y` axes.

The AIF 360 pacakge we are about to use and sklearn both use this notation.

- *target* or *labels*, denoted by for one sample (row) i $\mathbf{y_i}$.
- whole column of the target variable is Y
- "hat" notation for predictions/ output of prediction algorithm $\hat{y}_i$ and $\hat{Y}$
- "protected attribute" a_i and A

we use lowercase for one sample and uppercase for many.

```
help(skmetrics.accuracy_score)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[8], line 1
----> 1 help(skmetrics.accuracy_score)

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

12.4. Using AIF360

The AIF360 package implements fairness metrics, some of which are derived from metrics we have seen and some others. the documentation has the full list in a summary table with English explanations and details with most equations.

However, it has a few requirements:

- its constructor takes two `BinaryLabelDataset` objects
- these objects must be the same except for the label column
- the constructor for `BinaryLabelDataset` only accepts all numerical DataFrames

So, we have some preparation to do.

First, we'll make a numerical copy of the `compas_df` columns that we need. The only nonnumerical column that we need is race, wo we'll make a `dict` to replace that/

We need to used numerical values for the protected attribute. so lets make a mapping value

```
race_num_map = {r:i for i,r, in enumerate(compas_df['race'].value_counts().index)}
race_num_map
```

```
Cell In[9], line 1
----> 1 race_num_map = {r:i for i,r, in enumerate(compas_df['race'].value_counts().index)}
      2 race_num_map

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

```
compas_df['race'].replace(race_num_map)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[10], line 1
----> 1 compas_df['race'].replace(race_num_map)

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

We will also only use a few of the variables.

```
required_cols = ['race','two_year_recid','score_text_MedHigh']
num_compas = compas_df[required_cols].replace(race_num_map)
num_compas.head(2)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[11], line 2
      1 required_cols = ['race','two_year_recid','score_text_MedHigh']
----> 2 num_compas = compas_df[required_cols].replace(race_num_map)
      3 num_compas.head(2)

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

The scoring object requires that we have special data structures that wrap a DataFrame.

We need one aif360 binary labeled dataset for the true values and one for the predictions. ++

Next we will make two versions, one with race & the ground truth and ht eother with race & the predictions. It's easiest to drop the column we don't want.

The difference between the two datasets needs to be only the label column, so we drop the other variable from each small dataframe that we create.

```
num_compas_true = num_compas.drop(columns=['score_text_MedHigh'])
num_compas_pred = num_compas.drop(columns=['two_year_recid'])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[12], line 1
----> 1 num_compas_true = num_compas.drop(columns=['score_text_MedHigh'])
      2 num_compas_pred = num_compas.drop(columns=['two_year_recid'])

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

Now we make the `BinaryLabelDataset` objects, this type comes from AIF360 too. Basically, it is a DataFrame with extra attributes; some specific and some inherited from `StructuredDataset`.

> **Note**
>
> remember, you can inspect *any* object using the `__dict__` attribute

```
                                    df = num_compas_true,
                    label_names= ['two_year_recid'],
                   protected_attribute_names=['race'])
compas_predictions = BinaryLabelDataset(favorable_label=0,unfavorable_label=1,
                                        df = num_compas_pred,
                    label_names= ['score_text_MedHigh'],
                   protected_attribute_names=['race'])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[13], line 2
      1 # here we want actual favorable outcome
----> 2 broward_true = BinaryLabelDataset(favorable_label=0,unfavorable_label=1,
      3                                   df = num_compas_true,
      4                     label_names= ['two_year_recid'],
      5                    protected_attribute_names=['race'])
      6 compas_predictions = BinaryLabelDataset(favorable_label=0,unfavorable_label=1,
      7                                         df = num_compas_pred,
      8                     label_names= ['score_text_MedHigh'],
      9                    protected_attribute_names=['race'])

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

This type also has an `ignore_fields` column for when comparisons are made, since the requirement is that only the *content* of the label column is different, but in our case also the label names are different, we have to tell it that that's okay.

```
# beacuse our columsn are named differently, we have to ignore that
compas_predictions.ignore_fields.add('label_names')
broward_true.ignore_fields.add('label_names')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[14], line 2
      1 # beacuse our columsn are named differently, we have to ignore that
----> 2 compas_predictions.ignore_fields.add('label_names')
      3 broward_true.ignore_fields.add('label_names')

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

```
compas_fair_scorer = fairmetrics.ClassificationMetric(broward_true,
                                                       compas_predictions,
                                  unprivileged_groups=[{'race':0}],
                                 privileged_groups = [{'race':1}])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[15], line 1
----> 1 compas_fair_scorer = fairmetrics.ClassificationMetric(broward_true,
      2                                                       compas_predictions,
      3                                   unprivileged_groups=[{'race':0}],
      4                                  privileged_groups = [{'race':1}])

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

Now we can use the scores

```
compas_fair_scorer.accuracy()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[16], line 1
----> 1 compas_fair_scorer.accuracy()

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

By default, we get the overall accuracy. This calculation matches what we got using sklearn.

```
compas_fair_scorer.accuracy(True)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[17], line 1
----> 1 compas_fair_scorer.accuracy(True)

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

Here that is Caucasion people.

When `False` it's the unpriveleged group, here African American

```
compas_fair_scorer.accuracy(False)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[18], line 1
----> 1 compas_fair_scorer.accuracy(False)

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

These again match what we calculated before, the advantaged group (White) for True and disadvantaged group (Black) for False

```
compas_fair_scorer.error_rate_difference()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[19], line 1
----> 1 compas_fair_scorer.error_rate_difference()

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

the error rate alone does not tell the whole story because there are two types of errors. Plus there are even more ways we can think about if something is fair or not.

12.4.1. Disparate Impact

One way we might want to be fair is if the same % of each group of people (Black, $A = 0$ and White,$A = 1$) get the favorable outcome (a low score).

In Disparate Impact the ratio is of the positive outcome, independent of the predictor. So this is the ratio of the % of Black people not rearrested to % of white people rearrested.

$$D = \frac{\Pr(\hat{Y} = 1|A = 0)}{\Pr(\hat{Y} = 1|A = 1)}$$

This is equivalent to saying that the score is unrelated to race.

This type of fair is often the kind that most people think of intuitively. It is like dividing things equally.

```
compas_fair_scorer.disparate_impact()
```

```
Cell In[20], line 1
----> 1 compas_fair_scorer.disparate_impact()

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

US court doctrine says that this quantity has to be above .8 for employment decisions. Does COMPAS pass this criterion?

12.5. Equalized Odds Fairness

The journalists were concerned with the types of errors. They accepted that it is not the creators of COMPAS fault that Black people get arrested at higher rates (though actual crime rates are equal; Black neighborhoods tend to be overpoliced). They wanted to consider what actually happened and then see how COMPAS did within each group.

```
compas_fair_scorer.false_positive_rate(True)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[21], line 1
----> 1 compas_fair_scorer.false_positive_rate(True)

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

```
compas_fair_scorer.false_positive_rate(False)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[22], line 1
----> 1 compas_fair_scorer.false_positive_rate(False)

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

false positives are incorrectly got a low score.

This is different from how the problem was setup when we used sklearn because sklearn assumes tht 0 is the negative class and 1 is the “positive” class, but AIF360 lets us declre the favorable outcome(positive class) and unfavorable outcome (negative class)

White people were given a low score and then re-arrested almost twice as often as Black people.

Black people were given a low score and then re-arrested only a little more than half as often as white people. (White people were give an low score and rearrested almost twice as often)

To make a single metric, we might take a ratio. This is where the journalists found bias.

