Welcome to Programming for Data Science

Welcome to the course manual for CSC310 at URI.

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

Course Calendar with zoom links on BrightSpace



subscribe to that calendar in your favorite calendar application

Syllabus

Welcome to CSC/DSP310: Programming For Data Science.

In this syllabus you will find an overview of the course, information about your instructor, course policies, restatements of URI policies, reminders of relevant resources, and a schedule for the course.

About

About this course

Data science exists at the intersection of computer science, statistics, and machine learning. 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.

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 substantial 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 this semester

This semester is a lot of new things for all of us. This course will be completely online all semester, so we will get to use a single instructional format all semester, including when all campus activities move remote after Thanksgiving. I recognize that those last two weeks of the semester may change your obligations with siblings, parents, work, etc. In light of that, we will cover all of the most important topics and you will have the opportunity to achieve all of the course learning outcomes before Thanksgiving. The material in the last two weeks of the semester will be more advanced, likely interesting and definitely useful material, but if your ability to participate in class is less at that time, it will not hurt your grade.

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.

: ■ Contents

Syllabus

About

Tools and Resources

Grading

Learning Objective, Schedule, and

Rubric

Support

Policies

Class Notes

Class 2: intro to notebooks and python

Class 3: Welcome to Week 2

Class 4: Pandas

Class 5: Accessing Data, continued

Class 6: Exploratory Data Analysis

Class 7: Visualization for EDA

Class 8: Visualization and Starting

<u>to Clean Data</u>

Class 9: Preparing Data For

<u> Analysis</u>

Class 10: Cleaning review and Ray

Summit Keynotes

Class 11: Cleaning Data

Class 12: Constructing Datasets

from Multiple Sources

<u>Class 13: Data from mulitple</u> <u>sources and Databases</u>

<u>Assignments</u>

<u>Assignment 1: Portfolio Setup, Data</u>

Science, and Python

Assignment 2: Practicing Python

and Accessing Data

Assignment 3: Exploratory Data

<u>Analysis</u>

Assignment 4: Preparing Data for

Analysis

Assignment 5: Constructing

Datasets and Using Databases

<u>Portfolio</u>

Formatting Tips

Reflective Prompts

Analysis Prompts

FAC

Syllabus FAQ

<u>GitHub FAQ</u>

Resources

General Tips and Resources

References on Python

Data Sources

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

About your instructor

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

Dr. Brown is a new 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.

The best way to contact me 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.

Note

Important

TL;DR [1]

· check Brightspace

Make a GitHub AccountInstall Python

Install ZoomSetup your URI Zoom

Account

• Log in to Prismia Chat

Install Git

Whether you use CSC or DSP does not matter.

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 or Chromebook will be able to do all of the things required in this course.

All of the tools 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 <u>Brightspace site</u>. This is also where your grades will appear.

Zoom

This is where we will meet for synchronous class sessions. 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 <u>Zoom client</u> on your computer. Please <u>log in</u> and <u>configure</u> <u>your account</u>. 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.

Prismia chat

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

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

Course Manual

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/BrownFall20/. Links to the course reference text and code documentation will also be included here in the assignments and class notes.

GitHub Classroom

You will need a GitHub Account. If you do not already have one, please create one by the first day of class. There will be a link to our class GitHub Classroom on Brightspace.

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, etc)
- · Cit
- A web browser compatible with Jupyter Notebooks

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

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.

Textbook

The text for this class is a reference book and iwll 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 of topics.

Python for Data Science is available free online:

[1] Too long; didn't read.

Grading

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

Principles of Grading

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

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

- Earning a C in this class is intended to be easier than typical grading. I expect everyone to get at least a C.
- Earning a B in this class is intended to be very accessible, you can make a lot of mistakes along the way as you learn, as long as you learn by the end.
- Earning an A in this class will be challenging, but is possible even with making mistakes while you learn.

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.

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.

1 Note

I use atom, but I decided to use it by downloading both Atom and Sublime and trying different things in each for a week. I liked Atom better after that and I've stuck with it since. I used Atom to write all of the content in this syllabus.

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.

Participation

While attending synchronous class sessions, there will be understanding checks and in class exercises. Completing in class exercises and correctly answering questions in class can earn level 1 achievements. In class questions will be administered 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.

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



🛕 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.

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.

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

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

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

The letter grade can be computed as follows

```
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
```



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

```
compute_grade(15,15,15)

'A'

compute_grade(14,14,14)

'C-'

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

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

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

Late work

No late work will be graded. Every skill will be assessed through more than one assignment, so missing assignments occasionally *may* not 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.

Examples

Important

The following will make more sense after you read the next section of the syllabus and see the skills rubric sections.

If you always attend and get everything correct, you will earn and A and you won't need to submit the 4th portfolio check or assignment 13.

Getting A Without Perfection







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.

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

Map to a B, having trouble

	Level 1	Level 2	Level 3
python	A1	P1	
process	A1	P2	
access	A2	P1	
construct	A5	P1	
summarize	A3	P1	
visualize	А3	P2	
prepare	A5	P2	
classification	A10	P3	
regression	A11	P2	
clustering	A9	P3	
evaluate	A11	P3	
optimize	A11	P4	1
compare	A13	P3	
unstructured	A13	P4	1
tools	A13	P3	



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.

Learning Objective, Schedule, and Rubric

Learning Outcomes

There are five learning outcomes for this course.

- 1. (process) Describe the process of data science, define each phase, and identify standard tools
- 2. (data) Access and combine data in multiple formats for analysis
- $3. \, (exploratory) \, Perform \, exploratory \, data \, analyses \, including \, descriptive \, statistics \, and \, visualization \, analyses \, including \, descriptive \, statistics \, and \, visualization \, descriptive \, statistics \, descriptive \, desc$
- 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 MWF 1-1:50pm on Zoom. Every class will include participatory live coding (instructor types, students follow along)) instruction and small exercises for you to progress toward level 1 achievements of the new skills introduced in class that day.

Programming assignments that will be due each week Sunday by 11:59pm.



On the <u>BrightSpace calendar</u> 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.

	topics	skills
week		
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	$Modeling, Naive\ Bayes, classification\ performance\ metrics$	[classification, evaluate]
7	decision trees, cross validation	[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	Topic Modeling	[unstructured, tools]
14	Deep Learning	[tools, compare]

Skill Rubric

The skill rubric 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.

skill		Level 1	Level 2	Level 3
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 occur
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
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
classification	Apply classification	identify and describe what classification is, apply pre-fit classification models	fit 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	can fit linear regression models	can fit and explain nonlinear regression
clustering	Clustering	describe what clustering is	apply basic clustering	apply multiple clustering techniques, and interpret results
evaluate	Evaluate model performance	Explain basic performance metrics for different data science tasks	Apply basic model evaluation metrics to a held out test set	Evaluate a model with multiple metrics and cross validation
optimize	Optimize model parameters	Identify when model parameters need to be optimized	Manually 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

	skill	Level 1	Level 2	Level 3
keyword				
unstructured	model unstructured data	Identify options for representing text data and use them once data is tranformed	Apply at least one representation to transform unstructured data for model fitting or summarizing	apply mulitple representations and compare and contrast them for different end results
tools	use industry standard data science tools and workflows to solve data science problems	Solve well strucutred problems with a single tool pipeline	Solve semi- strucutred, completely specified problems with multiple tools	Scope, choose an appropriate tool pipeline and solve data science problems

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.

	A1	A2	А3	Α4	А5	A6	Α7	A8	Α9	A10	A11	A12	A13	# Assignments
keyword														
python	1	1	1	1	0	0	0	0	0	0	0	0	0	4
process	1	1	0	0	0	0	0	0	0	0	0	0	0	2
access	0	1	1	1	0	0	0	0	0	0	0	0	0	3
construct	0	0	0	0	1	1	0	0	0	0	0	0	0	2
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
classification	0	0	0	0	0	1	1	0	0	1	0	0	0	3
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
evaluate	0	0	0	0	0	0	0	0	0	1	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
unstructured	0	0	0	0	0	0	0	0	0	0	0	1	1	2
tools	0	0	0	0	0	0	0	0	0	1	1	1	1	4

Portfolios and Skills

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

	Level 3	P1	P2	Р3	P4
keyword					
python	$reliable, efficient, pythonic code that \ consistently \ adheres \ to \ pep 8$	1	1	0	0
process	Compare different ways that data science can occur	0	1	1	0
access	access data from both common and uncommon formats and identify best practices for formats in different contexts	1	1	0	0
construct	merge data that is not automatically aligned	1	1	0	0
summarize	Compute and interpret various summary statistics of subsets of data	1	1	0	0
visualize	generate complex plots with pandas and plotting libraries and customize with matplotlib	1	1	0	0
prepare	apply data reshaping, cleaning, and filtering manipulations reliably and correctly by assessing data as received	1	1	0	0
classification	fit and apply classification models and select appropriate classification models for different contexts	0	1	1	0
regression	can fit and explain nonlinear regression	0	1	1	0
clustering	apply multiple clustering techniques, and interpret results	0	1	1	0
evaluate	Evaluate a model with multiple metrics and cross validation	0	1	1	0
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
unstructured	apply mulitple representations and compare and contrast them for different end results	0	0	1	1
tools	Scope, choose an appropriate tool pipeline and solve data science problems	0	0	1	1

Support

Academic Enhancement Center

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 through drop-in centers and small group tutoring. 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.

Policies

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 process for requesting reasonable accommodations in the classroom. DSS can be reached by calling: 401-874-2098, visiting: web.uri.edu/disability, or emailing: dss@etal.uri.edu. They are available to meet with students enrolled in Kingston as well as Providence courses.

Academic Honesty

Students are expected to be honest in all academic work. A student's name on any written work, quiz or exam shall be 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 students. At this uncertain time, those concerns include minimizing the potential spread of COVID-19 within our community. While the university has worked this summer 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. Students are required to comply with Rhode Island state laws, including the Rhode Island Executive Orders related to health and safety, ordinances, regulations, and guidance adopted by the University as it relates to public health crises, such as COVID-19.

An addendum on policies and guidelines concerning your obligations during this crisis has recently been integrated into the Student Handbook. These obligations include:

- Wearing of face masks by all community members when on a URI campus in the presence of others
- Maintaining physical distancing of at least six feet at all times
- Following state rules on the number of individuals allowed in a group gathering
- · Completing a daily health self-assessment also available through the Rhody Connect app before coming to campus
- $\bullet \quad \text{Submitting to COVID-19 testing as the University monitors the health of our community} \\$
- Following the University's quarantine and isolation requirements

If you answer yes to any of the questions on the daily health assessment, do not go to campus. YOU MUST STAY HOME/IN YOUR ROOM and notify URI Health Services via phone at 401-874-2246 immediately.

If you are already on campus and start to feel ill, you need to remove yourself from the public and notify URI Health Services via phone immediately at 401-874-2246 and go home/back to your room and self-isolate while you await direction from Health Services.

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

Class Notes

Class notes will get posted here day by day

- 2020-09-11: Jupyter Notebook tour, conditionals, functions
- 2020-09-14: Iterables, Pandas
- 2020-09-16: Pandas loading and exploring
- 2020-09-18: Pandas, Functions as Object, Dictionaries
- 2020-09-21: Exploratory Data Analysis, Split, apply, Combine
- 2020-09-23: Visualization for EDA
- 2020-09-25: Viz & Cleaning
- 2020-09-28: Preparing data for analysis
- 2020-09-30: Reading complex data and Ray Summit
- 2020-10-02: Cleaning Data
- 2020-10-05: Merging DataFrames
- <u>2020-10-05</u>: Merging & Databases

Class 2: intro to notebooks and python

Agenda:

- 1. review Data Science
- 2. jupyter notebook
- 3. python: conditionals and functions

Jupyter Notebooks

To launch a Jupyter notebook, in your Anaconda prompt on Windows or terminal on Linux or Mac:

cd dir/you/want/to/work/**in**jupyter notebook

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

In []: # This is a comment in a code cell

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



If you press enter (or return) or click on the cell it changes to edit mode. The border is green in edit mode



Type code or markdown into boxes called cells. 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.



You can treat markdown cells like plain text, or use special formatting. Here is a <u>markdown cheatsheet</u>

Code cells can run like a calculator. If there is a value returned by the last line of a cell, it will be displayed.

```
9
```

For example, when we assign, python "returns" ${\bf None}$ so there is no output from this cell

```
a = 9
```

But this one does display the value of the cell

```
9

b = 4
b

a

4
```

Getting Help in Python and Jupyter

The standard way to get help in the help function

```
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.
```

There are two special ways to get help in Jupyter, one dynamically while you're working and one that stays displayed for a while

Python comments are indicated by the # symbol.

```
print() # shift +tab to view help
```

That will look like this:

```
In [ ]:

In [2]: print() # shift +tab to view help

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

Prints the values to a stream, or to sys.stdout by default.
```

Press tab twice for the longer version.

A Question mark puts it in a popup window that stays until you close it

```
print?

In [3]: print?

Docstring:
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.
Type: builtin_function_or_method
```

This means you can then use the displayed help to remember how to call the function

```
print(a,b, 'hello',sep='-')

9-4-hello

a
b
```

Note

Here in class we changed the value of a above and noted that the new value is shows here, but not in the previous cell that had ouput the value of a

Note that these are green in the jupyter notebook because they're python reserved words.



if the *concept* of an if in programming is new to you, you should talk to Dr. Brown. Basic programming is a prerequisite to this course, we're *reviewing* basic ideas but only at a level of detail to serve as a reminder.

The synta

```
if a >b:
    print('greater')

greater

if b > a:
    print('greater')
```



this is updated to include things that were skipped in class and discussed after the breakouts are discussed after the breakouts and discussed after the breakouts are discussed after the breakouts and discussed after the breakouts are discussed after the breakouts and discussed after the breakouts are discussed after the breakouts are discussed after the breakouts are discussed after the breakouts and discussed after the breakouts are discuss

You can check the contents of a string with the in keyword

```
name = 'sarah'
if 'a' in name:
    print(name, 'has an a')

sarah has an a
```

if we copy and change the name we get no output

```
name = 'Beibhinn'
if 'a' in name:
   print(name, 'has an a')
```

Functions



this is also updated to include things that were skipped in class and discussed after the breakouts

How to write functions in python:

- the def keyword starts a function definition
- then the function name
- then the parameters it accepts in ()
- end that line with a :
- the body of the function is spaced over one tab, but Jupyter will do it automatically for you. if it doesn't you might have forgotten the:

```
def greeting(name):
    a function that greets the person name by printing
    parameters
    .....
    name: string
        a name to be greeeted

    Returns
    ....
    nothing
    rint('hello', name)
```

```
greeting('sarah')
```

🥊 Tip

in works for all lists, we'll learn more about that next week



this is actually not a great function, because the printing is only a *side effect*. It's better to return the output of the function

```
hello sarah
```

A better version of that function might be:

```
def greeting(name):
    a function that greets the person name by printing

parameters
    name: string
    a name to be greeeted

Returns
    nothing
    '''
    return 'hello ' + name
```



1 Try it Yourself!

Write a function that checks if a string has a space in it and returns "please rename" if there is a space.

Remember a docstring. Call your function a couple of times to confirm it works.

Unhide the cell below to see the answer.

This is what some calls of the function look lik

```
check_string("my data.csv")

'please rename'
```

If there's no string we see no output

```
check_string("my_data.csv")
```

What does python actually return?

```
NoneType
```

Further Reading

- How Ipython Works
- <u>Ipython Overiver</u>
- <u>Jupyter Notebooks Technical Overview</u>
- Python If Statements
- Python Functions

Class 3: Welcome to Week 2

This week we will:

- clarify how this grading really works
- learn about accessing data
- use accessing data as motivation to review more python

- · Solution function posted.
- note: not a sum
- read the rubric
- Brightspace will show grades as they're earned
- In class, respond on prismia
- Portfolio
 - o will start posting prompts The docstring functions like a property of the function object, so it has to be inside.

Iterables

Python has a general data type for objects that are designed to facilitate repetition of some sort, they're called iterables

We've already seen one. Strings are Iterables

```
name = 'sarah'
```

which means we can index them

```
name[3]
```

● Tip

from 0

remember python indexes

Indexing with a negative number counts from the end

```
name[-1]
'h'
```

Loops in python have similar syntax to the if and functions we saw last week:

```
for char in name:
    print(char*3)

sss
aaa
rrr
aaa
hhh
```

some notes:

- char is called the loop variable
- name is called the collection- this can be any iterable type object in python
- print(char*3) is called the loop body
- python lets us use mathematical operations on strings

Lists and List Comprehensions

We make a list with square brackets

```
names = ['sarah', 'Jose', 'Cam', 'Bri']
```

we can also build lists by folding a loop into the list construction

```
['hello' + n for n in names]
['hellosarah', 'helloJose', 'helloCam', 'helloBri']
```

this is called a list comprehension

```
greetings = ['hello ' + n for n in names]
```

```
greetings[0]
'hello sarah'
```

Dictionaries

Dictionaries are a useful datatype in python. It is denoted by {} and contains key: value pairs separated by commas.

You can think of it like a list of the values with a named index.

```
gh_names['jdion62']

'Jacob Dion'
```

we can iterate over both the key and the value by using the items method on a dictionary. That makes another iterable object that can be used as a loop collection. It functions as a set of pairs now, so we get two loop variables:

```
for key, value in gh_names.items():
    print(value, "'s username is ", key)

Sarah Brown 's username is brownsarahm
Brianna MacDonald 's username is briannakathrynm1
Jacob Dion 's username is jdion62
```

If we iterate over the dictionary without that method, we get the keys.

```
for val in gh_names:
    print(val)

brownsarahm
briannakathrynm1
jdion62
```

Libraries

To use libraries in python we import them

We will use pandas a lot in this class. It's the Python Data Analysis Library.

```
import pandas
```

Once we import we can use the functions, datatypes, and values a library provides by using a . after the name. In a notebook, pressing tab will show you the options.

We can also use an alias to give a library a nickname to make it easier to use, pd is the standard alias for pandas



note that import is a keyword and that in a Jupyter notebook, we can import anywhere and then the library can be used in any cell that is run after the import cell is run. It's good practice to put them at the top and make your notebook runnable in sequence, but Jupyter won't force you to.

import pandas as pd

We can read in from a local path or a url. Let's read in the course map page of our course website.

pd.read_html('https://rhodyprog4ds.github.io/BrownFall20/syllabus/course_map.html')



For example if you don't remember what kind of read functions there are in pandas, type pandas. read and then press tab to see options.

```
Unnamed: 0_level_0
                                                                    topics \
                                                       Unnamed: 1 level 1
                 week
0
                                                   [admin, python review]
                                              Loading data, Python review
                    3
                                                Exploratory Data Analysis
3
                    4
                                                            Data Cleaning
                                            Databases, Merging DataFrames
                       Modeling, Naive Bayes, classification performa...
                                         decision trees, cross validation
                    8
                                                               Regression
                    9
8
                                                               Clustering
9
                   10
                                                    SVM, parameter tuning
                                                    KNN, Model comparison
10
                   11
11
                   12
                                                            Text Analysis
                                                           Topic Modeling
12
                   13
13
                   14
                                                            Deep Learning
                             skills
                 Unnamed: 2_level_1
0
                            process
1
       [access, prepare, summarize]
2
             [summarize, visualize]
    [prepare, summarize, visualize]
3
     [access, construct, summarize]
4
         [classification, evaluate]
5
6
         [classification, evaluate]
             [regression, evaluate]
8
             [clustering, evaluate]
9
                  [optimize, tools]
10
                   [compare, tools]
11
                     [unstructured]
12
              [unstructured, tools]
13
                   [tools, compare]
   Unnamed: 0 level 0
                                                                     skill \
              keyword
                                                       Unnamed: 1_level_1
0
               python
                                                    pythonic code writing
              process
1
                                       describe data science as a process
2
                                          access data in multiple formats
               access
3
            construct
                                 construct datasets from multiple sources
4
            summarize
                                              Summarize and describe data
5
            visualize
                                                           Visualize data
                                                prepare data for analysis
6
              prepare
                                                     Apply classification
       classification
8
           regression
                                                         Apply Regression
9
           clustering
                                                                Clustering
                                               Evaluate model performance
10
             evaluate
                                                Optimize model parameters
11
             optimize
                                                           compare models
12
              compare
13
         unstructured
                                                  model unstructured data
14
                tools use industry standard data science tools and w...
                                               Level 1
                                    Unnamed: 2_level_1
0
    python code that mostly runs, occasional pep8 ...
            Identify basic components of data science
    load data from at least one format; identify t...
3
    identify what should happen to merge datasets ...
    Describe the shape and structure of a dataset ...
5
    identify plot types, generate basic plots from...
    identify if data is or is not ready for analys...
    identify and describe what classification is, ...
    identify what data that can be used for regres...
8
9
                          describe what clustering is
    Explain basic performance metrics for differen...
10
11
    Identify when model parameters need to be opti...
                  Oualitatively compare model classes
12
13
    Identify options for representing text data an...
    Solve well strucutred problems with a single t...
                                               Level 2
                                   Unnamed: 3_level_1
    python code that reliably runs, frequent pep8 ...
1
    Describe and define each stage of the data sci...
    Load data for processing from the most common ...
                                    apply basic merges
    compute summary statndard statistics of a whol...
    generate multiple plot types with complete lab...
    apply data reshaping, cleaning, and filtering ...
    fit preselected classification model to a dataset
8
                     can fit linear regression models
                               apply basic clustering
    Apply basic model evaluation metrics to a held...
    Manually optimize basic model parameters such ...
11
    Compare model classes in specific terms and fi...
12
    Apply at least one representation to transform...
13
    Solve semi-strucutred, completely specified pr...
```

```
Level 3
                                   Unnamed: 4_level_1
0
    reliable, efficient, pythonic code that consis...
    Compare different ways that data science can o...
2
    access data from both common and uncommon form...
3
         merge data that is not automatically aligned
    Compute and interpret various summary statisti...
4
5
    generate complex plots with pandas and plottin...
    apply data reshaping, cleaning, and filtering ...
6
    fit and apply classification models and select...
8
             can fit and explain nonlinear regression
9
    apply multiple clustering techniques, and inte...
10
   Evaluate a model with multiple metrics and cro...
    Select optimal parameters based of mutiple qua...
11
   Evaluate tradeoffs between different model com...
12
   apply mulitple representations and compare and...
13
14
    Scope, choose an appropriate tool pipeline and...
   Unnamed: 0_level_0
                          A1
              keyword Unnamed: 1_level_1 Unnamed: 2_level_1
              python
0
1
              process
2
               access
            construct
                                        0
            summarize
5
            visualize
6
              prepare
       {\tt classification}
           regression
9
                                                           0
           clustering
                                                           0
10
             evaluate
11
             optimize
12
              compare
13
         {\tt unstructured}
                                                           0
14
                tools
                   А3
                                      Α4
   Unnamed: 3_level_1 Unnamed: 4_level_1 Unnamed: 5_level_1
0
                                        0
1
6
10
11
12
                                                           0
                   Α6
                                                          A8 \
   Unnamed: 6_level_1 Unnamed: 7_level_1 Unnamed: 8_level_1
                                        0
1
                                        0
                                                           0
3
6
                                                           0
8
10
                                                           0
11
                                                           0
12
13
14
                                                           A11 \
                   Α9
                                      A10
   Unnamed: 9_level_1 Unnamed: 10_level_1 Unnamed: 11_level_1
0
                    0
                                         0
                                         0
1
                                                             0
                                         0
3
                                         0
                                                             0
6
8
11
12
```

```
14
                    0
                                         1
                                                              1
                   A12
                                        A13
                                                  # Assignments
   Unnamed: 12_level_1 Unnamed: 13_level_1 Unnamed: 14_level_1
0
                                          0
1
                     0
                                          0
                                                               2
                                                               3
2
                     0
                                          0
3
                     0
                                          0
                                                              2
                     1
                                                              11
5
                                                              10
                     0
                                          0
6
                                                               3
                     0
                                          0
8
                     0
                                          0
                     0
                                          0
10
                     0
                                          0
                                                               2
11
                                          0
                     0
12
                     0
                                          1
                                                               2
13
                                                               2
14
                                          1
  Unnamed: 0_level_0
                                                                   Level 3 \
              keyword
                                                       Unnamed: 1_level_1
0
               python
                       reliable, efficient, pythonic code that consis...
              process Compare different ways that data science can o...
2
               access access data from both common and uncommon form...
3
            construct
                            merge data that is not automatically aligned
4
            summarize Compute and interpret various summary statisti...
            visualize generate complex plots with pandas and plottin...
6
              prepare apply data reshaping, cleaning, and filtering ...
       classification fit and apply classification models and select...
8
           regression
                                can fit and explain nonlinear regression
9
           clustering apply multiple clustering techniques, and inte...
10
                       Evaluate a model with multiple metrics and cro...
             evaluate
11
             optimize Select optimal parameters based of mutiple qua...
              compare Evaluate tradeoffs between different model com...
12
13
         unstructured
                       apply mulitple representations and compare and...
14
                tools Scope, choose an appropriate tool pipeline and...
  Unnamed: 2 level 1 Unnamed: 3 level 1 Unnamed: 4 level 1 Unnamed: 5 level 1
0
                                                            0
                                                                               0
2
                                                            0
                                                                               0
                                                            0
                                                                               0
3
                                        1
4
                                        1
                                                            0
                                                                               0
5
                                        1
                                                            0
                                                                               0
                                                                               0
                                                                               0
8
                                        1
                                                            1
9
10
                                                                               0
11
                                        0
12
                    0
                                                            1
                                        0
13
                    0
                                                            1
                                                                               1
14
                    0
                                                                               1 1
```

This makes a list of pandas. DataFrame objects. We can check that with the following

▲ Warning

This cell was added after class, but the explanation was given in class

```
type(pd.read_html('https://rhodyprog4ds.github.io/BrownFall20/syllabus/course_map.html'
))
list
```

To work with it though, we should save to a variable, then we can index into that list.

```
df_list =
pd.read_html('https://rhodyprog4ds.github.io/BrownFall20/syllabus/course_map.html')
df_list[0]
```

	Unnamed: 0_level_0	topics		skills
	week		Unnamed: 1_level_1	Unnamed: 2_level_1
0	1		[admin, python review]	process
1	2		Loading data, Python review	[access, prepare, summarize]
2	3		Exploratory Data Analysis	[summarize, visualize]
3	4		Data Cleaning	[prepare, summarize, visualize]
4	5		Databases, Merging DataFrames	[access, construct, summarize]
5	6		$\label{eq:ModelingModelingModelingModelingModelingModelingModelingModel} \begin{picture}(100,00) \put(0.00) \put($	[classification, evaluate]
6	7		decision trees, cross validation	[classification, evaluate]
7	8		Regression	[regression, evaluate]
8	9		Clustering	[clustering, evaluate]
9	10		SVM, parameter tuning	[optimize, tools]
10	11		KNN, Model comparison	[compare, tools]
11	12		Text Analysis	[unstructured]
12	13		Topic Modeling	[unstructured, tools]
13	14		Deep Learning	[tools, compare]

When you display DataFrames in jupyter, they get nice formatting.

Review & Further Reading

- strings are iterable, more specifically sequences
- for loops, looping techniques and list comprehensions
- <u>dictionaries</u>
- imported <u>pandas</u> and <u>read data from a website</u>

If you've made it this far, <u>let me know</u> how you found these notes.

Class 4: Pandas

Today we will:

Remember, Programming is a Practice

- if you're curious about something try it
- you don't need me to give you answers about how code works, the interpreter will tell you
- if you don't remember details, remember you can get help from Jupyter

with a ? after the function name withouth ()

print?

or using the tab key inside the () for a function

print()

or from the core python, with the ${\tt help}$ fucntion

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.
```

Data in Pandas

We can import pandas again as before

```
import pandas as pd
```

and we can read in data.

```
pd.read_csv('https://raw.githubusercontent.com/brownsarahm/python-socialsci-
files/master/data/SAFI_clean.csv')
```

	key_ID	village	interview_date	no_membrs	years_liv	respondent_wall_type	rooms
0	1	God	2016-11- 17T00:00:00Z	3	4	muddaub	1
1	1	God	2016-11- 17T00:00:00Z	7	9	muddaub	1
2	3	God	2016-11- 17T00:00:00Z	10	15	burntbricks	1
3	4	God	2016-11- 17T00:00:00Z	7	6	burntbricks	1

to be able to use this, we need to save it to a variable.

```
safi\_df = pd.read\_csv('https://raw.githubusercontent.com/brownsarahm/python-socialsci-files/master/data/SAFI\_clean.csv')
```

This is an excerpt from the <u>SAFI dataset</u>.

Another important thing to do is to check datatypes, this is how we know what things we can do with a variable.

```
type(safi_df)

pandas.core.frame.DataFrame
```

An important thing to check is the size of the dataset.

```
safi_df.shape (131, 14)
```

Recall that you can also tab complete

```
safi_df.shape (131, 14)
```



we can also see the size when we print the whole dataset as in the first time we read the data in. safi_df.head()

	key_ID	village	interview_date	no_membrs	years_liv	respondent_wall_type	rooms	mem
0	1	God	2016-11- 17T00:00:00Z	3	4	muddaub	1	
1	1	God	2016-11- 17T00:00:00Z	7	9	muddaub	1	
2	3	God	2016-11- 17T00:00:00Z	10	15	burntbricks	1	
3	4	God	2016-11- 17T00:00:00Z	7	6	burntbricks	1	
4	5	God	2016-11- 17T00:00:00Z	7	40	burntbricks	1	

We can call this function with a value to change how many rows are returned

safi_df.head(3)

	key_ID	village	interview_date	no_membrs	years_liv	respondent_wall_type	rooms	men
0	1	God	2016-11- 17T00:00:00Z	3	4	muddaub	1	
1	1	God	2016-11- 17T00:00:00Z	7	9	muddaub	1	
2	3	God	2016-11- 17T00:00:00Z	10	15	burntbricks	1	

To know how this works, we can view the documentation for the function

help(safi_df.head)



▲ Warning

this was changed from using ? for help in class so that the help is desplayed in the rendered website, but the pop up was fine in real time

```
Help on method head in module pandas.core.generic:
head(n: int = 5) -> ~FrameOrSeries method of pandas.core.frame.DataFrame instance
   Return the first `n` rows.
   This function returns the first `n` rows for the object based
   on position. It is useful for quickly testing if your object
   has the right type of data in it.
   For negative values of `n`, this function returns all rows except the last `n` rows, equivalent to ``df[:-n]``.
   Parameters
   n : int, default 5
       Number of rows to select.
   Returns
   same type as caller
       The first `n` rows of the caller object.
   See Also
   DataFrame.tail: Returns the last `n` rows.
   Examples
   >>> df
         animal
   0 alligator
            hee
   1
   2
         falcon
   3
           lion
         monkey
         parrot
   6
          shark
          whale
   8
          zebra
   Viewing the first 5 lines
   >>> df.head()
         animal
   0
     alligator
            bee
         falcon
   2
   3
           lion
         monkey
   Viewing the first `n` lines (three in this case)
   >>> df.head(3)
         animal
   0 alligator
            bee
         falcon
   For negative values of `n`
   >>> df.head(-3)
         animal
   0 alligator
            bee
         falcon
   2
   3
           lion
   4
         monkey
         parrot
```

Since it says n = 5 we know that the default value of the parameter n is 5. When a function has a default value, we can call the function without a value.

To view the last few lines, we use tail

```
safi_df.tail()
```

	key_ID	village	interview_date	no_membrs	years_liv	respondent_wall_type	rooms
126	126	Ruaca	2017-05- 18T00:00:00Z	3	7	burntbricks	1
127	193	Ruaca	2017-06- 04T00:00:00Z	7	10	cement	3
128	194	Ruaca	2017-06- 04T00:00:00Z	4	5	muddaub	1
129	199	Chirodzo	2017-06- 04T00:00:00Z	7	17	burntbricks	2
130	200	Chirodzo	2017-06- 04T00:00:00Z	8	20	burntbricks	2

We can also get an Index for the columns of the DataFrame.

an Index variable is iterable so we can index into it

1 Try it Yourself

How would you view the name of the 3rd column?

First the correct answer:

```
'interview_date'
```

Now some misconceptions:

```
safi_df['interview_date']
```

```
2016-11-17T00:00:00Z
      2016-11-17T00:00:00Z
      2016-11-17T00:00:00Z
2
3
      2016-11-17T00:00:00Z
      2016-11-17T00:00:00Z
4
      2017-05-18T00:00:00Z
126
127
      2017-06-04T00:00:00Z
      2017-06-04T00:00:00Z
128
      2017-06-04T00:00:00Z
129
      2017-06-04T00:00:00Z
Name: interview_date, Length: 131, dtype: object
```

Indexing with the column name) will return the values in the column

```
safi_df.columns(2)
```

<u>Index</u> is another data type that is defined by pandas.

```
TypeError Traceback (most recent call last)
<ipython-input-17-bd02c7e8a4a6> in <module>
----> 1 safi_df.columns(2)

TypeError: 'Index' object is not callable
```

Using () returns an error, because columns is an *attribute* which is referenced as is with no (). We get a type error because functions in python are objects of type callable and properties are values not functions.

```
TypeError Traceback (most recent call last)
<ipython-input-18-40e277f3074e> in <module>
----> 1 pd.DataFrame.columns[2]
TypeError: 'pandas._libs.properties.AxisProperty' object is not subscriptable
```

This doesn't work because columns is an attribute of an object of type pandas.DataFrame and pd.DataFrame.columns is not an object.

We can see what the type of pd.DataFrame is with the type function.

```
type(pd.DataFrame)
type
```

Knowing about types is helpful for the individual columns of a dataset as well.

```
safi_df.dtypes
 key_ID
                         int64
 village
                         object
 interview_date
                         object
 no_membrs
                          int64
 years_liv
                          int64
 respondent_wall_type
                         object
                         int64
 rooms
 memb_assoc
                         object
 affect_conflicts
                         object
                         int64
 liv_count
 items owned
                         object
 no_meals
                         int64
 months_lack_food
                         object
 instanceID
                         object
 dtype: object
```

Note that it uses int64 and object as the types.

```
key_ID village interview_date no_membrs years_liv respondent_wall_type rooms mem
```

0	1	God	2016-11- 17T00:00:00Z	3	4	muddaub	1
1	1	God	2016-11- 17T00:00:00Z	7	9	muddaub	1

We might want to look at what villages were included in the data.

```
pd.unique(safi_df['village'])
array(['God', 'Chirodzo', 'Ruaca'], dtype=object)
```

We can also get count of the number of of each value

```
safi_df['village'].value_counts()
```

```
Ruaca 49
God 43
Chirodzo 39
Name: village, dtype: int64
```

1 Try it Yourself!

how many surveyed farms have all type mauddaub?