```
compas_fair_scorer.false_positive_rate_ratio()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[23], line 1
----> 1 compas_fair_scorer.false_positive_rate_ratio()

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

This metric would be fair with a value of 1.

got a high score and did not re-arrested as a percentage of those who got a high score

We can look at the other type of error

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[24], line 1
----> 1 compas_fair_scorer.false_negative_rate(True)

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

```
compas_fair_scorer.false_negative_rate(False)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[25], line 1
----> 1 compas_fair_scorer.false_negative_rate(False)

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

```
compas_fair_scorer.false_negative_rate_ratio()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[26], line 1
----> 1 compas_fair_scorer.false_negative_rate_ratio()

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

Black people were given a high score and not rearrested almost twice as often as white people.

So while the accuracy was similar (see error rate ratio) for Black and White people; the algorithm makes the opposite types of errors.

12.5.1. Average Odds Difference

This is a combines the two errors we looked at separately into a single metric.

$$\frac{1}{2}[(FPR_{A=\text{unprivileged}} - FPR_{A=\text{privileged}}) + (TPR_{A=\text{unprivileged}} - TPR_{A=\text{privileged}}))]$$

```
compas_fair_scorer.average_odds_difference()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[27], line 1
----> 1 compas_fair_scorer.average_odds_difference()

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

note if time, discuss:

- What should this look like if it is fair?
- what could this metric hide?

After the journalists published the piece, the people who made COMPAS countered with a technical report, arguing that that the journalists had measured fairness incorrectly.

The journalists two measures false positive rate and false negative rate use the true outcomes as the denominator.

12.6. Sufficiency and Calibration

We can look at their preferred metrics too

```
compas_fair_scorer.false_omission_rate(True)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[28], line 1
----> 1 compas_fair_scorer.false_omission_rate(True)

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

```
compas_fair_scorer.false_omission_rate(False)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[29], line 1
----> 1 compas_fair_scorer.false_omission_rate(False)

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

```
compas_fair_scorer.false_omission_rate_ratio()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[30], line 1
----> 1 compas_fair_scorer.false_omission_rate_ratio()

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

```
compas_fair_scorer.false_discovery_rate_ratio()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[31], line 1
----> 1 compas_fair_scorer.false_discovery_rate_ratio()

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

On these two metrics, the ratio is closer to 1 and much less disparate.

The creators thought it was important for the score to mean the same thing for every person assigned a score. The journalists thought it was more important for the algorithm to have the same impact of different groups of people.
Ideally, we would like the score to both mean the same thing for different people and to have the same impact.

Researchers established that these are mutually exclusive, provably. We cannot have both, so it is very important to think about what the performance metrics mean and how your algorithm will be used in order to choose how to prepare a model. We will train models starting next week, but knowing these goals in advance is essential.

Importantly, this is not a statistical, computational choice that data can answer for us. This is about *human* values (and to some extent the law; certain domains have legal protections that require a specific condition).

The Fair Machine Learning book's classification Chapter has a section on relationships between criteria with the proofs.

To put it all together, we can make a plot. First we'll make a DataFrame

```
           'name': 'false omission rate',
           'group':'sufficiency',
           'preferred_by':'COMPAS'},
          {'score':compas_fair_scorer.false_discovery_rate_ratio(),
           'name': 'false discovery rate',
           'group':'sufficiency',
           'preferred_by':'COMPAS'},
          {'score':compas_fair_scorer.false_positive_rate_ratio(),
           'name': 'false positive rate',
           'group':'separation',
           'preferred_by':'ProPublica'},
          {'score':compas_fair_scorer.false_negative_rate_ratio(),
           'name': 'false negative rate',
           'group':'separation',
           'preferred_by':'ProPublica'}]
ratio_df = pd.DataFrame(ratios)
ratio_df
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[32], line 1
----> 1 ratios = [{'score':compas_fair_scorer.false_omission_rate_ratio(),
      2            'name': 'false ommission rate',
      3            'group':'sufficiency',
      4            'preferred_by':'COMPAS'},
      5           {'score':compas_fair_scorer.false_discovery_rate_ratio(),
      6            'name': 'false discovery rate',
      7            'group':'sufficiency',
      8            'preferred_by':'COMPAS'},
      9           {'score':compas_fair_scorer.false_positive_rate_ratio(),
     10            'name': 'false positive rate',
     11            'group':'separation',
     12            'preferred_by':'ProPublica'},
     13           {'score':compas_fair_scorer.false_negative_rate_ratio(),
     14            'name': 'false negative rate',
     15            'group':'separation',
     16            'preferred_by':'ProPublica'}]
     17 ratio_df = pd.DataFrame(ratios)
     18 ratio_df

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

```
%matplotlib inline
```

```
sns.catplot(data=ratio_df,y='score',x='name',hue='preferred_by',
            kind='bar',aspect=2)
sns.lineplot(x = [-1,4],y=[1,1],color='black',legend=False)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[34], line 1
----> 1 sns.catplot(data=ratio_df,y='score',x='name',hue='preferred_by',
      2             kind='bar',aspect=2)
      3 sns.lineplot(x = [-1,4],y=[1,1],color='black',legend=False)

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

These are all ratios, so 1 is fair. COMPAS does okay on the measures it was designed around and poorly on the ones the journalists preferred.

```
compas_fair_scorer.false_omission_rate_difference()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[35], line 1
----> 1 compas_fair_scorer.false_omission_rate_difference()

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

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[36], line 1
----> 1 compas_fair_scorer.false_discovery_rate_ratio()

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

1. Assignment 1: Portfolio Setup, Data Science, and Python

Due:

Eligible skills: (links to checklists)

- ⭐[1] python level 1 and level 2
- ⭐process[2] level 1

1.1. Related notes

- Welcome and Introduction
- Syllabus and Python Review

1.2. To Do

Important

If you have trouble, check the GitHub FAQ on the left before e-mailing

Note

If you get stuck on any of this after accepting the assignment and creating a repository, you can create an issue on your repository, describing what you're stuck on and tag us with `@rhodyprog4ds/{{ ghinstructors }}`.

To do this click Issues at the top, the green "New Issue" button and then type away.

Your task is to:

1. Install required software from the Tools & Resource page
2. Create your portfolio, by
3. Learn about your portfolio from the README file on your repository.
4. edit `_config.yml` to set your name as author and change the logo if you wish
5. Fill in `about/index.md` with information about yourself(not evaluated, but useful) and your own definition of data science (graded for **level 1 process**)
6. Add a Jupyter notebook called `grading.ipynb` to the `about` folder and write a function that computes a grade for this course, with the docstring below.
7. Add the line `  - file: about/grading` in your `_toc.yml` file.

Important

Do not merge your "Feedback" Pull Request

1.2.1. Docstring

```
    '''
    Computes a grade for CSC/DSP310 from numbers of achievements at each level

    Parameters:
    ------------
    num_level1 : int
      number of level 1 achievements earned
```

```
    num_level3 : int
      number of level 3 achievements earned

    Returns:
    --------
    letter_grade : string
      letter grade with possible modifier (+/-)
    '''
```

1.2.2. Sample tests

Here are some sample tests you could run to confirm that your function works correctly:

```
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+'
```

Warning

remember the difference between side effects and returns

Note

when the value of the expression after `assert` is `True`, it will look like nothing happened. `assert` is used for testing

1.2.3. 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.

1.2.4. Grading Notes:

- a basic function that uses conditionals in python will earn **level 1 python**
- to earn **level 2 python** use pythonic code to write a loop that tests your function's correctness, by iterating over a list or dictionary. Remember you will have many chances to earn level 2 achievement in python, so you do not need to do this step for this assignment if you are not sure how.

---

[1] skills will be marked like this on the first time they are eligible. There will also be a ⌛ on skills for the last assignment they are eligible

[2] process is a special skill. You'll earn level 1 in this assignment or a soon one and two in either portfolio 1 or assignments 6-10, then level 3 in portfolio 2,3, or 4.

2. Assignment 2: Practicing Python and Accessing Data

due :

2.1. Objective & Evaluation

Eligible skills: (links to checklists)

- **first chance** access 1 and 2
- python 1 and 2
- summarize 1 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`.