Review and Further reading

- reading data with pandas
- Python built in functions and in particular the type function
- Pandas DataFrames
- <u>value_counts</u>

If you've made it this far, <u>let me know</u> how you found these notes.

Class 5: Accessing Data, continued

Today's agenda:

- warm up/ review
- announcements
- working with dataframes
- the power of functions as objects
- (maybe) exploratory data analysis

1 Try it out ->

Read the tables off of the syllabus course map page with read_html and make a list of the shapes of all of the tables on the page. Save the output to a variable and paste the *value* of that variable as your answer to the question.

```
import pandas as pd
[df.shape for df in
pd.read_html('https://rhodyprog4ds.github.io/BrownFall20/syllabus/course_map.html')]
```

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

Announcements

- annotated notes are up
- beginning portfolio prompts and instructions are up
- Assignment due Sunday,
- office hours will remain Fridays
- TA office hours posted.

More Pandas

We'll go back to the SAFI dataset from Wednesday.

```
safi_df = pd.read_csv('https://raw.githubusercontent.com/brownsarahm/python-socialsci-
files/master/data/SAFI_clean.csv')
```

We downloaded the data into memory, but we can also write it to disk.

```
safi_df.to_csv('safi_clean.csv')
```

It will go to the same folder as the notebook, but we can also use a relative path. If we make a data folder in the folder where we've saved the notebook, we can write the file there instead.

%%bash mkdir data

safi_df.to_csv('data/safi_clean.csv')

Now we can read it in using the same path

```
safi_df2= pd.read_csv('data/safi_clean.csv')
```

Note that now it has an extra column

safi_df2.head(2)

Unnamed: key_ID village interview_date no_membrs years_liv respondent_wall_type ı 2016-11-0 0 God 3 4 muddaub 17T00:00:00Z 2016-11-1 God 7 muddaub 17T00:00:00Z

safi_df.head(2)

key_ID village interview_date no_membrs years_liv respondent_wall_type rooms mem 2016-11-0 1 God 3 muddaub 1 17T00:00:00Z 2016-11-1 1 God muddaub 1 17T00:00:00Z

We can prevent this by writing it out with the index parameter set to False

```
safi_df.to_csv('data/safi_clean.csv',index=False)
```

Now when we read it in, there's no extra column.

```
safi_df3 = pd.read_csv('data/safi_clean.csv')
safi_df3.head(3)
```

Tip

In class we used the Jupyter GUI to create a new folder. You could also use your computer's default file management tool (Windows Explorer, Mac Finder, etc). Here, since the notebooks have to run completely automatically for this website, we use a ipython bash magic cell to make the folder. Jupyter notebooks use an ipython kernel.

False must be with a capital letter to be a boolean variable in python, as with True. You'll know if you did it right in your jupyter notebook, if the word terms bold and green.

men	rooms	respondent_wall_type	years_liv	no_membrs	interview_date	village	key_ID	
	1	muddaub	4	3	2016-11- 17T00:00:00Z	God	1	0
	1	muddaub	9	7	2016-11- 17T00:00:00Z	God	1	1
	1	burntbricks	15	10	2016-11- 17T00:00:00Z	God	3	2

Recall, we indexed a column with the name in square brackets

```
safi_df['village']
              God
 1
             God
 2
              God
              {\sf God}
              God
 126
           Ruaca
 127
           Ruaca
 128
           Ruaca
        Chirodzo
 129
 130
        Chirodzo
 Name: village, Length: 131, dtype: object
```

To index rows, we can use loc

```
safi_df.loc[3]
```

key_ID	4
/illage	God
interview_date	2016-11-17T00:00:00Z
no membrs	7
_ years liv	6
respondent_wall_type	burntbricks
rooms	1
nemb_assoc	NaN
affect conflicts	NaN
liv count	2
items owned	<pre>bicycle;radio;cow plough;solar panel;mobile phone</pre>
no meals	2
months lack food	Sept;Oct;Nov;Dec
instanceID	uuid:148d1105-778a-4755-aa71-281eadd4a973
Name: 3, dtype: object	

To select a range, use:

safi _.

	key_ID	village	interview_date	no_membrs	years_liv	respondent_wall_type	rooms	men
3	4	God	2016-11- 17T00:00:00Z	7	6	burntbricks	1	
4	5	God	2016-11- 17T00:00:00Z	7	40	burntbricks	1	
5	6	God	2016-11- 17T00:00:00Z	3	3	muddaub	1	

You only have to have a number on one side of the colon, it will go from the beginnig up to that number like this:

safi	df	100	r • 41
301I	uı.	LUC	

	key_ID	village	interview_date	no_membrs	years_liv	respondent_wall_type	rooms	men
0	1	God	2016-11- 17T00:00:00Z	3	4	muddaub	1	
1	1	God	2016-11- 17T00:00:00Z	7	9	muddaub	1	
2	3	God	2016-11- 17T00:00:00Z	10	15	burntbricks	1	
3	4	God	2016-11- 17T00:00:00Z	7	6	burntbricks	1	
4	5	God	2016-11- 17T00:00:00Z	7	40	burntbricks	1	

With two : : we can also set an increment

safi_df.loc[::5]

	key_ID	village	interview_date	no_membrs	years_liv	respondent_wall_type	rooms
0	1	God	2016-11- 17T00:00:00Z	3	4	muddaub	1
5	6	God	2016-11- 17T00:00:00Z	3	3	muddaub	1
10	11	God	2016-11- 21T00:00:00Z	6	20	sunbricks	1
15	16	God	2016-11- 24T00:00:00Z	6	47	muddaub	1
20	21	God	2016-11- 21T00:00:00Z	8	20	burntbricks	1
25	26	Ruaca	2016-11- 21T00:00:00Z	3	20	burntbricks	2
30	31	Ruaca	2016-11- 21T00:00:00Z	3	2	muddaub	1
35	36	Chirodzo	2016-11- 17T00:00:00Z	6	23	sunbricks	1
40	41	God	2016-11- 17T00:00:00Z	7	22	muddaub	1
45	46	Chirodzo	2016-11- 17T00:00:00Z	10	42	burntbricks	2
50	51	Chirodzo	2016-11- 16T00:00:00Z	5	30	muddaub	1
55	56	Chirodzo	2016-11- 16T00:00:00Z	12	23	burntbricks	2
60	61	Chirodzo	2016-11- 16T00:00:00Z	10	14	muddaub	1
65	66	Chirodzo	2016-11- 16T00:00:00Z	10	37	burntbricks	3
70	71	Ruaca	2016-11- 18T00:00:00Z	6	14	burntbricks	1
75	155	God	2016-11- 24T00:00:00Z	4	4	burntbricks	1

	key_ID	village	interview_date	no_membrs	years_liv	respondent_wall_type	rooms
80	182	God	2016-11- 25T00:00:00Z	7	21	muddaub	3
85	197	God	2016-11- 28T00:00:00Z	5	19	burntbricks	2
90	73	Ruaca	2017-04- 26T00:00:00Z	7	9	burntbricks	2
95	101	God	2017-04- 27T00:00:00Z	3	4	muddaub	1
100	104	Ruaca	2017-04- 28T00:00:00Z	14	52	sunbricks	1
105	113	Ruaca	2017-05- 03T00:00:00Z	11	26	burntbricks	3
110	108	God	2017-05- 11T00:00:00Z	15	22	burntbricks	2
115	150	Ruaca	2017-05- 18T00:00:00Z	7	8	muddaub	1
120	167	Ruaca	2017-06- 03T00:00:00Z	8	24	muddaub	1
125	192	Chirodzo	2017-06- 03T00:00:00Z	9	20	burntbricks	1
130	200	Chirodzo	2017-06- 04T00:00:00Z	8	20	burntbricks	2

These can be combined to index a subset at an increment.

We can index columns in two ways, as we did on Wednesday

```
safi_df['village'].head(2)

0  God
1  God
Name: village, dtype: object
```

Or using a $\mbox{.}$

```
safi_df.village.head(2)

0 God
1 God
Name: village, dtype: object
```

We can select multiple columns, using a list of column names. We can define the list inline.

```
safi_df[['village','no_membrs','years_liv']].head(2)
```

village no_membrs years_liv

0	God	3	4
1	God	7	9

or in a separate variable

```
columns_of_interest = ['village','no_membrs','years_liv']
safi_df[columns_of_interest].head(2)
```

village no_membrs years_liv

0	God	3	4
1	God	7	9

Functions are objects

```
syllabus_df_list =
pd.read_html('https://rhodyprog4ds.github.io/BrownFall20/syllabus/course_map.html')
```

And we can put them in a dictionary. Lambda functions are special functions defined in a single line.

```
greetingl = lambda name: 'hello ' + name
greetingl('sarah')
```

```
'hello sarah'
```

is the same as

```
def greetingf(name):
    return 'hello ' + name
greetingf('sarah')
```

```
'hello sarah'
```

So, we can define a function in a dictionary like this:

The len function works on all iterables

```
for df in syllabus_df_list:
    num_row = len(df)
    view_rows[num_row%2](df)
```

🥊 Tip

this is how to do the equivalent of a switch or case in other languages in python

```
Unnamed: 0_level_0
                                              topics
                                 Unnamed: 1_level_1
                week
0
                   1
                             [admin, python review]
                        Loading data, Python review
2
                   3
                          Exploratory Data Analysis
                                      Data Cleaning
3
                   4
4
                      Databases, Merging DataFrames
                            skills
                Unnamed: 2_level_1
0
                           process
1
      [access, prepare, summarize]
           [summarize, visualize]
3
   [prepare, summarize, visualize]
    [access, construct, summarize]
  Unnamed: 0_level_0
                                                                    skill
             keyword
                                                       Unnamed: 1_level_1
                                               Evaluate model performance
             evaluate
                                                Optimize model parameters
11
             optimize
                                                           compare models
12
             compare
13
         unstructured
                                                  model unstructured data
14
                tools use industry standard data science tools and w...
                                               Level 1 \
                                   Unnamed: 2_level_1
10 Explain basic performance metrics for differen...
11
    Identify when model parameters need to be opti...
12
                 Qualitatively compare model classes
    Identify options for representing text data an...
13
    Solve well strucutred problems with a single t...
                                               Level 2
                                   Unnamed: 3_level_1
10 Apply basic model evaluation metrics to a held...
11
   Manually optimize basic model parameters such ...
12 Compare model classes in specific terms and fi...
   Apply at least one representation to transform...
   Solve semi-strucutred, completely specified pr...
                                               Level 3
                                   Unnamed: 4_level_1
    Evaluate a model with multiple metrics and cro...
   Select optimal parameters based of mutiple qua...
12 Evaluate tradeoffs between different model com...
   apply mulitple representations and compare and...
   Scope, choose an appropriate tool pipeline and...
   Unnamed: 0_level_0
                                      A1
                                                          Α2
             keyword Unnamed: 1_level_1 Unnamed: 2_level_1
10
             evaluate
                                       0
                                                           0
11
             optimize
                                       0
                                                           0
12
             compare
13
         unstructured
                                        0
                                                           0
                                       0
                                                           0
14
                tools
   Unnamed: 3_level_1
                      Unnamed: 4_level_1 Unnamed: 5_level_1
10
                                       0
                    0
11
                    0
                                       0
                                                           0
12
                    0
                                        0
                                                           0
13
                    0
                                        0
                                                           0
                                                           0
14
                    0
                                       0
                   A6
                                      Α7
                                                          A8
   Unnamed: 6_level_1
                      Unnamed: 7_level_1
                                         Unnamed: 8_level_1
10
                                        0
11
                    0
                                       0
                                                           0
                                       0
                                                           0
12
                    0
13
                    0
                                       0
                                                           0
                    0
                                        0
                                                           0
                                      A10
  Unnamed: 9_level_1 Unnamed: 10_level_1 Unnamed: 11_level_1
10
                    0
                                        1
11
12
                    0
                                        0
                                                             1
13
                    0
                                        0
                                                             0
14
                    0
                                       A13
                                                  # Assignments
  Unnamed: 12_level_1 Unnamed: 13_level_1 Unnamed: 14_level_1
10
                                         0
                     0
                                                              2
11
                     0
                                         0
12
                     0
                                         1
                                                              2
13
                                          1
14
                                         1
  Unnamed: 0_level_0
                                                                  Level 3 \
                                                       Unnamed: 1_level_1
```

11 optimize 12 compare 13 unstructured	Evaluate a model with mult Select optimal parameters Evaluate tradeoffs between apply mulitple representat Scope, choose an appropria	based of mutiple n different model tions and compare	qua com and
P1	P2	Р3	P4
Unnamed: 2_level_1	Unnamed: 3_level_1 Unnamed:	4_level_1 Unname	ed: 5_level_1
10 0	1	1	0
11 0	Θ	1	1
12 0	0	1	1
13 0	0	1	1
14 0	0	1	1

The beginning of Exploratory Data Analysis

Pandas will give us descriptive statistics

safi_df.describe()			
--------------------	--	--	--

	key_ID	no_membrs	years_liv	rooms	liv_count	no_meals
count	131.000000	131.00000	131.000000	131.000000	131.000000	131.000000
mean	85.473282	7.19084	23.053435	1.740458	2.366412	2.603053
std	63.151628	3.17227	16.913041	1.092547	1.082775	0.491143
min	1.000000	2.00000	1.000000	1.000000	1.000000	2.000000
25%	32.500000	5.00000	12.000000	1.000000	1.000000	2.000000
50%	66.000000	7.00000	20.000000	1.000000	2.000000	3.000000
75%	138.000000	9.00000	27.500000	2.000000	3.000000	3.000000
max	202.000000	19.00000	96.000000	8.000000	5.000000	3.000000

 $The \ statistics \ of \ the \ key_ID \ column \ don't \ make \ a \ lot \ of \ sense. \ We \ can \ avoid \ that \ by \ making \ it \ the \ index$

```
safi_df.head()
```

	key_ID	village	interview_date	no_membrs	years_liv	respondent_wall_type	rooms	men
0	1	God	2016-11- 17T00:00:00Z	3	4	muddaub	1	
1	1	God	2016-11- 17T00:00:00Z	7	9	muddaub	1	
2	3	God	2016-11- 17T00:00:00Z	10	15	burntbricks	1	
3	4	God	2016-11- 17T00:00:00Z	7	6	burntbricks	1	
4	5	God	2016-11- 17T00:00:00Z	7	40	burntbricks	1	

the inplace parameter of a pandas functions applies the operation to the DataFrame in memory, but then the function returns nothing, but if we display after that, we see that now the key_ID column is now the index.

```
safi_df.set_index('key_ID',inplace=True)
safi_df.head(2)
```

village interview_date no_membrs years_liv respondent_wall_type rooms memb_a

muddaub

key_ID

1	God	2016-11- 17T00:00:00Z	3	4	muddaub	1

and if we describe again, we see it doesn't compute on that column

God

2016-11-

17T00:00:00Z

safi_df.describe()

	no_membrs	years_liv	rooms	liv_count	no_meals
count	131.00000	131.000000	131.000000	131.000000	131.000000
mean	7.19084	23.053435	1.740458	2.366412	2.603053
std	3.17227	16.913041	1.092547	1.082775	0.491143
min	2.00000	1.000000	1.000000	1.000000	2.000000
25%	5.00000	12.000000	1.000000	1.000000	2.000000
50%	7.00000	20.000000	1.000000	2.000000	3.000000
75%	9.00000	27.500000	2.000000	3.000000	3.000000
max	19.00000	96.000000	8.000000	5.000000	3.000000

We can also call any of those on one column or one statistic.

```
safi_df['rooms'].mean()
1.7404580152671756
```

Pandas also has some built in plotting functions.

```
safi_df.plot.scatter('no_membrs', 'rooms')

<a href="mailto:scatter">
<a href="mailto:scatter">safi_df.plot.scatter('no_membrs', 'rooms')</a>

<a href="mailto:scatter">
<a href="mailto:scatter">safi_df.plot.scatter</a>
```

After Class Questions



More Practice

These additional questions are for if you want more practice with things we've done this week, before class next week.



Further Reading

If you've made it this far, <u>let me know</u> how you found these notes.

Class 6: Exploratory Data Analysis

Warmup

```
topics = ['what is data science', 'jupyter', 'conditional','functions', 'lists',
    'dictionaries','pandas' ]

['what is data science',
    'jupyter',
    'conditional',
    'functions',
    'lists',
    'dictionaries',
    'pandas']

What happens when we index with -1?

topics[-1]

'pandas'
```

We get the last value.

Recall last class we used: to index DataFrames with.loc, we can do that with lists too:

```
topics[-2:]

['dictionaries', 'pandas']
```

Announcements

- next assignment will go up today
- try practingin throughout the week
- Check Brightspace for TA office hours

Why Exploratory Data Analysis (EDA) before cleaning?