2.2. Related notes

- Grading review, Pandas, and Iterables
- Pandas and Indexing

2.3. Setting

Next week, we are going to learn about summarizing data. In this assignment, you are going to build a small dataset about datasets. In class next week, we will combine all of your datasets about datasets together in order to be able to answer questions like:

- how much total data did you all load
- how many students picked the same dataset?
- how many total rows of data did each student load?

2.4. Tasks

First, . It contains a notebook with some template structure (and will set you up for grading).

2.4.1. Find Datasets

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.

In your notebook, create a markdown cell for each dataset that includes:

- heading of 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
- 1-2 questions you would like to answer with that dataset.

2.4.2. Store them for loading

Create a list of dictionaries in `datasets.py`, so that there is one dictionary for each dataset. Each dictionary should have the following keys:

`url`	with the url
`short_name`	a short name
`load_function`	(the actual function handle) what function should be used to load the data into a `pandas.DataFrame`.

2.4.3. 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:

> **Hint**
>
> See they python module docs for examples

1. 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.
2. record attributes about the dataset as in the table below in a list of lists or dictionary
3. Use that to create a DataFrame with columns that match the rows of the following table.

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

Meta Data Description of the DataFrame to build

2.4.4. Explore Your Datasets

In 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 last 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
- display the heading with the first three rows
- display the 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
- display the first 3 multiples of 3 rows (eg 3,6,9) of the data for two columns of your choice

2.4.5. Exploring data files

Continue in your `exploration.ipynb`. 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.

2.5. Submission

Upload to your repository.

2.6. Thinking ahead

Important

This section is not required, but is intended to help you get started thinking about ideas for your portfolio. 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 this after submitting the assignment, since you do not have to get a grade for it. If you want, you could discuss these ideas in office hours.

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: __

Important

You have the option to work with a partner. You must plan this in advance so that you have access to collaborate. If you did not find a partner in class and you would like one, try to find one on the class discussion. @brownsarahm if you do not get a reply.

3.1. Objective & Evaluation

This week your goal is to do a small exploratory data analysis for two datasets of your choice.

Eligible skills: (links to checklists)

- process 1
- access 1 and 2
- **first chance** summarize 1 and 2
- **first chance** visualize 1 and 2

3.2. Related notes

- Exploratory Data Analysis (EDA)
- Visualization

3.3. 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 last week, if they meet the criteria below.

3.3.1. 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

3.3.2. Dataset 2 (d2)

must include at least:

- two categorical variables **and**
- one continuous valued variable

Use a separate notebook for each dataset, name them `dataset_01.ipynb` and `dataset_02.ipynb`.

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. Explore the dataset in a notebook enough to describe its structure use the heading `## Description`
    - shape
    - columns
    - variable types
    - overall summary statisics
3. Write a short description of what the data contains and what it could be used for
4. Include overall summary for the data and interpret what that means. This should include code that generates the statistical summary and sentences in English in a markdown cell with your conclusions and explanation of the statistical summary. Are there limitations in how to safely interpre the data that the summary helps you see? are the variables what you expect?
5. Ask and answer 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.
6. 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.

3.4.1. 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

3.4.2. 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
- 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.

3.4.3. 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

> **Tip**
>
> Be sure to start early and use help hours to make sure you have a plan for all of these.

3.5. Peer Review

This is optional, but if you do a review, you only need to do one analysis each.

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.

In your review:

- Use inline comments to denote places that are confusing or if you see solutions to problems your classmate could not solve
- Use the template below for your summary review

3.5.1. Review Questions

1. How was the analysis overall to read? easy? hard? cohesive? jumpy?
2. Did the data summaries tell you enough about the data to understand the analysis and anticipate what kinds of questions could be answered? If not, what questions do you still have about the data?
3. Do the questions make sense based on the data? Are they interesting questions? What could improve the questions
4. Are the statistics and plots appropriate for the questions?
5. Are the interpretations complete, clear, and consistent with the statistics and plots?
6. What could be done to make the explanations more clear and complete?
7. What additional analysis might make the analysis more compelling and clear?

```
<!-- delete sections that are not needed -->
## Overall

This analysis was ...

## Data Summaries

- [ ] complete

To understand this analysis I still need to know ...

## Checklist

- [ ] questions fit the data
- [ ] questions are in natural language
- [ ] chosen statistics and plots match questions
- [ ] all statistics and plots have an interpretation in English

## Areas of improvement
```

3.5.2. Response

Respond to your review either inline comments, replies, or by updating your analysis accordingly.

Think Ahead

1. How could you make more customized summary tables?
2. Could you use any of the variables in this dataset to add more variables that would make interesting ways to apply split-apply-combine? (eg thresholding a continuous value to make a categorical value)

Warning

This section is not required, but is intended to help you get started thinking about ideas for your portfolio. 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.

4. Assignment 4:

Due:

- **first chance** prepare 1 and 2
- access 1 and 2
- python 1 and 2
- summarize 1 and 2
- visualize 1 and 2

4.1. Related notes

- Tidy Data and Reshaping Datasets
- Reparing values

4.2. 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.

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
- missing values
- a column that was continuous, but more interesting as a categorical
- too many header rows

> **Think Ahead**
>
> *this box is not required, but ideas for portfolio* cleaning a dataset to make it able to answer questions that were not possible could satisfy the level 3 prepare requirements.

4.3. Clean example datasets

There are notebooks in the template that have instructions for how to work with each dataset, including how to load it and what high level cleaning should be done. Your job is to execute.

To earn prepare level 2, clean any dataset and do just enough exploratory data analysis to show that the data is usable (eg 1 stat and/or plot).

To *also* earn python level 2: clean the CS degrees dataset (use a function or lambda AND loop or list/dictionary comprehension)

To *also* earn access level 2: clean the airline data (to get data in a second file type).

To *also* earn summarize and/or visualize level 2: add extra exploratory data analyses of your cleaned dataset meeting the criteria from the checklist (eg follow a3 checklists).

This means that if you want to earn prepare, python, and access, you will need to clean two datasets.

> **Hint**
>
> renaming things is often done well with a dictionary comprehension or lambda.

4.4. Study Cleaned Datasets

notebook in your repository. (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 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.

Important

Remember to run the “Submit” Workflow from the actions tab of your repository. see how on the How tos page

5. Assignment 5: Constructing Datasets and Using Databases

accept the assignment

Due: 2023-03-01

Eligible skills: (links to checklists)

- **first chance** construct 1 and 2
- ⧗ [1] access 1 and 2
- ⧗ [1] python 1 and 2
- ⧗ [1] prepare 1 and 2
- summarize 1 and 2
- visualize 1 and 2

5.1. Related notes

- Merging and Databases
- Web Scraping

5.2. Constructing Datasets

Your goal is to programmatically construct two ready to analyze datasets from multiple sources.

- Each dataset must combine at least 2 source tables(4 total source tables).
- At least one source table must come from a database or from web scraping.
- You should use at least three different joins(types of merges, or concat).

The notebook you submit should include:

- a motivating question for why you’re combining the datasets in an introduction section

Warning

Web scraping can be very open-ended. Start early so that you have time to get help if you get stuck.

- exploratory data analysis that shows why you built the data and confirms that is prepared enough to analyze.
- For one pair of tables, show how a different merge could answer a different question.

For construct, this can be very minimal EDA.

The notebook you submit should include:

- a motivating question for why you're combining the datasets in an introduction section
- code and description of how you built and prepared each 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.

For construct, 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. See your Achievement checklist for which aspects of prepare you still need, but 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

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 focus on the aspects of these achiements you have not successfully demonstrated.

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

> **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.

---

[1](1,2,3) skills will be marked like this on the last assignment they are eligible

Portfolio

```
aggregations is deprecated and will be removed in a future version. Use groupby instead.
df.sum(level=1) should use df.groupby(level=1).sum().
  assignment_dummies  = pd.get_dummies(rubric_df['assignments'].apply(pd.Series).stack()).sum(level=0)
/tmp/ipykernel_2022/1159722460.py:31: FutureWarning: Using the level keyword in DataFrame and Series
aggregations is deprecated and will be removed in a future version. Use groupby instead.
df.sum(level=1) should use df.groupby(level=1).sum().
  portfolio_dummies  = pd.get_dummies(rubric_df['portfolios'].apply(pd.Series).stack()).sum(level=0)
```

This section of the site has a set of portfolio prompts and this page has instructions for portfolio submissions.

Starting in week 3 it is recommended that you spend some time each week working on items for your portfolio, that way when it's time to submit you only have a little bit to add before submission.

The portfolio is your only chance to earn Level 3 achievements, however, if you have not earned a level 2 for any of the skills in a given check, you could earn level 2 then instead. The prompts provide a starting point, but remember that to earn achievements, you'll be evaluated by the rubric. You can see the full rubric for all portfolios in the syllabus. Your portfolio is also an opportunity to be creative, explore things, and answer your own questions that we haven't answered in class to dig deeper on the topics we're covering. Use the feedback you get on assignments to inspire your portfolio.

Each submission should include an introduction and a number of 'chapters'. The grade will be based on both that you demonstrate skills through your chapters that are inspired by the prompts and that your summary demonstrates that you *know* you learned the skills. See the formatting tips for advice on how to structure files.

On each chapter(for a file) of your portfolio, you should identify which skills by their keyword, you are applying.

You can view a (fake) example in this repository as a pdf or as a rendered website

Upcoming Checks

- Portfolio Check 1 is due March 6
- Portfolio Check 2 is due April 7
- Portfolio check 3 is due April 21
- Portfolio check 4 is due on our assigned final exam date

Portfolio check 1 will assess the following *new* achievements in addition to an a chance to make up any that you have missed:

	Level 3
keyword	
python	reliable, efficient, pythonic code that consistently adheres to pep8
access	access data from both common and uncommon formats and identify best practices for formats in different contexts
construct	merge data that is not automatically aligned
summarize	Compute and interpret various summary statistics of subsets of data
visualize	generate complex plots with pandas and plotting libraries and customize with matplotlib or additional parameters
prepare	apply data reshaping, cleaning, and filtering manipulations reliably and correctly by assessing data as received

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 will run and compile Jupyter notebooks

This page will cover a few basic tips.

Managing Files and version

You can either convert your ipynb files to earier to read locally or on GitHub.

The GitHub version means installing less locally, but means that after you push changes, you'll need to pull the changes that GitHub makes.

To manage with a precommit hook jupytext conversion

change your `.pre-commit-config.yaml` file to match the following:

```
repos:
-   repo: https://github.com/mwouts/jupytext
    rev: v1.10.0  # CURRENT_TAG/COMMIT_HASH
    hooks:
    - id: jupytext
      args: [--from, ipynb, --to, myst]
```

Run Precommit over all the files to actually apply that script to your repo.

```
pre-commit install
pre-commit run --all-files
```

If you do `git status` now, you should have a `.md` file for each `ipynb` file that was in your repository, now add and commit those.

Now, each time you commit, it will run jupytext first.

To manage with a gh action jupytext conversion

create a file at `.github/workflows/jupytext.yml` and paste the following:

```
name: jupytext

# Only run this when the master branch changes
on:
  push:
    branches:
    - main
    # If your git repository has the Jupyter Book within some-subfolder next to
    # unrelated files, you can make this run only if a file within that specific
    # folder has been modified.
    #
    # paths:
    # - some-subfolder/**

# This job installs dependencies, build the book, and pushes it to `gh-pages`
jobs:
  jupytext:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2

    # Install dependencies
    - name: Set up Python 3.7
      uses: actions/setup-python@v1
      with:
        python-version: 3.7

    - name: Install dependencies
      run: |
        pip install jupytext
    - name: convert
      run: |
```

```
      uses: EndBug/add-and-commit@v4 # You can change this to use a specific version
      with:
        # The arguments for the `git add` command (see the paragraph below for more info)
        # Default: '.'
        add: '.'

        # The name of the user that will be displayed as the author of the commit
        # Default: author of the commit that triggered the run
        author_name: Your Name

        # The email of the user that will be displayed as the author of the commit
        # Default: author of the commit that triggered the run
        author_email: you@uri.edu

        # The local path to the directory where your repository is located. You should use
actions/checkout first to set it up
        # Default: '.'
        cwd: '.'

        # Whether to use the --force option on `git add`, in order to bypass eventual gitignores
        # Default: false
        force: true

        # Whether to use the --signoff option on `git commit`
        # Default: false
        signoff: true

        # The message for the commit
        # Default: 'Commit from GitHub Actions'
        message: 'convert notebooks to md'

        # Name of the branch to use, if different from the one that triggered the workflow
        # Default: the branch that triggered the workflow (from GITHUB_REF)
        ref: 'main'

        # Name of the tag to add to the new commit (see the paragraph below for more info)
        # Default: ''
        tag: "v1.0.0"

      env:
        # This is necessary in order to push a commit to the repo
        GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }} # Leave this line unchanged
```

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

It is best practice to name files without spaces. Each `chapter` or file should have a descriptive file name (`with_no_spaces`) and descriptive title for it.

Syncing markdown and ipynb files

If you have the precommit hook working, git will call a script and convert your notebook files from the ipynb format (which is json like) to Myst Markdown, which is more plain text with some header information. The markdown format works better with version control, largely because it doesn't contain the outputs.

If you don't get the precommit hook working, but you do get jupytext installed, you can set each file to sync.

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:
```

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.

Portfolio Check 1 Ideas

Remember you'll be graded against the [rubric] and the [achievement checklists], but these are ideas for the structure.

You can mix and match different formats to cover the skills collectively.

If you goal is, for example, a B+ (you need 5 level 3s) you could only do 1-2 skills per portfolio check (there are 4).

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 all of the skills.

Be sure to check the checklists for how level 3s are more complex than level 2s. I recommend using office hours to help get ideas if you are not sure how to extend your analysis.

Several shorter reflections/analyses

You could also submit a few shorter pieces that in total cover all of the skills. Some example formats:

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

Correction & Reflection

If you had trouble with an assignment so far, 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. Then go a little deeper in exploring the topic in that context to also earn level 3.

Extension

assignments 2, 3, and 5.

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)

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 Beibhinn (beibhinn)
- 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?

There is no specific weight for any activities, because your grade is based on earning achievements for the skills listed in the skills rubric.

However, if you do not submit (or earn no achievements from) assignments or portfolios, the maximum grade you can earn is a C. If you do not submit (or earn no achievements from) your portfolio, the maximum grade you can earn is a B.

What time are assignments due?

End of day. I could start grading at any time the next morning. If your work is not there when I start grading it will not be graded, but if it is, I won't check the time stamp.

Can I submit this assignment late if …?

Late assignments are not accepted, however, your grade is based on the skills, not the assignments. All skills are assessed in at least two assignments, so missing any one will not hurt your grade. If you need an accommodation because you cannot submit multiple assignments, contact Dr. Brown.

I don't understand my grade on this assignment

If you have questions about your grade, the best place to get feedback is to reply on the Feedback PR. Either reply directly to one of the inline comments, or the summary.

Be specific about what you think you should have earned and why.

I can’t push to my repository, I get an error that updates were rejected

```
! [rejected] main -> main (fetch first)
error: failed to push some refs to <repository name>
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Your local version and github version are out of sync, you need to pull the changes from github to your local computer before you can push new changes there.

After you run

```
git pull
```

You’ll probably have to resolve a merge conflict

The content I added to my portfolio isn’t in the pdf

There was an error in the original `_toc.yml` file, change yours to match the following:

```yaml
format: jb-book
root: intro
parts:
  - caption: About
    chapters:
    - file: about/index
    - file: about/grading
#  - caption: Check 1
#    chapters:
#    - file: submission_1_intro
```

uncomment the later lines and add any new files you add.