Typically a Data process looks like one of these figures:

Data Science Process OBTAIN 0 S Е Μ N Gather data from Clean data to formats Find significant patterns Construct models to Put the results into and trends using statistical methods predict and forecast relevant sources that machine good use understands Exploratory Data Analysis Cleaning Model Model Collection Building Deployment Data Engineers Data Analysts Machine Learning Engineers

Data cleaning, which some of you asked about on Friday, would typically, happen before EDA, but it's hard to know what good data looks like, until you're used to manipulating it. Also, it's hard to check if data is good (and your cleaning worked) without some EDA. So, I'm choosing to teach EDA first. We'll do data cleaning next, with these tools from EDA available as options for checking and determining what cleaning to do.

Data Scientists

EDA

import pandas as pd

and pull in the data.

```
safi_df = pd.read_csv(data_url)
safi_df.head(2)
```

	key_id	interview_date	quest_no	start	end	province	district
0	1	17 November 2016	1	2017-03- 23T09:49:57.000Z	2017-04- 02T17:29:08.000Z	Manica	Manica
1	2	17 November 2016	1	2017-04- 02T09:48:16.000Z	2017-04- 02T17:26:19.000Z	Manica	Manica

2 rows × 65 columns

Recall on Friday, we can use describe to see several common descriptive statistics

safi_df	safi_df.describe()									
	key_id	quest_no	years_farm	note2	no_membrs	members_count	year			
count	131.000000	131.000000	131.000000	0.0	131.00000	131.00000	131.000			
mean	66.000000	85.473282	15.832061	NaN	7.19084	7.19084	23.053			
std	37.960506	63.151628	10.903883	NaN	3.17227	3.17227	16.913			
min	1.000000	1.000000	1.000000	NaN	2.00000	2.00000	1.000			
25%	33.500000	32.500000	8.000000	NaN	5.00000	5.00000	12.000			
50%	66.000000	66.000000	15.000000	NaN	7.00000	7.00000	20.000			
75%	98.500000	138.000000	20.500000	NaN	9.00000	9.00000	27.500			

8 rows × 25 columns

max 131.000000 202.000000

Then we remember that this includes key_id as a variable tht we don't actually want to treat as a variable, it's an index. We can change that with set_index and we can do it in memory (without assignment) using the inplace keyword.

60.000000

19.00000

96.000

19.00000

```
safi_df.set_index('key_id',inplace=True)
safi_df.describe()
```

	quest_no	years_farm	note2	no_membrs	members_count	years_liv	respond
count	131.000000	131.000000	0.0	131.00000	131.00000	131.000000	
mean	85.473282	15.832061	NaN	7.19084	7.19084	23.053435	
std	63.151628	10.903883	NaN	3.17227	3.17227	16.913041	
min	1.000000	1.000000	NaN	2.00000	2.00000	1.000000	
25%	32.500000	8.000000	NaN	5.00000	5.00000	12.000000	
50%	66.000000	15.000000	NaN	7.00000	7.00000	20.000000	
75%	138.000000	20.500000	NaN	9.00000	9.00000	27.500000	
max	202.000000	60.000000	NaN	19.00000	19.00000	96.000000	
_							

8 rows × 24 columns

Try it yourself!

You can use settings on pd. read_csv to set the index when the data is read in instead of doing this after the fact

We can also use the descriptive statistics on a single variable.

Note

In class, we used %load http://drsmb.co/310 to pull in the url for the data. Load is an ipython (ther kernel thta Jupyter uses) magic function. It allows us to read in data from another place. I've set that short url to link to the download url for this hackmd as markdown(plain text). That way I can paste things and you can pull them directly into your notebook. We'll use it like a shared copy-paste.

```
safi_df['years_farm'].describe()
```

```
131.000000
count
          15.832061
mean
std
          10.903883
           1.000000
min
           8.000000
25%
          15.000000
50%
75%
          20.500000
max
          60.000000
Name: years_farm, dtype: float64
```

Note however that this is not well formatted, that's because it's a Series instead of a DataFrame

```
type(safi_df['years_farm'].describe())

pandas.core.series.Series
```

We can use ${\tt reset_index}()$ to make it back into a dataframe

```
safi_df['years_farm'].describe().reset_index()
```

	index	years_farm
0	count	131.000000
1	mean	15.832061
2	std	10.903883
3	min	1.000000
4	25%	8.000000
5	50%	15.000000
6	75%	20.500000
7	max	60.000000

And we can drop the added index:

```
safi_df['years_farm'].describe().reset_index().set_index('index')
```

years_farm index 131.000000 count mean 15.832061 10.903883 std min 1.000000 25% 8.000000 50% 15.000000 75% 20.500000 max 60.000000



See that we can chain operations together. This is a helpful feature, but to be used with care. <u>Pep8</u>, the style conventions for python, recommends no more than 79 characters per line. A <u>summary</u> and <u>another summary</u>

We can also use individual summary statistics on DataFrame or Series objects

```
safi_df['no_membrs'].max()
```

19

Split - Apply - Combine

A powerful tool in pandas (and data analysis in general) is the ability to apply functions to subsets of the data.



For example, we can get descriptive statistics per village

```
safi_df.groupby('village').describe()
```

	quest_no						years_farm			
	count	mean	std	min	25%	50%	75%	max	count	mean
village										
Chirodzo	39.0	62.487179	44.261705	8.0	44.5	55.0	64.5	200.0	39.0	17.2820
God	43.0	81.720930	77.863839	1.0	14.5	40.0	168.5	202.0	43.0	14.8837
Ruaca	49.0	107.061224	55.024013	23.0	72.0	113.0	152.0	194.0	49.0	15.5102
3 rows × 192 columns										

We can also rearrange this into a more usable format than this very wide format:

safi_df.groupby('village').describe().unstack().reset_index()

	level_0	level_1	village	0
0	quest_no	count	Chirodzo	39.000000
1	quest_no	count	God	43.000000
2	quest_no	count	Ruaca	49.000000
3	quest_no	mean	Chirodzo	62.487179
4	quest_no	mean	God	81.720930
571	gps_Accuracy	75%	God	13.500000
572	gps_Accuracy	75%	Ruaca	12.000000
573	gps_Accuracy	max	Chirodzo	39.000000
574	gps_Accuracy	max	God	30.000000
575	gps_Accuracy	max	Ruaca	2099.999000
576 rc	ows × 4 columns			

 $This, however, gives some funny variable \ names, to fix \ that, we first save it to a variable.$

Tip
one column of a
pd.DataFrame is a pd.Series

1 Note

In class, we used the load magic again here to get this url. Then split the cell, changed the second one to markdown with the code for the image

← Tip

To include an image in a notebook, use:

![alt text here]
(url/or/path/to/image
)

in a markdown cell

village_summary_df = safi_df.groupby('village').describe().unstack().reset_index()

Now we can use the rename function

 $\verb|help(village_summary_df.rename)|$



Rename is also one thing you might do in cleaning a dataset.

```
Help on method rename in module pandas.core.frame:
rename(mapper=None, index=None, columns=None, axis=None, copy=True, inplace=False,
level=None, errors='ignore') method of pandas.core.frame.DataFrame instance
    Alter axes labels.
    Function / dict values must be unique (1-to-1). Labels not contained in
    a dict / Series will be left as-is. Extra labels listed don't throw an
    See the :ref:`user guide <basics.rename>` for more.
    Parameters
    mapper : dict-like or function
        Dict-like or functions transformations to apply to that axis' values. Use either ``mapper`` and ``axis`` to specify the axis to target with ``mapper``, or ``index`` and
          `columns``
    index : dict-like or function
        Alternative to specifying axis (``mapper, axis=0`` is equivalent to ``index=mapper``).
    columns : dict-like or function
        Alternative to specifying axis (``mapper, axis=1``
    is equivalent to ``columns=mapper``).
axis : {0 or 'index', 1 or 'columns'}, default 0
        Axis to target with ``mapper``. Can be either the axis name
        ('index', 'columns') or number (0, 1). The default is 'index'.
    copy : bool, default True
        Also copy underlying data.
    inplace : bool, default False
        Whether to return a new DataFrame. If True then value of copy is
        ignored.
    level : int or level name, default None
        In case of a MultiIndex, only rename labels in the specified
        level.
    errors : {'ignore', 'raise'}, default 'ignore'
        If 'raise', raise a `KeyError` when a dict-like `mapper`, `index`,
        or `columns` contains labels that are not present in the \ensuremath{\mathsf{Index}}
        being transformed.
        If 'ignore', existing keys will be renamed and extra keys will be
        ignored.
    Returns
    DataFrame
        DataFrame with the renamed axis labels.
    Raises
    KeyError
        If any of the labels is not found in the selected axis and
        "errors='raise'"
    See Also
    DataFrame.rename_axis : Set the name of the axis.
    Examples
    ``DataFrame.rename`` supports two calling conventions
    * ``(index=index_mapper, columns=columns_mapper, ...)``
    * ``(mapper, axis={'index', 'columns'}, ...)
    We *highly* recommend using keyword arguments to clarify your
    intent.
    Rename columns using a mapping:
    >>> df = pd.DataFrame(\{"A": [1, 2, 3], "B": [4, 5, 6]\})
    >>> df.rename(columns={"A": "a", "B": "c"})
       а с
    0 1 4
    1 2 5
    2 3 6
    Rename index using a mapping:
    >>> df.rename(index={0: "x", 1: "y", 2: "z"})
      A B
    x 1 4
      2 5
    z 3 6
    Cast index labels to a different type:
```

```
>>> df.index
RangeIndex(start=0, stop=3, step=1)
>>> df.rename(index=str).index
Index(['0', '1', '2'], dtype='object')
>>> df.rename(columns={"A": "a", "B": "b", "C": "c"}, errors="raise")
Traceback (most recent call last):
KeyError: ['C'] not found in axis
Using axis-style parameters
>>> df.rename(str.lower, axis='columns')
   a b
0
  1 4
  2
     5
2 3 6
>>> df.rename({1: 2, 2: 4}, axis='index')
   A B
  1 4
2 2 5
4
   3
      6
```

Rename takes a dict-like or function that maps from what's there to what to change it to. We can either provide the mapper and an axis or pass the mapper as the columns parameter. The columns parameter makes it more human readable, and explicit what we're doing.

Now we have a much better summary table. This is in what's called tidy format. Each colum

```
village_summary_df.head()
```

	variable	statistic	village	value
0	quest_no	count	Chirodzo	39.000000
1	quest_no	count	God	43.000000
2	quest_no	count	Ruaca	49.000000
3	quest_no	mean	Chirodzo	62.487179
4	quest_no	mean	God	81.720930

How could we use this to compute the average across villages for each statistic of each variable?

1 Try it yourself!

What would it look like to do this from the first result of describe() on the groupby, instead of with the unstack and reset_index?

We do need to groupby 'statistic' and use mean, but that's not enough:

```
village_summary_df.groupby('statistic').mean()
```

value statistic 25% 44.807522 50% 48.428030 75% 54.062308 30.388889 count 105.468983 max 49.868824 mean 13.756913 min std 24.588967

inplace again. See questions at the bottom for more on what it does

This averages across all of the variables, too. So instead we need to groupby two variables.

```
village_summary_df.groupby(['statistic','variable']).mean()
```

```
value
statistic
                           variable
   25%
             buildings_in_compound
                                       1.000000
                      gps_Accuracy
                                       8.000000
                       gps_Altitude 691.833333
                       gps_Latitude
                                     -19.112221
                     gps_Longitude
                                      33.480944
    std respondent_wall_type_other
                                            NaN
                                       1.082409
                            rooms
                        years_farm
                                      10.832121
                                      16.549343
                           years liv
                                       1.074600
                   yes_group_count
```

192 rows × 1 columns

```
safi_df.columns
```

```
Index(['interview_date', 'quest_no', 'start', 'end', 'province', 'district',
    'ward', 'village', 'years_farm', 'agr_assoc', 'note2', 'no_membrs',
    'members_count', 'remittance_money', 'years_liv', 'parents_liv',
    'sp_parents_liv', 'grand_liv', 'sp_grand_liv', 'respondent_roof_type',
    'respondent_wall_type', 'respondent_wall_type_other',
    'respondent_floor_type', 'window_type', 'buildings_in_compound',
    'rooms', 'other_buildings', 'no_plots', 'plots_count', 'water_use',
    'no_group_count', 'yes_group_count', 'no_enough_water',
    'months_no_water', 'period_use', 'exper_other', 'other_meth',
    'res_change', 'memb_assoc', 'resp_assoc', 'fees_water',
    'affect_conflicts', 'note', 'need_money', 'money_source',
    'money_source_other', 'crops_contr', 'emply_lab', 'du_labour',
    'liv_owned', 'liv_owned_other', 'liv_count', 'poultry',
    'du_look_aftr_cows', 'items_owned', 'items_owned_other', 'no_meals',
    'months_lack_food', 'no_food_mitigation', 'gps_Latitude',
    'gps_Longitude', 'gps_Altitude', 'gps_Accuracy', 'instanceID'],
    dtype='object')
```

What is the most common combination of respondent_wall_type and respondent_floor_type?

```
safi_df.groupby(['respondent_wall_type','respondent_floor_type'])
['instanceID'].count().reset_index()
```

	respondent_wall_type	respondent_floor_type	instanceID
0	burntbricks	cement	30
1	burntbricks	earth	37
2	cement	cement	1
3	muddaub	cement	3
4	muddaub	earth	43
5	sunbricks	cement	4
6	sunbricks	earth	13

We can read this table to see the result. Or we might want it in a different way

```
safi_df.groupby(['respondent_wall_type','respondent_floor_type'])
['instanceID'].count().unstack()
```

respondent_floor_type	cement	earth
respondent_wall_type		
burntbricks	30.0	37.0
cement	1.0	NaN
muddaub	3.0	43.0
sunbricks	4.0	13.0

Questions After Class

What does the inplace parameter do?

We used inplace for the first time today right after we read in the data. Let's make a new DataFrame and compare what happens with and without.





I'm surprised I've remembered it repeatedly in class. I usually forget inplace don't get the output I want and then go back and add it. That's a normal part of programming.

	interview_date	quest_no	start	end	province	district	ν
key_id							
1	17 November 2016	1	2017-03- 23T09:49:57.000Z	2017-04- 02T17:29:08.000Z	Manica	Manica	E
2	17 November 2016	1	2017-04- 02T09:48:16.000Z	2017-04- 02T17:26:19.000Z	Manica	Manica	E
3	17 November 2016	3	2017-04- 02T14:35:26.000Z	2017-04- 02T17:26:53.000Z	Manica	Manica	E
4	17 November 2016	4	2017-04- 02T14:55:18.000Z	2017-04- 02T17:27:16.000Z	Manica	Manica	E
5	17 November 2016	5	2017-04- 02T15:10:35.000Z	2017-04- 02T17:27:35.000Z	Manica	Manica	E

 $5 \text{ rows} \times 64 \text{ columns}$

Without inplace, set_index returns a dataframe, with the index changed, with inplace it returns None. But, without inplace, we don't see the desired effect when we do our next step, but with inplace the index is changed.

Without the inplace, to save the changes you need to use an assignment. The following is equivalent to using inplace.

```
safi_df_noinplace= safi_df_noinplace.set_index('key_id'))
safi_df_noinplace.head()

File "<ipython-input-30-la84c65392b3>", line 1
    safi_df_noinplace= safi_df_noinplace.set_index('key_id'))

SyntaxError: invalid syntax
```

Why is a for loop slower than a pandas operation?

TL; DR: for this example, it's actually possible to relatively easily write a faster loop, but for other operations, it's unlikely.

Basically, the advantage of using the library functions is that someone has already put a lot of though into the optimal way to implement things and it leverages core features of the data structure. Here is a <u>blog post</u> about iterating and applying functions in pandas. It covers topics we haven't yet seen, but will.

The pandas implementation is also fewer lines and can scale for large datasets eaisly.

To test, we can do an experiment. Let's take the last thing we covered in class today, finding the most common combination of floor and wall types.

First we'll look at two solutions and verify that they're the same.

First, how to do it with pandas

```
wall_floor_df = safi_df.groupby(['respondent_wall_type','respondent_floor_type'])
['instanceID'].count()
wall_floor_df
```

```
respondent_wall_type respondent_floor_type
burntbricks
                      cement
                                               30
                                               37
                      earth
cement
                      cement
                                                1
muddaub
                      cement
                                                3
                                               43
sunbricks
                      cement
                                                4
                                               13
                      earth
Name: instanceID, dtype: int64
```

We can read the answer off of that table, but let's get it programmatically:



For example using modin that allows you to change 1 line of code (import pandas as pd to import modin.pandas as pd) to leverage parallelized computation. Base python loops won't do that.

```
wall_floor_df.idxmax()
    ('muddaub', 'earth')
```

Next, how to do it with a for loop (if you have a better for loop, make a PR so share it).

```
wall_floor_dict = {}

for wall,floor in
zip(safi_df['respondent_wall_type'],safi_df['respondent_floor_type']):
    wf_key = '_'.join([wall,floor])
    if wf_key in wall_floor_dict.keys():
        wall_floor_dict[wf_key] +=1
    else:
        wall_floor_dict[wf_key] =1

wall_floor_dict
```

```
{'muddaub_earth': 43,
   'burntbricks_cement': 30,
   'burntbricks_earth': 37,
   'sunbricks_earth': 13,
   'muddaub_cement': 3,
   'sunbricks_cement': 4,
   'cement_cement': 1}
```

Again, we can read from the dict, but lets find it

```
max(wall_floor_dict, key=wall_floor_dict.get).split('_')

['muddaub', 'earth']
```

Now we can use a special feature of Jupyter notebooks to time them and see which is faster, called the <u>%timeit</u> magic. Displaying visual things is always slow, so we'll take the line that would display out of both options.