My command line says I cannot use a password

GitHub has strong rules about authentication You need to use SSH with a public/private key; HTTPS with a Personal Access Token or use the GitHub CLI auth

My .ipynb file isn’t showing in the staging area or didn’t push

.ipynb files are json that include all of the output, including tables as html and plots as svg, so, unlike plain code files, they don’t play well with version control.

Your portfolio has `*/*.ipynb` in the `.gitignore` file, so that these files do not end up in your repository. Instead, you’ll convert your notebooks to Myst Markdown with jupytext via a precommit hook.

Your portfolio has the code to do this already, what you should do is make sure that `pre-commit` is installed and then run `pre-commit install`
(see your portfolio’s README.md file for more detail)

If this doesn’t work, you can follow the alterntive in the porfolio readme.

If that doesn’t work, and you have time before the deadline, create an issue to get help.

My portfolio won't compile

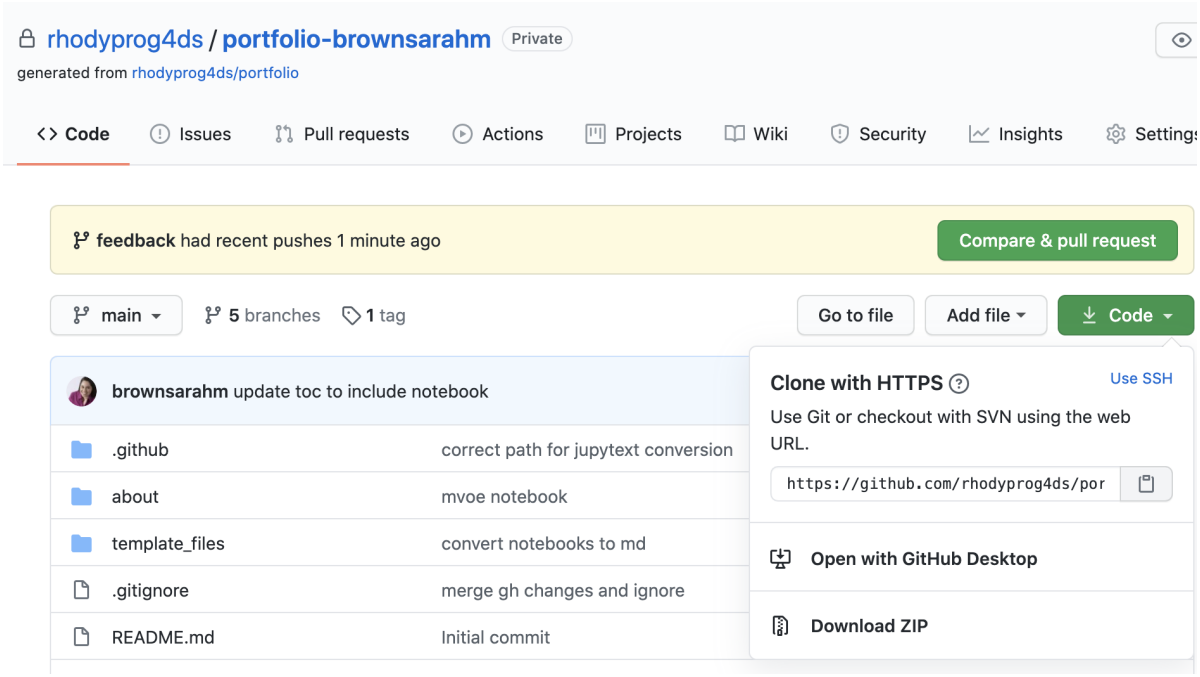
If there's an error your notebook it can't complete running. You can allow it to run if the error is on purpose by changing settings as mentioned on the formatting page.

Help! I accidentally merged the Feedback Pull Request before my assignment was graded

That's ok. You can fix it.

You'll have to work offline and use GitHub in your browser together for this fix. The following instuctions will work in terminal on Mac or Linux or in GitBash for Windows. (see Programming Environment section on the tools page).

First get the url to clone your repository (unless you already have it cloned then skip ahead): on the main page for your repository, click the green "Code" button, then copy the url that's show



Next open a terminal or GitBash and type the following.

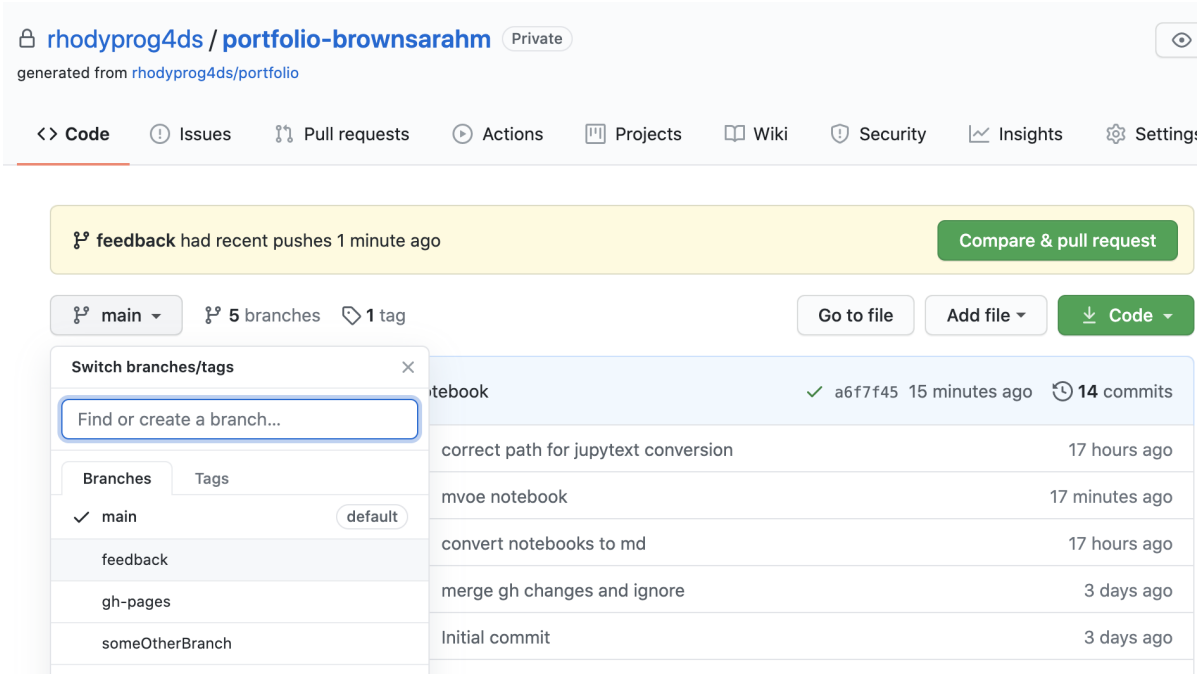
```
git clone
```

then past your url that you copied. It will look something like this, but the last part will be the current assignment repo and your username.

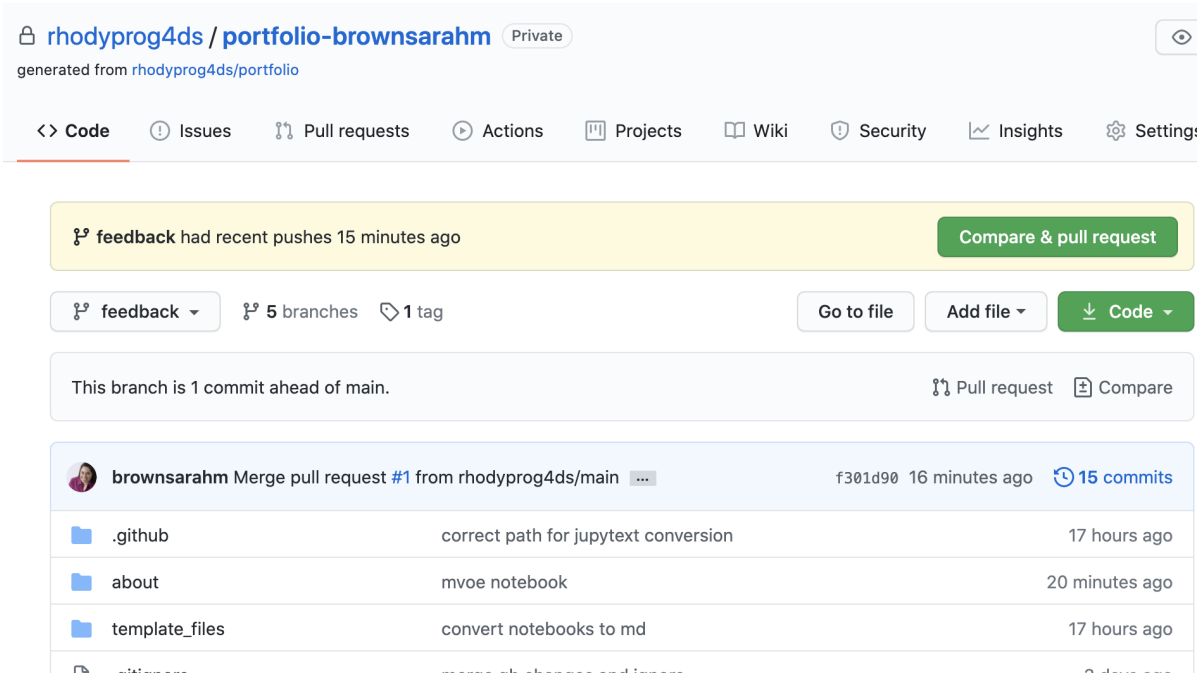
```
git clone https://github.com/rhodyprog4ds/portfolio-brownsarahm.git
```

When you merged the Feedback pull request you advanced the `feedback` branch, so we need to hard reset it back to before you did any work. To do this, first check it out, by navigating into the folder for your repository (created when you cloned above) and then checking it out, and making sure it's up to date with the `remote` (the copy on GitHub)

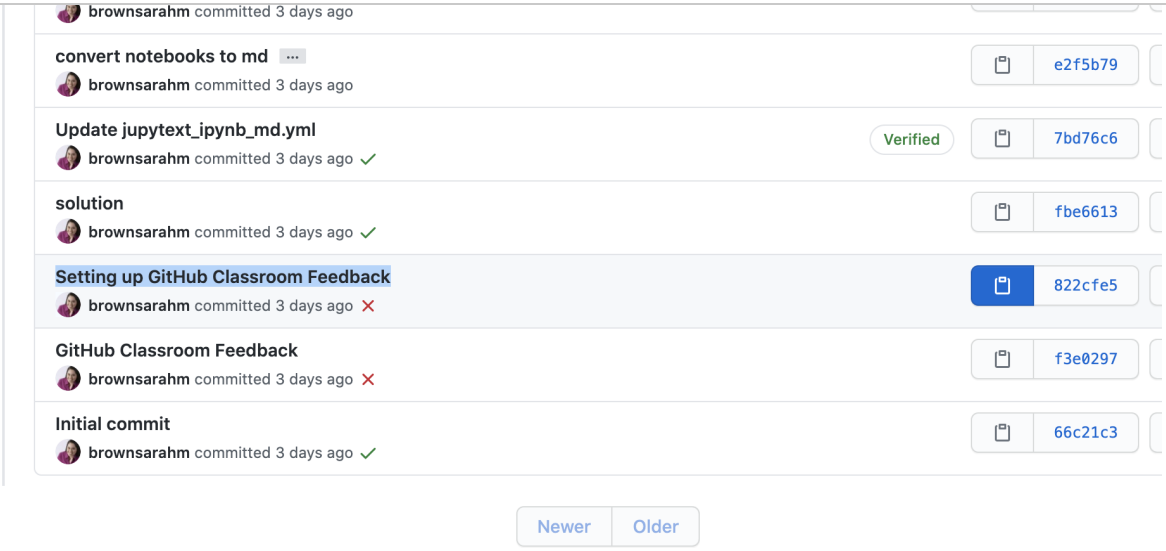
```
cd portfolio-brownsarahm
git checkout feedback
git pull
```



Now view the list of all of the commits to this branch, by clicking on the clock icon with a number of commits



On the commits page scroll down and find the commit titled “Setting up GitHub Classroom Feedback” and copy its hash, by clicking on the clipboard icon next to the short version.



Now, back on your terminal, type the following

```
git reset --hard
```

then paste the commit hash you copied, it will look something like the following, but your hash will be different.

```
git reset --hard 822cfe51a70d356d448bcaede5b15282838a5028
```

If it works, your terminal will say something like

```
HEAD is now at 822cfe5 Setting up GitHub Classroom Feedback
```

but the number on yours will be different.

Now your local copy of the `feedback` branch is reverted back as if you had not merged the pull request and what's left to do is to push those changes to GitHub. By default, GitHub won't let you push changes unless you have all of the changes that have been made on their side, so we have to tell Git to force GitHub to do this.

Since we're about to do something with forcing, we should first check that we're doing the right thing.

```
git status
```

and it should show something like

```
On branch feedback
Your branch is behind 'origin/feedback' by 12 commits, and can be fast-forwarded.
  (use "git pull" to update your local branch)
```

Your number of commits will probably be different but the important things to see here is that it says `On branch feedback` so that you know you're not deleting the `main` copy of your work and `Your branch is behind origin/feedback` to know that reverting worked.

Now to make GitHub match your reverted local copy.