Now timing the pandas way, with a human deciding

```
%timeit -0

wall_floor_df = safi_df.groupby(['respondent_wall_type','respondent_floor_type'])

['instanceID'].count().reset_index()

2.4 ms ± 16.7 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)

<TimeitResult : 2.4 ms ± 16.7 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)>

t_pandas = _
```

And capture the time by using the _ it gets the std out from the previous cell.

For loop with required interpretation

```
%*timeit -0
wall_floor_dict = {}

for wall,floor in
zip(safi_df['respondent_wall_type'],safi_df['respondent_floor_type']):
    wf_key = '_'.join([wall,floor])
    if wf_key in wall_floor_dict.keys():
        wall_floor_dict[wf_key] +=1
    else:
        wall_floor_dict[wf_key] =1

max(wall_floor_dict, key=wall_floor_dict.get).split('_')
```

```
89.1~\mu s~\pm~238~ns per loop (mean \pm~std.~dev.~of~7~runs,~10000~loops~each)
```



🥊 Tip

This timing can be good when you have a large dataset, you can load a small part and and compare two ways of doing something you want to do with that. Then use only the faster on the whole dataset.

More detail on timing and profiling from the <u>text book</u>

```
<TimeitResult : 89.1 \mus \pm 238 ns per loop (mean \pm std. dev. of 7 runs, 10000 loops each)> t\_loop = \_
```

On my local computer it was about 12x faster with that loop. Here, we'll calculate it for the GitHub servers that host the course manual.

```
t_pandas.average/t_loop.average
26.916631381633525
```

For a more a complex question (say we wanted to know a statistic more complicated than the count) the loop gets harder (and probably slower), but the pandas operation stays about the same.

If you've made it this far, let me know how you found these notes.

Class 7: Visualization for EDA

Announcements

Syllabus updated

- 1. <u>rubric</u> for summarize and visualize are slightly changed
- 2. Please accept assignments if you plan to not complete for any reason

Assignment updated to clarify continuous and categorical variables

Loading Data

Importing the libraries for today. We'll continue plotting with pandas and we'll use <u>seaborn</u> as well. Seaborn provides higher level plotting functions and better formatting.

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

Loading the data as usual.

```
data_url = 'https://raw.githubusercontent.com/brownsarahm/python-socialsci-
files/master/data/SAFI_full_shortname.csv'
```

We know that the key_id column should be used as an index, not as data, so we'll use the index_col parameter t do that from the beginning.

```
safi_df = pd.read_csv(data_url,index_col='key_id')
safi_df.head()
```

🥊 Tip

You can also try it on your computer and see how it compares. Or try other loops/ways of doing it in pandas.

The alias for seaborn is sns the result of an <u>inside joke</u> among the developers in reference so <u>Samuel Norman Seaborn</u> on The West Wing, per <u>stackexchange</u>

key_id							
1	17 November 2016	1	2017-03- 23T09:49:57.000Z	2017-04- 02T17:29:08.000Z	Manica	Manica	E
2	17 November 2016	1	2017-04- 02T09:48:16.000Z	2017-04- 02T17:26:19.000Z	Manica	Manica	E
3	17 November 2016	3	2017-04- 02T14:35:26.000Z	2017-04- 02T17:26:53.000Z	Manica	Manica	E
4	17 November 2016	4	2017-04- 02T14:55:18.000Z	2017-04- 02T17:27:16.000Z	Manica	Manica	E
5	17 November 2016	5	2017-04- 02T15:10:35.000Z	2017-04- 02T17:27:35.000Z	Manica	Manica	E

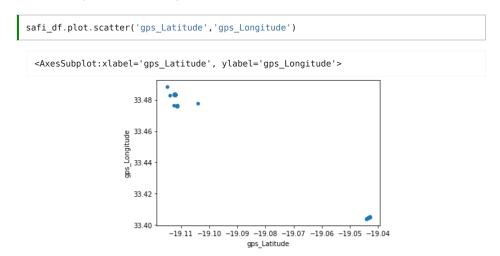
end

province district v

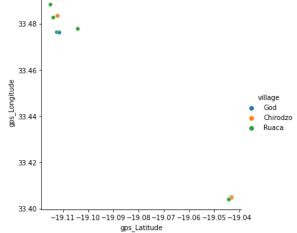
 $5 \text{ rows} \times 64 \text{ columns}$

We can make scatter plots as we saw Friday.

interview_date quest_no start



With seaborn, however, we can control it more, changing the color of the points based on a column of the data.



We can also plot a single variable to see the quantiles (the box is 25%-50%) and see if there are outliers (the points outside the box).



We can do some more conditiioning, even with only pandas. Using the by parameter will do a groupby operation first and then make the plot.

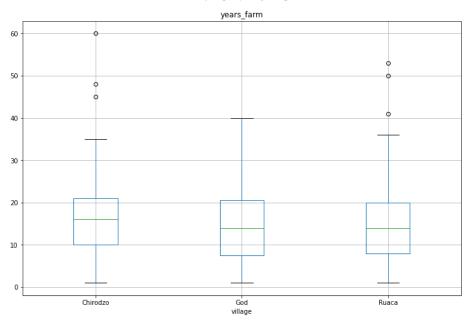


We can also make the figure larger $\,$

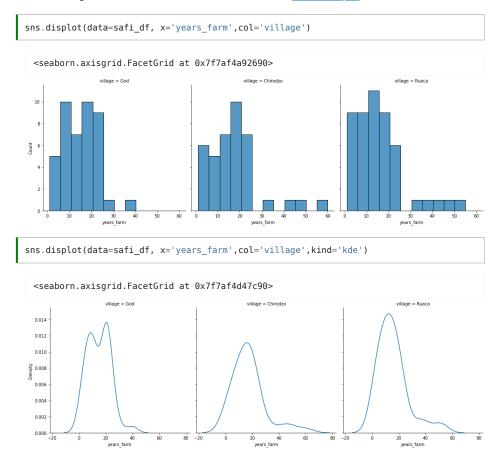
```
safi_df.boxplot('years_farm', by='village',figsize=(12,8))

<AxesSubplot:title={'center':'years_farm'}, xlabel='village'>
```





We can see how a single variable is distributed in more detail with the <u>seaborn displot</u>.



Updating Seaborn

the displot is new in seaborn 0.11, on your terminal (mac, linux) or anaconda prompt(Windows):

pip install update seaborn

Then restart your notebook's kernel and re-run all cells

or, in a notebook, you can update with

Regression plots

We can also plot with some calculations done for us already.

And we can make grids of plots with the row and col parameters. We can turn off the regression lines with the fit_reg parameter.

Questions after class

Test out the parameters of the plotting functions to see what they do.

Further Reading

- Pandas Plotting
- Seaborn Gallery
- Seaborn Tutorial

If you've made it this far, <u>let me know</u> how you found these notes.

Class 8: Visualization and Starting to Clean Data

Announcements

- 1. closing notebooks
 - o recall that they don't stop when you close the tabs
 - o you need to stop it at the terminal it launched from
 - o with: ctr + c as it says when you first launch a notebook
- 2. restart and rerun notebooks
 - o notebooks are a continuous REPL as long as you have it open
 - if you change code, you could have, for example a variable that is no longer defined in the notebook as written, but that still exists in memory, so code that depends on it will still run, for now, but not after you restart next (eg if we run it while grading)
 - o to check, <u>restart and rerun</u> your notebook
- 3. say hello on zoom for attendance

Setup

First we import packages and load data as normal.

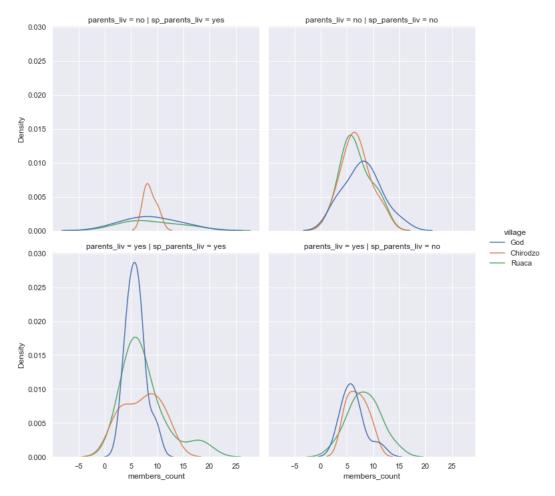
```
import pandas as pd
import seaborn as sns

data_url = 'https://raw.githubusercontent.com/brownsarahm/python-socialsci-
files/master/data/SAFI_full_shortname.csv'
sns.set(font_scale=1.25)
```

I've added one new command here, Seaborn's []set function] (https://seaborn.pydata.org/generated/seaborn.set.html). It sets a bunch of theme aspects for plotting, here I used it to increase the font size.

Warmup Activity

How recreate this plot?

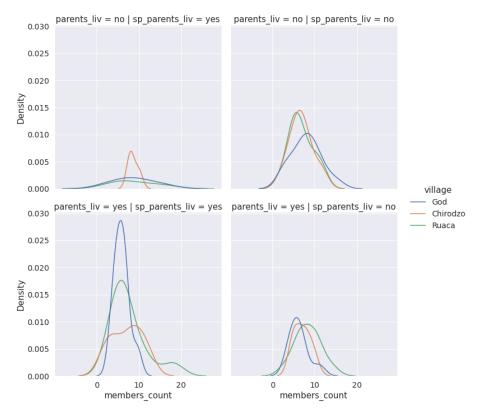


First, load the data

```
safi_df = pd.read_csv(data_url)
```

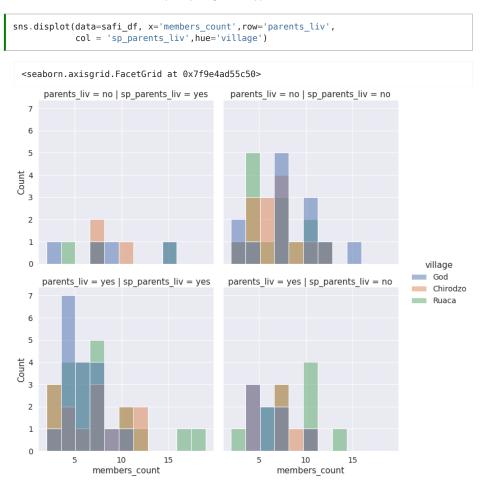
This is a <u>displot</u>.

<seaborn.axisgrid.FacetGrid at 0x7f9e501a6090>



This allows comparisons of how the village, respondent's parents living (parents_liv), and the spouse's (sp_parents_liv) influence the number of members of the household (members_count). The kind='kde' parameter uses the underlying kdeplot to apply kernel density estimation

We can understand better what this does, by comparing what happens when we take it out.



🥊 Tip

This is a general strategy. Testing out parameters with different values is a valuable way to learn what they do, and to build intuition about what they do. The documentation also has demos of a lot of values.

What about the rest of the columns?

We've used this SAFI dataset a lot, but we've only used a few of the many columns. Let's look at all of them.

```
safi_df.columns
```

```
Index(['key_id', 'interview_date', 'quest_no', 'start', 'end', 'province',
    'district', 'ward', 'village', 'years_farm', 'agr_assoc', 'note2',
    'no_membrs', 'members_count', 'remittance_money', 'years_liv',
    'parents_liv', 'sp_parents_liv', 'grand_liv', 'sp_grand_liv',
    'respondent_roof_type', 'respondent_wall_type',
    'buildings_in_compound', 'rooms', 'other_buildings', 'no_plots',
    'plots_count', 'water_use', 'no_group_count', 'yes_group_count',
    'no_enough_water', 'months_no_water', 'period_use', 'exper_other',
    'other_meth', 'res_change', 'memb_assoc', 'resp_assoc', 'fees_water',
    'affect_conflicts', 'note', 'need_money', 'money_source',
    'money_source_other', 'crops_contr', 'emply_lab', 'du_labour',
    'liv_owned', 'liv_owned_other', 'liv_count', 'poultry',
    'du_look_aftr_cows', 'items_owned', 'items_owned_other', 'no_meals',
    'months_lack_food', 'no_food_mitigation', 'gps_Latitude',
    'gps_Longitude', 'gps_Altitude', 'gps_Accuracy', 'instanceID'],
    dtype='object')
```

We can look at the first few row to recall what sort of data are available in each column.

	key_id	interview_date	quest_no	start	end	province	district
0	1	17 November 2016	1	2017-03- 23T09:49:57.000Z	2017-04- 02T17:29:08.000Z	Manica	Manica
1	2	17 November 2016	1	2017-04- 02T09:48:16.000Z	2017-04- 02T17:26:19.000Z	Manica	Manica
2	3	17 November 2016	3	2017-04- 02T14:35:26.000Z	2017-04- 02T17:26:53.000Z	Manica	Manica
3	4	17 November 2016	4	2017-04- 02T14:55:18.000Z	2017-04- 02T17:27:16.000Z	Manica	Manica
4	5	17 November 2016	5	2017-04- 02T15:10:35.000Z	2017-04- 02T17:27:35.000Z	Manica	Manica

5 rows × 65 columns

It might be interesting to use the items_owned column to compare find out if there are differences between farms based on what items they have.

Maybe we'd want to see how many farms own each item, maybe we want to know how many farms own bicycle. Let's try value_counts out like we used before.

```
safi_df['items_owned'].value_counts()
```

The problem is this gives the count of each *list* of items instead of each item. What we'd want instead is one column for each item, where the values in the column are 1 when the farm owns that item and 0 when they don't. This representation is sometimes called 1 hot encoding and other times (including in pandas) it's called dummy variables.

Using Dummy Variables

Let's try this function out, first on the village column.

```
pd.get_dummies(data= safi_df,columns=['village'])
```

	key_id	interview_date	quest_no	start	end	province	dist
0	1	17 November 2016	1	2017-03- 23T09:49:57.000Z	2017-04- 02T17:29:08.000Z	Manica	Mar
1	2	17 November 2016	1	2017-04- 02T09:48:16.000Z	2017-04- 02T17:26:19.000Z	Manica	Mar
2	3	17 November 2016	3	2017-04- 02T14:35:26.000Z	2017-04- 02T17:26:53.000Z	Manica	Mar
3	4	17 November 2016	4	2017-04- 02T14:55:18.000Z	2017-04- 02T17:27:16.000Z	Manica	Mar
4	5	17 November 2016	5	2017-04- 02T15:10:35.000Z	2017-04- 02T17:27:35.000Z	Manica	Mar
	•••						
126	127	18 May 2017	126	2017-05- 18T04:13:37.000Z	2017-05- 18T04:35:47.000Z	Manica	Mar
127	128	04 June 2017	193	2017-06- 04T09:36:20.000Z	2017-06- 04T10:13:32.000Z	Manica	Mar
128	129	04 June 2017	194	2017-06- 04T10:13:36.000Z	2017-06- 04T10:32:06.000Z	Manica	Mar
129	130	04 June 2017	199	2017-06- 04T10:33:55.000Z	2017-06- 04T10:52:22.000Z	Manica	Mar
130	131	04 June 2017	200	2017-06- 04T10:52:46.000Z	2017-06- 04T11:08:13.000Z	Manica	Mar

131 rows × 67 columns

Instead of scrolling, we can isolate the new columns we just created.

```
safi_df_villages = pd.get_dummies(data= safi_df,columns=['village'])
keep_cols = [col for col in safi_df_villages.columns if 'village_' in col]
safi_df_villages[keep_cols]
```

	village_Chirodzo	village_God	village_Ruaca
0	0	1	0
1	0	1	0
2	0	1	0
3	0	1	0
4	0	1	0
126	0	0	1
127	0	0	1
128	0	0	1
129	1	0	0
130	1	0	0

Note that now we have 3 columns, one for each village. This column was already usable, but it was well formatted an useful to illustrate what get_dummies does.



try this out and compare the two, test out get_dummies on another column to be sure you know what it does.

Now, we can try it on $items_owned$. _this time we'll filter the columns for display and inspection right away

```
safi_df_items = pd.get_dummies(data= safi_df,columns=['items_owned'])
keep_cols = [col for col in safi_df_items.columns if 'village_' in col]
safi_df_items[keep_cols]
```

0

1

2

3

126

127

128

129

130

131 rows × 0 columns

This still isn't quite what we want, because each value in the items_owned column *looks* like a list but it's actually a string, we can check that with the type function.

```
type(safi_df['items_owned'][0])
str
```

Cleaning One Cell of Data

So, let's clean the value from one cell of that column and see what else is going on. First, we save it to a variable.

```
farml_items = safi_df['items_owned'][0]
```

Let's look at it first.

```
farml_items

"['bicycle'; 'television'; 'solar_panel'; 'table']"
```

We could try first casting it to a list.

```
list(farml_items)
```

```
'ι',
'e',
']'j
```

That doesn't quite do it though, let's try separating it with the split method at the ';'.

```
farm1_items.split(';')

["['bicycle' ", " 'television' ", " 'solar_panel' ", " 'table']"]
```

This still has some extra characters, in it though. We should remove those, probably before we do the split so that we don't end up with empty items in the list.

```
farm1_items.replace('[','').replace(']','').replace("'","")

'bicycle ; television ; solar_panel ; table'
```

That is now ready to split.

```
farm1_items.replace('[','').replace(']','').replace("'","").split(';')

['bicycle ', ' television ', ' solar_panel ', ' table']
```

There are some empty spaces lest. Python has a string function strip that removes leading and trailing (on the ends) whitespace (spaces, tabs, new lines), we have to apply it to each individual item of that list above. We can use a list comprehension for this.

```
[i.strip() for i in
farml_items.replace('[','').replace(']','').replace("'","").split(';')]
```

```
['bicycle', 'television', 'solar_panel', 'table']
```

Applying this to the rest of the data

We can put that in a function so that we can reuse it.