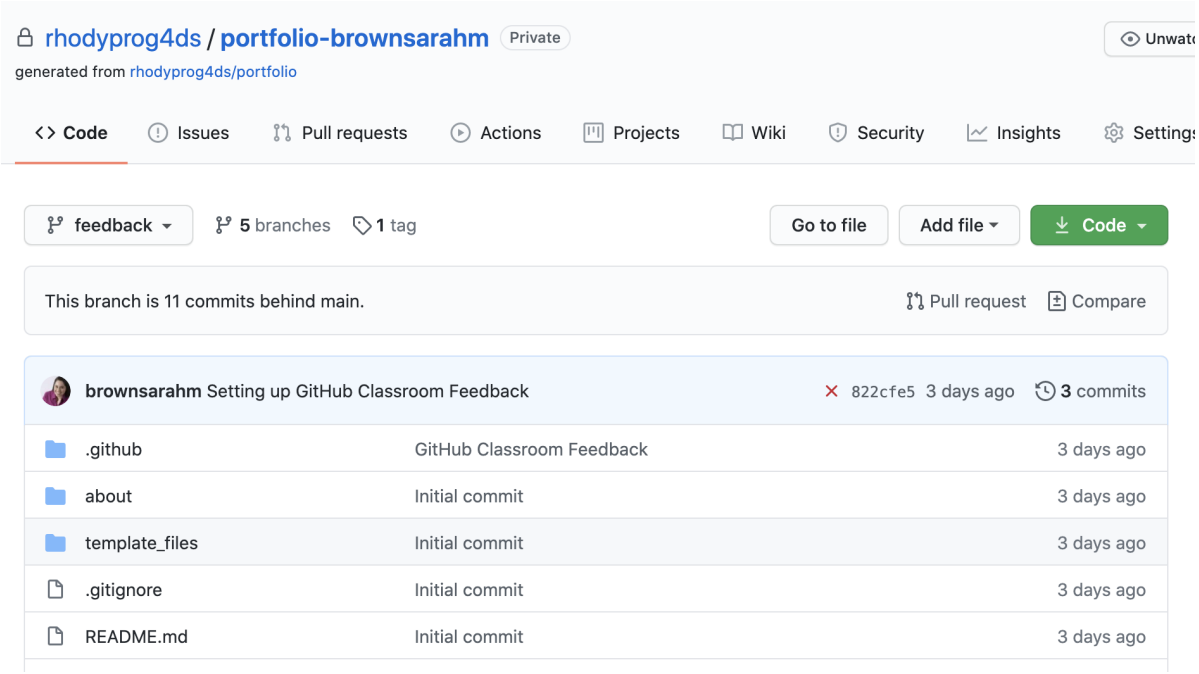
```
git push origin -f
```

and you'll get something like this to know that it worked

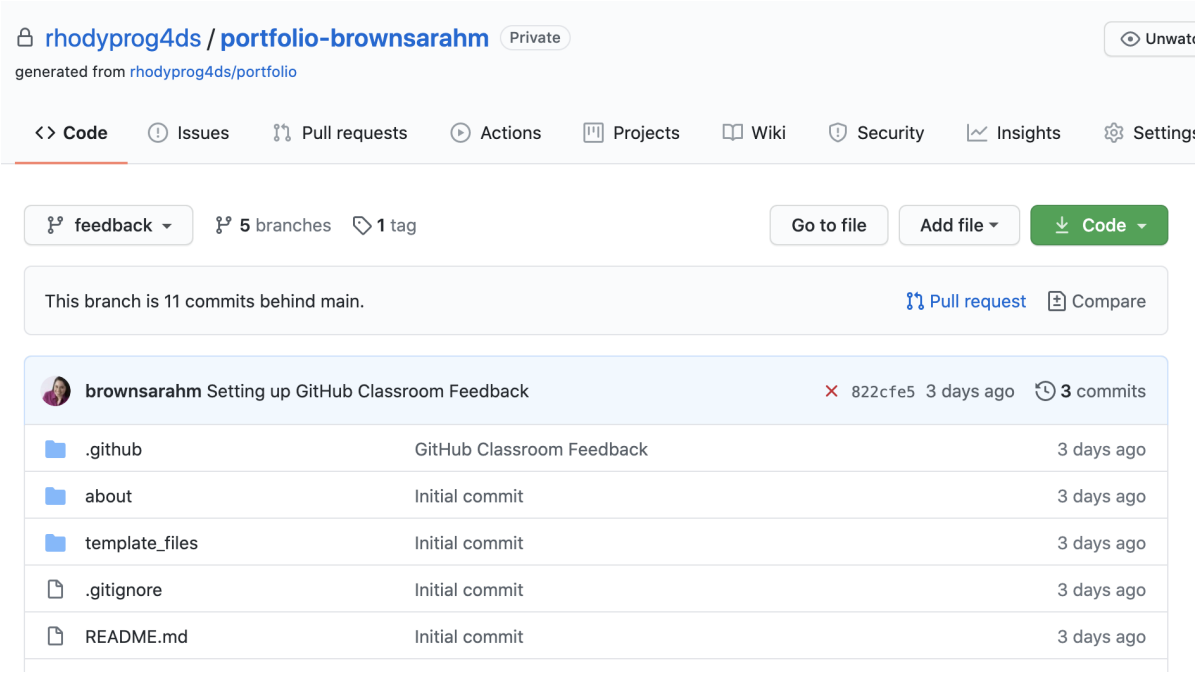
```
 + f301d90...822cfe5 feedback -> feedback (forced update)
```

Again, the numbers will be different and it will be your url, not mine.

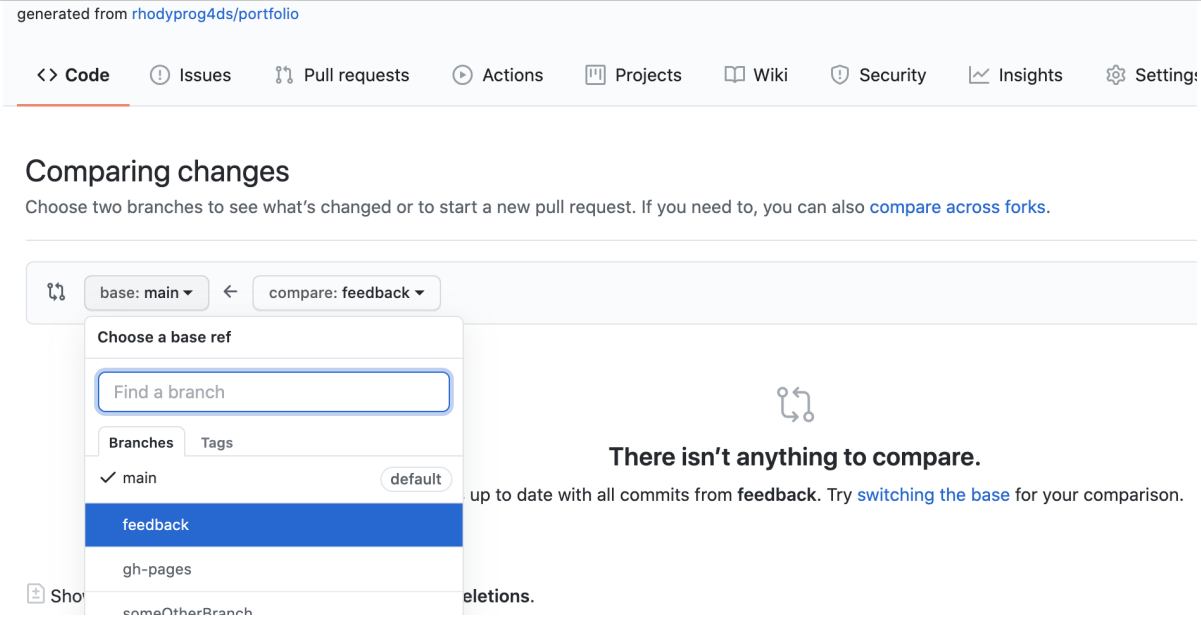
Now back on GitHub, in your browser, click on the code tab. It should look something like this now. Notice that it says, “This branch is 11 commits behind main” your number will be different but it should be 1 less than the number you had when you checked `git status`. This is because we reverted the changes you made to main (11 for me) and the 1 commit for merging main into feedback. Also the last commit (at the top, should say “Setting up GitHub Classroom Feedback”).



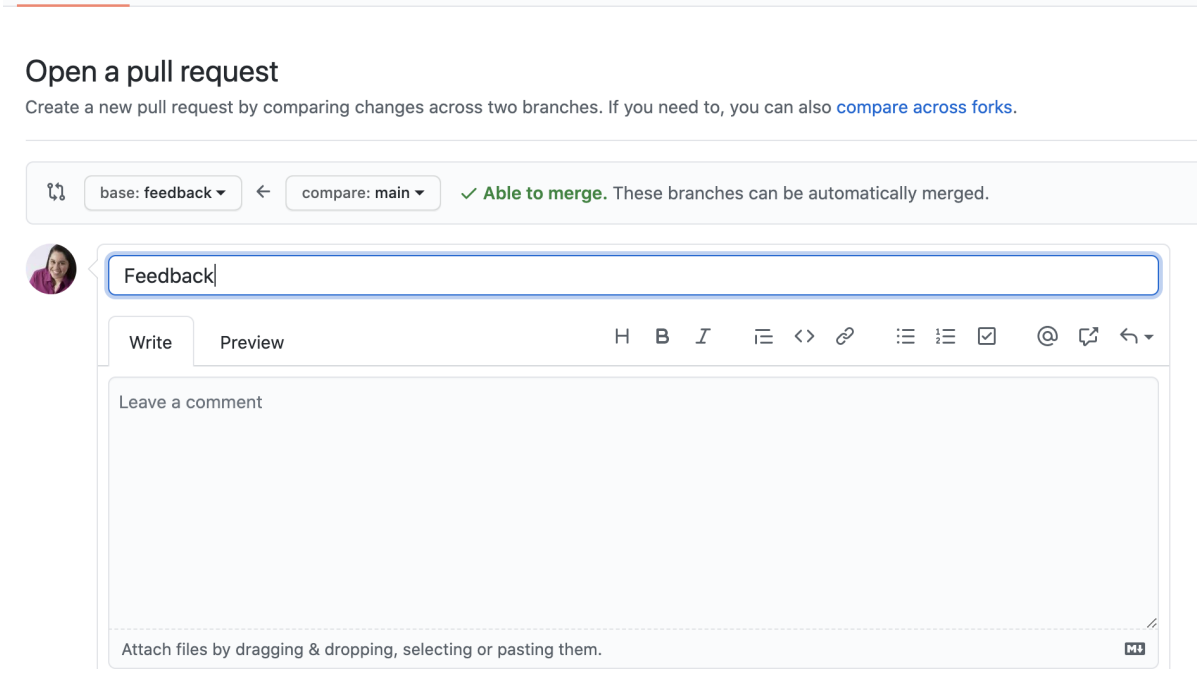
Now, you need to recreate your Pull Request, click where it says pull request.



It will say there isn’t anything to compare, but this is because it’s trying to use `feedback` to update `main`. We want to use `main` to update `feedback` for this PR. So we have to swap them. Change base from `main` to `feedback` by clicking on it and choosing `feedback` from the list.



Then the change the compare `feedback` on the right to `main`. Once you do that the page will change to the "Open a Pull Request" interface.



Make the title "Feedback" put a note in the body and then click the green "Create Pull Request" button.

Now you're done!

If you have trouble, create an issue and tag `@@rhodyprog4ds/fall22instructors` for help.

Code Errors

Key Error

If you get a key error for a pandas operation, it means that the column name as you typed it is not in the DataFrame. Check the spelling, leading or trailing whitespace can be especially troubling.

You're probably missing `()` on a method, so Python returned the method itself as an object instead of calling it and returning the output.

Glossary

> **Ram Token Opportunity**
>
> Contribute glossary items and links for further reading using the suggest an edit button behind the GitHub menu at the top of the page.

aggregate

to combine data in some way, a function that can produce a customized summary table

anonymous function

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

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

any object in python that can return its members one at a time. The most common example is a list, but 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>`

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

suffix

additional part of the name that gets added to end of a name in a merge operation

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

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

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)

TraceBack

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.

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

use it to check which way is which.

```
df.sum(axis=0)
```

```
A    7
B    8
dtype: int64
```

```
df.sum(axis=1)
```

```
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

- Lots of wikipedia pages have tables in them.

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
- Caribou Tracking
- Video games from steam 2021 and from 2019
- BBC Rap Artists
- character psychometrics
- weather forecast accuracy

Datasets with time

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.

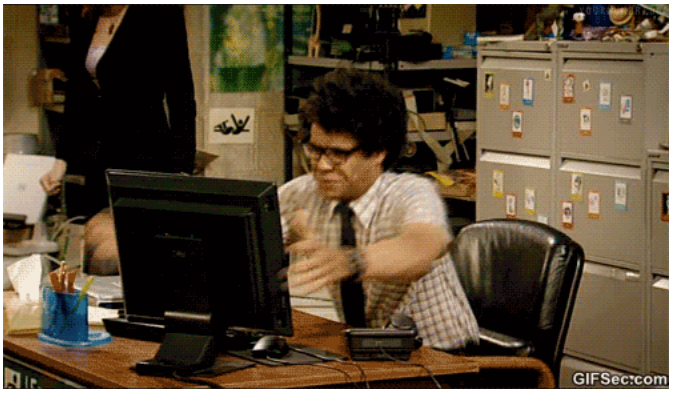
on email

- 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 book that might be of interest if you find programming classes hard is the Programmers Brain As of 2021-09-07, it is available for free by clicking on chapters at that linked table of contents section.

> **Where are your help tools?**
>
> In Python and Jupyter notebooks, what help tools do you have?

Learning in class

> 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

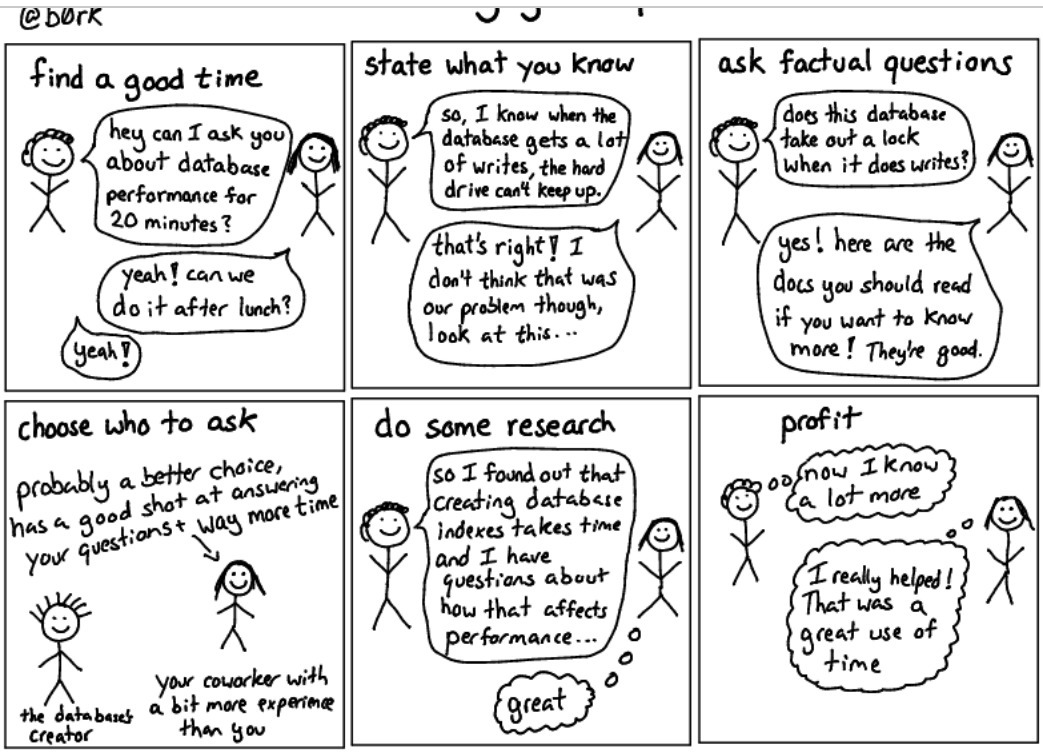
In assignments, you will be asked to practice with specific concepts at an intermediate level. Assignments will apply the concepts from class with minimal extensions. You will probably need to use help funcitons and read documentation to complete assignments, but mostly to look up things you saw in class and make minor variations. Most of what you need for assignments will be in the class notes, which is another reason to read them after class.

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

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.

> **Note**
>
> A fun version of this is rubber duck debugging

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.

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}
> ```
> ```

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.

> **Note**
>
> We know that we don't currently teach a lot of this in our department, so in Spring 22 I'm teaching a brand new course on Computer Systems, that will help you understand the underlying concepts that make all of this stuff make sense, instead of just following recipes and debugging here and there.

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.

> If, for example, you come to me in week 5 and have never got an any environment working and you're trying for the first time, your grade will be hurt because you will be very far behind at that point. Ask for help early and 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.

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.

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.

Letters to Future students

This section is a place for students enrolled in Fall 2020 to write letters to future students taking this class with Professor Brown. The websites for future sections will link back here for them to read.

Contributed Notes

Reviewing notes

Especially when it comes to the difficult topics make sure you go back through the notes and try and add to them yourself. Actively using the new topics will help you learn a lot better than just copying the notes.

Attend Office Hours

Attending office hours will help you better understand course material, complete homework assignments, and learn new things that might not normally come up in class.

- David

To contribute

Via GitHub directly:

1. Use the edit button above to add a note to this file following the example that's commented out
2. create a pull request