Pandas provides us special function to apply a function to a dataframe along a given axis.

```
safi_df.apply(separate_items,axis=1)
```

```
Traceback (most recent call last)
<ipython-input-20-26c2d810f773> in <module>
----> 1 safi_df.apply(separate_items,axis=1)
/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/core/frame.py in apply(self, func, axis, raw, result_type, args,
**kwds)
   7546
                    kwds=kwds.
   7547
-> 7548
                return op.get_result()
   7549
            def applymap(self, func) -> "DataFrame":
  7550
/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/core/apply.py in get_result(self)
                   return self.apply_raw()
    178
    179
--> 180
                return self.apply_standard()
    181
            def apply_empty_result(self):
    182
/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/core/apply.py in apply_standard(self)
    269
    270
            def apply_standard(self):
                results, res_index = self.apply_series_generator()
--> 271
   272
    273
                # wrap results
/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/core/apply.py in apply_series_generator(self)
   298
                        for i, v in enumerate(series_gen):
    200
                            # ignore SettingWithCopy here in case the user mutates
--> 300
                            results[i] = self.f(v)
    301
                            if isinstance(results[i], ABCSeries):
   302
                                # If we have a view on v, we need to make a copy
because
<ipython-input-19-f7c175567037> in separate_items(row)
     4
     5
---> 6
            return [i.strip() for i in
row['items_owned'].replace('[','').replace("]','').replace("'","").split(';')]
AttributeError: 'float' object has no attribute 'replace'
```

This error means that there are some elements of that column that is a float instead of a string. We should check what that might be. We can check with a list comprehension and look at the ones that are float

Since the floats are nan, we know that our function more or less is the right thing to do, but we need to modify our function to accommodate those values. We can replace them with an empty list.

```
def separate_items(row):
    '''
    if type(row['items_owned'])==str:
        return [i.strip() for i in
row['items_owned'].replace('[','').replace(']','').replace("'","").split(';')]
    else:
        return []
```

We can try this again:

```
safi_df['items_owned_clean'] = safi_df.apply(separate_items,axis=1)
```

it runs successfully and we can look at the output.

```
safi_df[['items_owned_clean']].head()
```

items owned clean

- 0 [bicycle, television, solar_panel, table]
- 1 [cow_cart, bicycle, radio, cow_plough, solar_p...
- 2 [solar_torch]
- 3 [bicycle, radio, cow_plough, solar_panel, mobi...
- 4 [motorcyle, radio, cow_plough, mobile_phone]

This looks all good and we'll pick up from here on Monday.

Class 9: Preparing Data For Analysis

- Say hello in the zoom chat
- join Prismia

 $Checking \ types \ is \ an important \ part \ of \ cleaning \ data, we need \ to \ figure \ out \ what \ is \ wrong \ with \ data \ before \ we \ can \ fix \ it.$

Remember that data cleaning is a lot of exploration and iteration in the data. Let's review a few data types we've seen.

Warmup: type review

Lists are enclosed by square brackets([]), they're ordered and iterable.

```
type([char for char in 'abcde'])
list
```

Dictionaries are comprised of key: value pairs and enclosed in curly brackets ({}) they're iterable and indexable by the keys.

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

dict
```

This is a tuple, this data type is used to tie together items, but not usually to iterate over. It's used for pairs of things often or groups to iterate over multiple things at once. For example with the zip function.

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



Tip

If this were a regular analysis and not a tutorial, we would probably just edit the cell above, but for the purpose of making this tutorial more readable, we'll copy the function from above, into another cell, edit that cell, and overwrite in python memory by leaving it the same

If instead we wanted to be able to compare the two functions, we'd give the second one a different name.

function name.

```
tuple
```

Splitting a string makes a list.

```
type('a b c d e'.split(' '))
list
```

Loading Data

Loading the data and packages as normal.

```
# %load http://drsmb.co/310
import pandas as pd
import seaborn as sns
```

```
safi_url = 'https://raw.githubusercontent.com/brownsarahm/python-socialsci-
files/master/data/SAFI_full_shortname.csv'
safi_df = pd.read_csv(safi_url)
```

Exploring Types

On Friday we figured out that the following function would clean one cell of the $safi_df['items_owned']$ column.

```
def separate_items(row):
    '''
    return [i.strip() for i in
    row['items_owned'].replace('[','').replace(']','').replace("'","").split(';')]
```

But when we applied it, we got an error.

```
safi_df.apply(separate_items,axis=1)
```

```
AttributeError
                                         Traceback (most recent call last)
<ipython-input-8-26c2d810f773> in <module>
----> 1 safi_df.apply(separate_items,axis=1)
/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/core/frame.py in apply(self, func, axis, raw, result_type, args,
                    kwds=kwds,
  7547
-> 7548
               return op.get_result()
  7549
  7550
           def applymap(self, func) -> "DataFrame":
/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/core/apply.py in get_result(self)
   178
                    return self.apply_raw()
   179
--> 180
                return self.apply standard()
   181
   182
           def apply_empty_result(self):
/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/core/apply.py in apply_standard(self)
   269
   270
            def apply_standard(self):
--> 271
               results, res_index = self.apply_series_generator()
   272
   273
                # wrap results
/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/core/apply.py in apply_series_generator(self)
                       for i, v in enumerate(series_gen):
                           {\it \# ignore SettingWithCopy here in case the user mutates}
   299
--> 300
                            results[i] = self.f(v)
   301
                            if isinstance(results[i], ABCSeries):
   302
                                # If we have a view on v, we need to make a copy
because
<ipython-input-7-1663c693d570> in separate_items(row)
     3
     4
---> 5
           return [i.strip() for i in
row['items_owned'].replace('[','').replace(']','').replace("'","").split(';')]
AttributeError: 'float' object has no attribute 'replace'
```

Let's figure out why. First we'll look at the data again to remember what we expected.

```
safi_df['items_owned']
```

```
['bicycle'; 'television'; 'solar_panel'; ...
['cow_cart'; 'bicycle'; 'radio'; 'cow_pl...
0
1
                                                ['solar_torch']
2
        ['bicycle'; 'radio'; 'cow_plough'; 'sola...
3
        ['motorcyle'; 'radio'; 'cow_plough'; 'mo...
4
        ['motorcyle'; 'radio'; 'solar_panel']
['car'; 'lorry'; 'television'; 'radio';...
126
127
        ['radio'; 'solar_panel'; 'solar_torch'; ...
128
        ['cow_cart'; 'lorry'; 'motorcyle'; 'comp...
['radio'; 'cow_plough'; 'solar_panel'; '...
129
130
Name: items_owned, Length: 131, dtype: object
```

The error says that some item is a float, let's look at the value of only the ones that are float

```
[item_list for item_list in safi_df['items_owned'] if type(item_list)==float]
```

Now we can modify the cleaning function

```
def separate_items(row):
    '''
    if type(row['items_owned'])==str:
        return [i.strip() for i in
row['items_owned'].replace('[','').replace(']','').replace("'","").split(';')]
    else:
        return []
```

and apply it again, to see if it works now.

```
safi_df.apply(separate_items,axis=1)
```

```
[bicycle, television, solar_panel, table]
      [cow_cart, bicycle, radio, cow_plough, solar_p...
1
                                           [solar_torch]
      [bicycle, radio, cow_plough, solar_panel, mobi...
3
4
            [motorcyle, radio, cow_plough, mobile_phone]
126
                         [motorcyle, radio, solar_panel]
      [car, lorry, television, radio, sterio, cow_pl...
127
128
         [radio, solar_panel, solar_torch, mobile_phone]
129
      [cow_cart, lorry, motorcyle, computer, televis...
      [radio, cow_plough, solar_panel, solar_torch, ...
Length: 131, dtype: object
```

Dummies From Lists

```
safi_df['items_owned'] = safi_df.apply(separate_items,axis=1)
```

```
pd.get_dummies(safi_df['items_owned'].apply(pd.Series).stack())
```

		bicycle	car	computer	cow_cart	cow_plough	electricity	fridge	lorry	mobile_pho
0	0	1	0	0	0	0	0	0	0	
	1	0	0	0	0	0	0	0	0	
	2	0	0	0	0	0	0	0	0	
	3	0	0	0	0	0	0	0	0	
1	0	0	0	0	1	0	0	0	0	
130	1	0	0	0	0	1	0	0	0	
	2	0	0	0	0	0	0	0	0	
	3	0	0	0	0	0	0	0	0	
	4	0	0	0	0	0	0	0	0	
	5	0	0	0	0	0	0	0	0	
(24 47										

621 rows × 17 columns

```
safi_df['items_owned'].shape
```

```
(131,)
```

```
pd.get_dummies(safi_df['items_owned'].apply(pd.Series).stack()).index
```

```
pd.get_dummies(safi_df['items_owned'].apply(pd.Series).stack()).sum(level=0)
```

bicycle	car	computer	cow_cart	cow_plough	electricity	fridge	lorry	mobile_phone
1	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	1
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
0	0	0	0	1	0	0	0	1
								
0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	1	1	1
0	0	0	0	0	0	0	0	1
0	0	1	1	1	1	0	1	1
0	0	0	0	1	0	0	0	1
	1 1 0 1 0 0 0 0	1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0	1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	1 0 0 0 0 1 0 0 1 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1	1 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1	1 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0	1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 <td< th=""></td<>

121 rows × 17 columns

```
safi_df['items_owned'].head()
```

```
safi_item_df =
pd.get_dummies(safi_df['items_owned'].apply(pd.Series).stack()).sum(level=0)
```

Pandas concat

```
safi_df = pd.concat([safi_df, safi_item_df],axis=1)

safi_df.shape

(131, 82)

safi_df.head()
```

	key_id	interview_date	quest_no	start	end	province	district		
0	1	17 November 2016	1	2017-03- 23T09:49:57.000Z	2017-04- 02T17:29:08.000Z	Manica	Manica		
1	2	17 November 2016	1	2017-04- 02T09:48:16.000Z	2017-04- 02T17:26:19.000Z	Manica	Manica		
2	3	17 November 2016	3	2017-04- 02T14:35:26.000Z	2017-04- 02T17:26:53.000Z	Manica	Manica		
3	4	17 November 2016	4	2017-04- 02T14:55:18.000Z	2017-04- 02T17:27:16.000Z	Manica	Manica		
4	5	17 November 2016	5	2017-04- 02T15:10:35.000Z	2017-04- 02T17:27:35.000Z	Manica	Manica		
5 rows × 82 columns									

Working with Dummy Variables

```
safi_df['bicycle']
 0
        1.0
        1.0
        0.0
 3
        1.0
        0.0
 126
       0.0
 127
        0.0
 128
        0.0
 129
        0.0
 130
        0.0
 Name: bicycle, Length: 131, dtype: float64
sum(safi_df['bicycle'])
 nan
safi_df['bicycle'].sum()
 60.0
safi_df['bicycle'].count()
 121
safi_df['bicycle'].shape
 (131,)
pd.concat([safi\_df, safi\_item\_df], axis=0).shape
 (252, 82)
```

Questions after class

What does the get_dummies function do?

To illustrate, let's first make a small dataframe to look at it

```
ex_data = [['a',2,'x'],['b',5,'o'],['a',4,'o'],['c',2,'x'],['b',3,'x']]
ex_df = pd.DataFrame(data =ex_data, columns=['char','num','symbol'])
ex_df
```

	char	num	symbol
0	а	2	х
1	b	5	О
2	а	4	О
3	С	2	х
4	b	3	х

When we apply get_dummies to one column (a pd. Series) it makes a column for each value of that column. First we look at that column to focus on what it looks like. Note that there are 3 different values.

```
ex_df['char']

0    a
1    b
2    a
3    c
4    b
Name: char, dtype: object
```

Now conver it to dummy variables, now we have 3 columns. For each row there is a 1 in the column corresponding to the value and zeros elsewhere.

```
pd.get_dummies(ex_df['char'])

a b c
0 1 0 0
1 0 1 0
2 1 0 0
3 0 0 1
4 0 1 0
```

We can confirm by summing across the rows, it's all ones.

```
pd.get_dummies(ex_df['char']).sum(axis=1)

0    1
1    1
2    1
3    1
4    1
dtype: int64
```

If we sum down the columns, we get the count of each value

```
pd.get_dummies(ex_df['char']).sum(axis=0)

a    2
b    2
c    1
dtype: int64
```

it's the same thing as value_counts()

```
ex_df['char'].value_counts()

a    2
b    2
c    1
Name: char, dtype: int64
```

We can also apply it one column, but append it to the whole dataset this way.

```
pd.get_dummies(ex_df,columns=['char'])
```

	num	symbol	char_a	char_b	char_c
0	2	х	1	0	0
1	5	0	0	1	0
2	4	0	1	0	0
3	2	х	0	0	1
4	3	х	0	1	0

This way, it prepends the column values with the original column name. This way concatenates on it's own and we don't have to do it separately.

When we apply get_dummies on a whole DataFrame, without indicating a column it converts all of the columns of pandas type object

```
ex_df.dtypes

char object
num int64
symbol object
dtype: object

pd.get_dummies(ex_df,)
```

	num	char_a	char_b	char_c	symbol_o	symbol_x
0	2	1	0	0	0	1
1	5	0	1	0	1	0
2	4	1	0	0	1	0
3	2	0	0	1	0	1
4	3	0	1	0	0	1

Class 10: Cleaning review and Ray Summit Keynotes

- Say hello on zoom chat
- join prismia
- $\bullet\,$ sign up so you can watch Ray Summit talks by Pandas and Scikit learn

```
import pandas as pd

# %load http://drsmb.co/310
data_url = 'https://github.com/rhodyprog4ds/inclass-data/raw/main/ca_dds_summary.xlsx'

pd.read_excel(data_url)
```

	Ethnicity	Asian	Unnamed: 2	Unnamed: 3	Unnamed: 4	Black	Unnamed: 6	Unnamed: 7	Unn
0	Gender	Female	NaN	Male	NaN	Female	NaN	Male	
1	NaN	count	mean	count	mean	count	mean	count	
2	Age Cohort	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3	0 - 5	6	1544.33	NaN	NaN	NaN	NaN	NaN	
4	51+	9	56577.7	NaN	NaN	NaN	NaN	NaN	
5	13-17	10	3385.7	10	3632.5	NaN	NaN	8	43
6	18-21	18	10083.3	23	9218.52	NaN	NaN	7	91
7	22-50	12	39470.9	17	39657.9	8	41284.1	9	;
8	6-12	6	2165.33	12	2164.17	7	2566.57	NaN	

pd.read_excel(data_url,header=list(range(4)))

	Ethnicity	Asian				Black		
	Gender	Female		Male		Female		
	Unnamed: 0_level_2	count	mean	count	mean	count mean		
	Age Cohort	Unnamed: 1_level_3	Unnamed: 2_level_3	Unnamed: 3_level_3	Unnamed: 4_level_3	Unnamed: 5_level_3	Unnam 6_leve	
0	0 - 5	6	1544.333333	NaN	NaN	NaN	N	
1	51+	9	56577.666667	NaN	NaN	NaN	N	
2	13-17	10	3385.700000	10.0	3632.500000	NaN	N	
3	18-21	18	10083.277778	23.0	9218.521739	NaN	N	
4	22-50	12	39470.916667	17.0	39657.882353	8.0	41284.1250	
5	6-12	6	2165.333333	12.0	2164.166667	7.0	2566.5714	

ca_dds_df = pd.read_excel(data_url,header=list(range(4)))

ca_dds_df.head()

	Ethnicity	Asian			Black			
	Gender	Female		Male		Female		
	Unnamed: 0_level_2	count	mean	count	mean	count	mean	
	Age Cohort	Unnamed: 1_level_3	Unnamed: 2_level_3	Unnamed: 3_level_3	Unnamed: 4_level_3	Unnamed: 5_level_3	Unnamed: 6_level_3	
0	0 - 5	6	1544.333333	NaN	NaN	NaN	NaN	
1	51+	9	56577.666667	NaN	NaN	NaN	NaN	
2	13-17	10	3385.700000	10.0	3632.500000	NaN	NaN	
3	18-21	18	10083.277778	23.0	9218.521739	NaN	NaN	
4	22-50	12	39470.916667	17.0	39657.882353	8.0	41284.125	

ca_dds_df.columns

```
MultiIndex([(
                                                                                                                                                                                                                                                                                                                                                                                                                               'mean', ...),
'count', ...),
'mean', ...),
                                                                                                                                                                                          'Asian', 'Female',
                                                                                                                                                                                          'Asian', 'Male',
'Asian', 'Male',
                                                                                                                                                                                        'Asian', 'Male',
'Black', 'Female',
'Black', 'Female',
                                                                                                                                                                                                                                                                                                                                                                                                                            'count', ...),
                                                                                                                                                                                                                                                                                                                                                                                                                                    'mean', ...),
                                                                                                                                                                  'Black', 'Male',
'Black', 'Male',
'Hispanic', 'Female',
                                                                                                                                                                                                                                                                                                                                                                                                                          'count', ...),
                                                                                                                                                                                                                                                                                                                                                                                                                          'mean', ...),
                                                                                    ( 'Hispanic', 'Female', ( 'Hispanic', 'Female', ( 'Hispanic', 'Male', ( 'Hispanic', 'Male', ( 'White not Hispanic', 'Female', ( 'White not Hispanic', 'Female', 'White not Hispanic', 'Female', 'White not Hispanic', 'Female', 'White not Hispanic', 'Female', 'White not Hispanic', 'Male', 'White not Hispanic', 'The not Hispanic'
                                                                                                                                                                                                                                                                                                                                                                                                                          'mean', ...),
'count', ...),
                                                                                                                                                                                                                                                                                                                                                                                                                                 'mean', ...),
                                                                                                                                                                                                                                                                                                                                                                                                                             'count', ...),
'mean', ...),
                                                                                      ('White not Hispanic', ('White not Hispanic',
                                                                                                                                                                                                                                                                                                                                                                                                                             'count', ...),
'mean', ...)],
                                                                                                                                                                                                                                                                'Male',
                                                                                                                                                                                                                                                                        'Male',
```

Ray Summit Notes

contribute things you learned here

Pandas, by Wes

- Pandas was designed to do data science on your laptop
- It's designed to be coupled tightly to numpy, which is why it's not very fast, especially with strings

•

Scikit Learn

- Data science for the many not the mighty
- Machine learning for all

Class 11: Cleaning Data

```
import pandas as pd

# %load http://drsmb.co/310
data_url = 'https://github.com/rhodyprog4ds/inclass-data/raw/main/ca_dds_summary.xlsx'

ca_dds_df = pd.read_excel(data_url, header=list(range(3)))
ca_dds_df
```

	Ethnicity	Asian				Black			
	Gender	Female		Male		Female		Male	
	Unnamed: 0_level_2	count	mean	count	mean	count	mean	count	
0	Age Cohort	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	0 - 5	6.0	1544.333333	NaN	NaN	NaN	NaN	NaN	
2	51+	9.0	56577.666667	NaN	NaN	NaN	NaN	NaN	
3	13-17	10.0	3385.700000	10.0	3632.500000	NaN	NaN	8.0	
4	18-21	18.0	10083.277778	23.0	9218.521739	NaN	NaN	7.0	
5	22-50	12.0	39470.916667	17.0	39657.882353	8.0	41284.125000	9.0	;
6	6-12	6.0	2165.333333	12.0	2164.166667	7.0	2566.571429	NaN	

```
print(ca_dds_df)
```

```
Ethnicity Asian
                                                               Black \
              Gender Female
                                           Male
                                                               Female
 Unnamed: 0_level_2
                                     mean count
                                                               count
          Age Cohort
                        NaN
                                            NaN
                                      NaN
                                                          NaN
                                                                 NaN
               0 - 5
                        6.0
                              1544.333333
                                            NaN
1
                                                          NaN
                                                                 NaN
               51 +
                        9.0 56577.666667
2
                                            NaN
                                                          NaN
                                                                 NaN
3
               13-17
                       10.0
                              3385.700000
                                           10.0
                                                  3632.500000
                                                                 NaN
4
               18-21
                       18.0
                             10083.277778
                                           23.0
                                                  9218.521739
                                                                  NaN
5
               22-50
                       12.0
                             39470.916667
                                           17.0
                                                 39657.882353
                                                                 8.0
                6-12
                             2165.333333 12.0
                                                  2164.166667
                                                                 7.0
6
                       6.0
                                    Hispanic
                 Male
                                                            Male
                                      Female
           mean count
                               mean
                                       count
                                                      mean count
0
            NaN
                  NaN
                                NaN
                                         NaN
                                                       NaN
                                                             NaN
1
            NaN
                  NaN
                                NaN
                                        18.0
                                               1431.333333
                                                            26.0
2
            NaN
                  NaN
                                NaN
                                         NaN
                                                       NaN
                                                            12.0
                        4367.250000
                                        50.0
                                               4043.720000
            NaN
                  8.0
                                                            53.0
4
            NaN
                  7.0
                        9119.428571
                                        41.0
                                              10199.390244
                                                            37.0
5
  41284.125000
                  9.0
                       39941.000000
                                              41005.541667
                                        24.0
                                                            19.0
6
   2566.571429
                  NaN
                                NaN
                                        54.0
                                               2443.777778 37.0
                White not Hispanic
                            Female
                                                  Male
           mean
                             count
                                            mean count
                                                                mean
0
            NaN
                               NaN
                                             NaN
                                                   NaN
                                                                 NaN
    1366.807692
                              11.0
                                     1316.818182
                                                   9.0
                                                         1428.111111
  53019.000000
                              37.0
                                   53449.486486
                                                  29.0
                                                        51676.448276
    3871.849057
                                     3818.810811
                                                  30.0
                                                         4009.866667
3
                              37.0
4
    9694.405405
                              36.0
                                   10452.944444
                                                  33.0
                                                         9784.090909
5
  40821.263158
                              60.0
                                   42113.816667 73.0
                                                        38604.452055
   2120.135135
                              24.0
                                     2112.750000
                                                  22.0
                                                         1986.272727
```

ca dds df.index

RangeIndex(start=0, stop=7, step=1)

ca_dds_df.columns[0]

('Ethnicity', 'Gender', 'Unnamed: 0_level_2')

ca_dds_df.set_index(ca_dds_df.columns[0],inplace=True)

ca_dds_df

Asian **Black Female** Male **Female** Male count mean count mean count mean count mea (Ethnicity, Gender, Unnamed: 0_level_2) Age NaN NaN NaN NaN NaN NaN NaN Cohort 0-5 1544.333333 NaN NaN NaN NaN NaN 51+ 9.0 56577.666667 NaN NaN NaN NaN NaN 13-17 10.0 3385.700000 10.0 3632.500000 NaN NaN 8.0 43 18-21 18.0 10083.277778 23.0 9218.521739 7.0 91 NaN 22-50 12.0 39470.916667 17.0 39657.882353 8.0 41284.125000 9.0 399 6-12 6.0 2165.333333 12.0 2164.166667 7.0 2566.571429 NaN

 $ca_dds_df.dropna(how='all',inplace=\pmb{True})$

ca_dds_df.index

```
Index(['\ 0\ -\ 5',\ '\ 51\ +',\ '13-17',\ '18-21',\ '22-50',\ '6-12'],\ dtype='object',\ name='('Ethnicity',\ 'Gender',\ 'Unnamed:\ 0\_level\_2'))
```

```
ca_dds_df.index.rename('Age Cohort',inplace=True)
```

```
ca_dds_df.T.unstack(level= [0,1]).T.reset_index()
```

	Age Cohort	level_1	level_2	count	mean
0	0 - 5	Asian	Female	6.0	1544.333333
1	0 - 5	Asian	Male	NaN	NaN
2	0 - 5	Black	Female	NaN	NaN
3	0 - 5	Black	Male	NaN	NaN
4	0 - 5	Hispanic	Female	18.0	1431.333333
5	0 - 5	Hispanic	Male	26.0	1366.807692
6	0 - 5	White not Hispanic	Female	11.0	1316.818182
7	0 - 5	White not Hispanic	Male	9.0	1428.111111
8	51+	Asian	Female	9.0	56577.666667
9	51+	Asian	Male	NaN	NaN
10	51+	Black	Female	NaN	NaN
11	51+	Black	Male	NaN	NaN
12	51+	Hispanic	Female	NaN	NaN
13	51+	Hispanic	Male	12.0	53019.000000
14	51+	White not Hispanic	Female	37.0	53449.486486
15	51+	White not Hispanic	Male	29.0	51676.448276
16	13-17	Asian	Female	10.0	3385.700000
17	13-17	Asian	Male	10.0	3632.500000
18	13-17	Black	Female	NaN	NaN
19	13-17	Black	Male	8.0	4367.250000
20	13-17	Hispanic	Female	50.0	4043.720000
21	13-17	Hispanic	Male	53.0	3871.849057
22	13-17	White not Hispanic	Female	37.0	3818.810811
23	13-17	White not Hispanic	Male	30.0	4009.866667
24	18-21	Asian	Female	18.0	10083.277778
25	18-21	Asian	Male	23.0	9218.521739
26	18-21	Black	Female	NaN	NaN
27	18-21	Black	Male	7.0	9119.428571
28	18-21	Hispanic	Female	41.0	10199.390244
29	18-21	Hispanic	Male	37.0	9694.405405
30	18-21	White not Hispanic	Female	36.0	10452.944444
31	18-21	White not Hispanic	Male	33.0	9784.090909
32	22-50	Asian	Female	12.0	39470.916667
33	22-50	Asian	Male	17.0	39657.882353
34	22-50	Black	Female	8.0	41284.125000
35	22-50	Black	Male	9.0	39941.000000
36	22-50	Hispanic	Female	24.0	41005.541667
37	22-50	Hispanic	Male	19.0	40821.263158
38	22-50	White not Hispanic	Female	60.0	42113.816667
39	22-50	White not Hispanic	Male	73.0	38604.452055
40	6-12	Asian	Female	6.0	2165.333333
41	6-12	Asian	Male	12.0	2164.166667
42	6-12	Black	Female	7.0	2566.571429
43	6-12	Black	Male	NaN	NaN
44	6-12	Hispanic	Female	54.0	2443.777778

```
Age Cohort
                           level_1 level_2 count
                                                          mean
 45
           6-12
                          Hispanic
                                     Male
                                            37.0
                                                   2120.135135
           6-12 White not Hispanic Female
                                            24.0
                                                   2112.750000
 46
           6-12 White not Hispanic
                                     Male
                                            22.0
                                                   1986.272727
ca_dds_clean = ca_dds_df.T.unstack(level= [0,1]).T.reset_index()
ca_dds_clean.head()
    Age Cohort
                 level_1 level_2 count
                                               mean
 0
          0 - 5
                                    6.0 1544.333333
                  Asian Female
          0 - 5
 1
                  Asian
                           Male
                                  NaN
                                                NaN
 2
          0 - 5
                   Black Female
                                  NaN
                                                NaN
          0 - 5
 3
                   Black
                           Male
                                  NaN
                                                NaN
          0-5 Hispanic Female
 4
                                  18.0 1431.333333
ca_dds_clean.columns[1] = 'Race'
 TypeError
                                           Traceback (most recent call last)
 <ipython-input-14-f771ccd0e865> in <module>
 ----> 1 ca_dds_clean.columns[1] = 'Race'
 /opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
 packages/pandas/core/indexes/base.py in __setitem__(self, key, value)
    4079
    4080
             def __setitem__(self, key, value):
 -> 4081
                 raise TypeError("Index does not support mutable operations")
    4082
             def __getitem__(self, key):
    4083
 TypeError: Index does not support mutable operations
type(ca_dds_clean.columns)
 pandas.core.indexes.base.Index
ca_dds_clean.rename(columns= {'level_1':'Race', 'level_2':'Gender'},inplace=True)
ca_dds_clean.head()
    Age Cohort
                   Race Gender count
                                                mean
 0
                                    6.0 1544.333333
          0 - 5
                   Asian
                         Female
 1
          0-5
                  Asian
                           Male
                                   NaN
                                                 NaN
 2
          0 - 5
                   Black
                         Female
                                   NaN
                                                 NaN
 3
          0 - 5
                   Black
                           Male
                                   NaN
                                                 NaN
 4
          0-5 Hispanic Female
                                   18.0 1431.333333
ca_dds_clean['count'][0]
 6.0
ca_dds_clean.Gender[0]
 'Female'
ca_dds_clean.Gender[0].lower()
 'female'
```

```
clean_cols = {c:c.lower().replace(' ','_') for c in ca_dds_clean.columns}
ca_dds_clean.rename(columns=clean_cols,inplace=True)
ca_dds_clean.head()
```

	age_cohort	race	gender	count	mean
0	0 - 5	Asian	Female	6.0	1544.333333
1	0 - 5	Asian	Male	NaN	NaN
2	0 - 5	Black	Female	NaN	NaN
3	0 - 5	Black	Male	NaN	NaN
4	0 - 5	Hispanic	Female	18.0	1431.333333

```
ca_dds_clean.fillna('*')
```

	age_cohort	race	gender	count	mean
0	0 - 5	Asian	Female	6	1544.33
1	0 - 5	Asian	Male	*	*
2	0 - 5	Black	Female	*	*
3	0 - 5	Black	Male	*	*
4	0 - 5	Hispanic	Female	18	1431.33
5	0 - 5	Hispanic	Male	26	1366.81
6	0 - 5	White not Hispanic	Female	11	1316.82
7	0 - 5	White not Hispanic	Male	9	1428.11
8	51+	Asian	Female	9	56577.7
9	51+	Asian	Male	*	*
10	51+	Black	Female	*	*
11	51+	Black	Male	*	*
12	51+	Hispanic	Female	*	*
13	51+	Hispanic	Male	12	53019
14	51+	White not Hispanic	Female	37	53449.5
15	51+	White not Hispanic	Male	29	51676.4
16	13-17	Asian	Female	10	3385.7
17	13-17	Asian	Male	10	3632.5
18	13-17	Black	Female	*	*
19	13-17	Black	Male	8	4367.25
20	13-17	Hispanic	Female	50	4043.72
21	13-17	Hispanic	Male	53	3871.85
22	13-17	White not Hispanic	Female	37	3818.81
23	13-17	White not Hispanic	Male	30	4009.87
24	18-21	Asian	Female	18	10083.3
25	18-21	Asian	Male	23	9218.52
26	18-21	Black	Female	*	*
27	18-21	Black	Male	7	9119.43
28	18-21	Hispanic	Female	41	10199.4
29	18-21	Hispanic	Male	37	9694.41
30	18-21	White not Hispanic	Female	36	10452.9
31	18-21	White not Hispanic	Male	33	9784.09
32	22-50	Asian	Female	12	39470.9
33	22-50	Asian	Male	17	39657.9
34	22-50	Black	Female	8	41284.1
35	22-50	Black	Male	9	39941
36	22-50	Hispanic	Female	24	41005.5
37	22-50	Hispanic	Male	19	40821.3
38	22-50	White not Hispanic	Female	60	42113.8
39	22-50	White not Hispanic	Male	73	38604.5
40	6-12	Asian	Female	6	2165.33
41	6-12	Asian	Male	12	2164.17
42	6-12	Black	Female	7	2566.57
43	6-12	Black	Male	*	*
44	6-12	Hispanic	Female	54	2443.78

mean	count	gender	race	age_cohort	
2120.14	37	Male	Hispanic	6-12	45
2112.75	24	Female	White not Hispanic	6-12	46
1986.27	22	Male	White not Hispanic	6-12	47

Class 12: Constructing Datasets from Multiple Sources

```
import pandas as pd
```

To use relative paths as in class (data/2018-games.csv) instead of a full url

https://raw.githubusercontent.com/rhodyprog4ds/inclass-data/main/2018-games.csv, download data from this <u>GitHub repo</u> by clicking on the green code button and choosing .zip. Then unzip the data and save it in a data folder in the same folder as the notebook. For the class notes, the urls make it so that the notebook can run without having to store the data in another place.

```
games_df18 = pd.read_csv('https://raw.githubusercontent.com/rhodyprog4ds/inclass-
data/main/2018-games.csv')
games_df19 = pd.read_csv('https://raw.githubusercontent.com/rhodyprog4ds/inclass-
data/main/2019-games.csv')
```

Stacking DataFrames

```
games_df18.head(2)
              GAME_DATE_EST GAME_ID GAME_STATUS_TEXT HOME_TEAM_ID VISITOF
 0
       16196
                   2019-06-13 41800406
                                                                1610612744
                                                       Final
                                                                                  16
                   2019-06-10 41800405
       16197
                                                       Final
                                                                1610612761
 1
                                                                                  16
2 rows × 22 columns
games_df19.head(2)
              GAME_DATE_EST GAME_ID GAME_STATUS_TEXT HOME_TEAM_ID VISITOF
                   2020-03-01 21900895
                                                       Final
                                                                1610612766
                                                                                  16
                   2020-03-01 21900896
                                                       Final
                                                                1610612750
                                                                                  16
2 rows × 22 columns
games_df18.shape
 (1378, 22)
games_df19.shape
 (965, 22)
games_df = pd.concat([games_df18,games_df19])
games_df.shape
 (2343, 22)
games_df.columns
```

games_df.drop(columns= 'Unnamed: 0',inplace=True,)
games_df.head(2)

GAME_DATE_EST GAME_ID GAME_STATUS_TEXT HOME_TEAM_ID VISITOR_TEAM_ID

 0
 2019-06-13
 41800406
 Final
 1610612744
 1610612761

 1
 2019-06-10
 41800405
 Final
 1610612761
 1610612744

2 rows × 21 columns

Merging Data Frames

teams_df = pd.read_csv('https://raw.githubusercontent.com/rhodyprog4ds/inclassdata/main/teams.csv')
teams_df.head(2)

LEAGUE_ID TEAM_ID MIN_YEAR MAX_YEAR ABBREVIATION NICKNAME YEARF

0 0 1610612737 1949 2019 ATL Hawks

1 0 1610612738 1946 2019 BOS Celtics

 $\label{eq:mergel_df} $$ merge(teams_df,games_df,left_on='TEAM_ID', right_on = 'HOME_TEAM_ID') $$ mergel_df.head(2) $$$

LEAGUE_ID TEAM_ID MIN_YEAR MAX_YEAR ABBREVIATION NICKNAME YEARF

0 0 1610612737 1949 2019 ATL Hawks

1 0 1610612737 1949 2019 ATL Hawks

2 rows × 35 columns

mergel_df.shape

(2343, 35)

merge2_df = pd.merge(teams_df, games_df,left_on='TEAM_ID', right_on = 'HOME_TEAM_ID',
how='outer')

merge2_df.head(2)

LEAGUE_ID TEAM_ID MIN_YEAR MAX_YEAR ABBREVIATION NICKNAME YEARF

0 0 1610612737 1949 2019 ATL Hawks

1 0 1610612737 1949 2019 ATL Hawks

2 rows × 35 columns

merge2_df.shape

(2343, 35)

	LEAGUE_ID	TEAM_ID	MIN_YEAR	MAX_YEAR	YEARFOUNDED	ļ
ARENA						
AT&T Center	0.0	1.610613e+09	1976.000000	2019.0	1976.000000	
American Airlines Center	0.0	1.610613e+09	1980.000000	2019.0	1980.000000	
AmericanAirlines Arena	0.0	1.610613e+09	1988.000000	2019.0	1988.000000	
Amway Center	0.0	1.610613e+09	1989.000000	2019.0	1989.000000	
Bankers Life Fieldhouse	0.0	1.610613e+09	1976.000000	2019.0	1976.000000	
Barclays Center	0.0	1.610613e+09	1976.000000	2019.0	1976.000000	
Capital One Arena	0.0	1.610613e+09	1961.000000	2019.0	1961.000000	
Chase Center	0.0	1.610613e+09	1946.000000	2019.0	1946.000000	
Chesapeake Energy Arena	0.0	1.610613e+09	1967.000000	2019.0	1967.000000	
FedExForum	0.0	1.610613e+09	1995.000000	2019.0	1995.000000	
Fiserv Forum	0.0	1.610613e+09	1968.000000	2019.0	1968.000000	
Golden 1 Center	0.0	1.610613e+09	1948.000000	2019.0	1948.000000	
Little Caesars Arena	0.0	1.610613e+09	1948.000000	2019.0	1948.000000	
Madison Square Garden	0.0	1.610613e+09	1946.000000	2019.0	1946.000000	
Moda Center	0.0	1.610613e+09	1970.000000	2019.0	1970.000000	
Pepsi Center	0.0	1.610613e+09	1976.000000	2019.0	1976.000000	
Quicken Loans Arena	0.0	1.610613e+09	1970.000000	2019.0	1970.000000	
Scotiabank Arena	0.0	1.610613e+09	1995.000000	2019.0	1995.000000	
Smoothie King Center	0.0	1.610613e+09	2002.000000	2019.0	2002.000000	
Spectrum Center	0.0	1.610613e+09	1988.000000	2019.0	1988.000000	
Staples Center	0.0	1.610613e+09	1959.210191	2019.0	1959.210191	
State Farm Arena	0.0	1.610613e+09	1949.000000	2019.0	1949.000000	
TD Garden	0.0	1.610613e+09	1946.000000	2019.0	1946.000000	
Talking Stick Resort Arena	0.0	1.610613e+09	1968.000000	2019.0	1968.000000	
Target Center	0.0	1.610613e+09	1989.000000	2019.0	1989.000000	
Toyota Center	0.0	1.610613e+09	1967.000000	2019.0	1967.000000	
United Center	0.0	1.610613e+09	1966.000000	2019.0	1966.000000	
Vivint Smart Home Arena	0.0	1.610613e+09	1974.000000	2019.0	1974.000000	
Wells Fargo Center	0.0	1.610613e+09	1949.000000	2019.0	1949.000000	
29 rows x 25 column	ıc					

29 rows × 25 columns

How to combine the same data for different outcomes

players18 = pd.read_csv('https://raw.githubusercontent.com/rhodyprog4ds/inclassdata/main/2018-players.csv')
players19 = pd.read_csv('https://raw.githubusercontent.com/rhodyprog4ds/inclassdata/main/2019-players.csv')
players18.head()

	Unnamed: 0	PLAYER_NAME	TEAM_ID	PLAYER_ID	SEASON
0	626	Kawhi Leonard	1610612761	202695	2018
1	627	Pascal Siakam	1610612761	1627783	2018
2	628	Marc Gasol	1610612761	201188	2018
3	629	Danny Green	1610612761	201980	2018
4	630	Kyle Lowry	1610612761	200768	2018

players18.shape, players19.shape

((748, 5), (626, 5))

pd.concat([players18,players19]).shape

(1374, 5)

pd.merge(players18,players19,)

Unnamed: 0 PLAYER_NAME TEAM_ID PLAYER_ID SEASON

Using how='inner' gives us only the 'PLAYER_ID' that are in both DataFrames

pd.merge(players18,players19,on='PLAYER_ID', how='inner')

PLAYE	Unnamed: 0_y	SEASON_x	PLAYER_ID	TEAM_ID_x	PLAYER_NAME_x	Unnamed: 0_x	
Kav	299	2018	202695	1610612761	Kawhi Leonard	626	0
Pas	275	2018	1627783	1610612761	Pascal Siakam	627	1
	276	2018	201188	1610612761	Marc Gasol	628	2
	276	2018	201188	1610612763	Marc Gasol	1157	3
Di	202	2018	201980	1610612761	Danny Green	629	4
Α	561	2018	203583	1610612760	Abdul Gaddy	1339	533
Andr	566	2018	203460	1610612760	Andre Roberson	1342	534
١	24	2018	203658	1610612755	Norvel Pelle	1348	535
Denze	58	2018	1627756	1610612741	Denzel Valentine	1350	536
	573	2018	203912	1610612754	C.J. Wilcox	1353	537

538 rows × 9 columns

Using outer gives us the 'PLAYER_ID' that are in one or both of the DataFrame

pd.merge(players18,players19,on='PLAYER_ID', how='outer')

	Unnamed: 0_x	PLAYER_NAME_x	TEAM_ID_x	PLAYER_ID	SEASON_x	Unnamed: 0_y	PLAY
0	626.0	Kawhi Leonard	1.610613e+09	202695	2018.0	299.0	Ka
1	627.0	Pascal Siakam	1.610613e+09	1627783	2018.0	275.0	Р
2	628.0	Marc Gasol	1.610613e+09	201188	2018.0	276.0	
3	1157.0	Marc Gasol	1.610613e+09	201188	2018.0	276.0	
4	629.0	Danny Green	1.610613e+09	201980	2018.0	202.0	
			•••				
922	NaN	NaN	NaN	1629097	NaN	619.0	
923	NaN	NaN	NaN	203461	NaN	621.0	Antł
924	NaN	NaN	NaN	203906	NaN	623.0	D
925	NaN	NaN	NaN	1629755	NaN	624.0	Has
926	NaN	NaN	NaN	1629721	NaN	625.0	Ja
007	0 1						

927 rows × 9 columns

Using left gives us the 'PLAYER_ID' that are in the left(players18) DataFrame, including those that are in both DataFrame right would give players in the players19 DataFrame or both DataFrames

pd.merge(players18,players19,on='PLAYER_ID', how='left')

	Unnamed: 0_x	PLAYER_NAME_x	TEAM_ID_x	PLAYER_ID	SEASON_x	Unnamed: 0_y	PLAYE
0	626	Kawhi Leonard	1610612761	202695	2018	299.0	Kav
1	627	Pascal Siakam	1610612761	1627783	2018	275.0	Pas
2	628	Marc Gasol	1610612761	201188	2018	276.0	
3	629	Danny Green	1610612761	201980	2018	202.0	Di
4	630	Kyle Lowry	1610612761	200768	2018	451.0	
749	1369	Tyrius Walker	1610612752	1629246	2018	NaN	
750	1370	Marcus Lee	1610612748	1629159	2018	NaN	
751	1371	Trey Lewis	1610612762	1629163	2018	NaN	
752	1372	Emanuel Terry	1610612743	1629150	2018	NaN	
753	1373	Justin Bibbs	1610612738	1629167	2018	NaN	

754 rows × 9 columns

Try it yourself

Try different merges and inspect them:

- how many rows & columns?
- Where are NaN values inserted?
- What rows from the original datasets are not included?
- describe each type of merge in your own words

Split a DataFrame into separate data frames by subsetting the columns and indexing the rows with loc, then use concat to put it back together. Programmatically check that it's back together correctly.

Class 13: Data from mulitple sources and Databases

Welcome:

- 1. Say hello on zoom
- $2.\, Download\, data\, from\, Mondays'\, notes\, if\, you\, dont\, have\, it\, already$
- 3. log onto prismiaa

Merging review

```
df18_players = pd.read_csv('data/2018-players.csv')
df19_players = pd.read_csv('data/2019-players.csv')
```

```
FileNotFoundError
                                        Traceback (most recent call last)
<ipython-input-2-e28c0e41012a> in <module>
---> 1 df18_players = pd.read_csv('data/2018-players.csv')
     2 df19_players = pd.read_csv('data/2019-players.csv')
/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/io/parsers.py in read_csv(filepath_or_buffer, sep, delimiter, header,
names, index_col, usecols, squeeze, prefix, mangle_dupe_cols, dtype, engine,
converters, true_values, false_values, skipinitialspace, skiprows, skipfooter, nrows,
na_values, keep_default_na, na_filter, verbose, skip_blank_lines, parse_dates,
infer_datetime_format, keep_date_col, date_parser, dayfirst, cache_dates, iterator,
chunksize, compression, thousands, decimal, lineterminator, quotechar, quoting,
doublequote, escapechar, comment, encoding, dialect, error_bad_lines, warn_bad_lines,
delim_whitespace, low_memory, memory_map, float_precision)
   684
   685
--> 686
           return read(filepath or buffer, kwds)
   687
   688
/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/io/parsers.py in _read(filepath_or_buffer, kwds)
   450
   451
            # Create the parser.
--> 452
           parser = TextFileReader(fp_or_buf, **kwds)
   453
   454
           if chunksize or iterator:
/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/io/parsers.py in __init__(self, f, engine, **kwds)
                   self.options["has_index_names"] = kwds["has_index_names"]
   945
--> 946
               self._make_engine(self.engine)
   947
           def close(self):
   948
/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/io/parsers.py in _make_engine(self, engine)
   1176
           def _make_engine(self, engine="c"):
   1177
               if engine == "c":
-> 1178
                   self._engine = CParserWrapper(self.f, **self.options)
  1179
               else:
   1180
                   if engine == "python":
/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
2007
-> 2008
               self._reader = parsers.TextReader(src, **kwds)
  2009
               self.unnamed cols = self. reader.unnamed cols
   2010
pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader.__cinit__()
pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._setup_parser_source()
FileNotFoundError: [Errno 2] No such file or directory: 'data/2018-players.csv'
```

```
df18_players.shape, df19_players.shape
```

```
NameError Traceback (most recent call last)
<ipython-input-3-e14e0125446c> in <module>
----> 1 df18_players.shape, df19_players.shape

NameError: name 'df18_players' is not defined
```

```
pd.merge(df18_players,df19_players,how='inner',on='PLAYER_ID')
```

```
Traceback (most recent call last)
 NameError
 <ipython-input-4-f10edcc4c468> in <module>
 ---> 1 pd.merge(df18_players,df19_players,how='inner',on='PLAYER_ID')
 NameError: name 'df18_players' is not defined
pd.merge(df18_players,df19_players,how='outer',on='PLAYER_ID')
 NameError
                                         Traceback (most recent call last)
 <ipython-input-5-5e596036c586> in <module>
 ---> 1 pd.merge(df18_players,df19_players,how='outer',on='PLAYER_ID')
 NameError: name 'df18_players' is not defined
pd.merge(df18_players,df19_players,how='left',on='PLAYER_ID')
 NameError
                                          Traceback (most recent call last)
 <ipython-input-6-07bd548ea943> in <module>
 ----> 1 pd.merge(df18_players,df19_players,how='left',on='PLAYER_ID')
 NameError: name 'df18_players' is not defined
```

```
df18_games = pd.read_csv('data/2018-games.csv')
df19_games = pd.read_csv('data/2019-games.csv')
```

```
FileNotFoundError
                                          Traceback (most recent call last)
 <ipython-input-7-5fc4302bda75> in <module>
 ----> 1 df18_games = pd.read_csv('data/2018-games.csv')
       2 df19 games = pd.read csv('data/2019-games.csv')
 /opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
 packages/pandas/io/parsers.py in read_csv(filepath_or_buffer, sep, delimiter, header,
 names, index_col, usecols, squeeze, prefix, mangle_dupe_cols, dtype, engine,
 converters, true_values, false_values, skipinitialspace, skiprows, skipfooter, nrows,
 na_values, keep_default_na, na_filter, verbose, skip_blank_lines, parse_dates,
 infer_datetime_format, keep_date_col, date_parser, dayfirst, cache_dates, iterator,
 chunksize, compression, thousands, decimal, lineterminator, quotechar, quoting,
 doublequote, escapechar, comment, encoding, dialect, error_bad_lines, warn_bad_lines,
 delim_whitespace, low_memory, memory_map, float_precision)
     684
     685
 --> 686
             return _read(filepath_or_buffer, kwds)
     687
     688
 /opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
 packages/pandas/io/parsers.py in _read(filepath_or_buffer, kwds)
     450
     451
             # Create the parser.
 --> 452
             parser = TextFileReader(fp_or_buf, **kwds)
     453
     454
             if chunksize or iterator:
 /opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
 packages/pandas/io/parsers.py in __init__(self, f, engine, **kwds)
     944
                     self.options["has_index_names"] = kwds["has_index_names"]
     945
 --> 946
                 self. make engine(self.engine)
     947
     948
             def close(self):
 /opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
 packages/pandas/io/parsers.py in _make_engine(self, engine)
    1176
             def _make_engine(self, engine="c"):
    1177
                if engine == "c":
 -> 1178
                    self._engine = CParserWrapper(self.f, **self.options)
    1179
                 else:
                     if engine == "python":
    1180
 /opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
 2007
 -> 2008
                 self._reader = parsers.TextReader(src, **kwds)
    2009
                 self.unnamed_cols = self._reader.unnamed_cols
    2010
 pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader.__cinit__()
 pandas/ libs/parsers.pyx in pandas. libs.parsers.TextReader. setup parser source()
 FileNotFoundError: [Errno 2] No such file or directory: 'data/2018-games.csv'
games_df = pd.concat([df18_games,df19_games])
games_df.head()
 NameError
                                          Traceback (most recent call last)
 <ipython-input-8-d6ac2a3c67d8> in <module>
 ---> 1 games_df = pd.concat([df18_games,df19_games])
       2 games df.head()
 NameError: name 'df18_games' is not defined
```

Data

```
import sqlite3

con = sqlite3.connect('data/SQL_SAFI.sqlite')

cur = con.cursor()
cur.execute("SELECT * FROM Farms")
```

```
OperationalError
                                           Traceback (most recent call last)
 <ipython-input-11-9e9dd793f26c> in <module>
       1 cur = con.cursor()
 ----> 2 cur.execute("SELECT * FROM Farms")
 OperationalError: no such table: Farms
rows = cur.fetchall()
rows
 []
type(rows)
 list
type(rows[0])
                                          Traceback (most recent call last)
 IndexError
 <ipython-input-14-f2835a2e0aff> in <module>
 ----> 1 type(rows[0])
 IndexError: list index out of range
df = pd.read_sql_query("SELECT * FROM Farms",con)
 OperationalError
                                           Traceback (most recent call last)
 /opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/sql.py in
 execute(self, *args, **kwargs)
    1680
                 try:
 -> 1681
                     cur.execute(*args, **kwargs)
    1682
                     return cur
 OperationalError: no such table: Farms
 The above exception was the direct cause of the following exception:
 DatabaseError
                                           Traceback (most recent call last)
 <ipython-input-15-70a84586ef73> in <module>
 ----> 1 df = pd.read_sql_query("SELECT * FROM Farms",con)
 /opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/sql.py in
 read_sql_query(sql, con, index_col, coerce_float, params, parse_dates, chunksize)
     381
                 coerce_float=coerce_float,
     382
                 parse_dates=parse_dates,
 --> 383
                 chunksize=chunksize,
     384
             )
     385
 /opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/sql.py in
 read_query(self, sql, index_col, coerce_float, params, parse_dates, chunksize)
    1725
    1726
                 args = convert params(sql, params)
 -> 1727
                 cursor = self.execute(*args)
    1728
                 columns = [col_desc[0] for col_desc in cursor.description]
 /opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/sql.py in
 execute(self, *args, **kwargs)
                     ex = DatabaseError(f"Execution failed on sql '{args[0]}': {exc}")
    1692
 -> 1693
                     raise ex from exc
    1694
    1695
             @staticmethod
```

df.head()

DatabaseError: Execution failed on sql 'SELECT * FROM Farms': no such table: Farms

```
NameError Traceback (most recent call last)
<ipython-input-16-c42a15b2c7cf> in <module>
----> 1 df.head()

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

```
OperationalError
                                          Traceback (most recent call last)
/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/sql.py in
execute(self, *args, **kwargs)
  1680
               try:
                   cur.execute(*args, **kwargs)
-> 1681
  1682
                    return cur
OperationalError: no such table: Farms
The above exception was the direct cause of the following exception:
                                          Traceback (most recent call last)
<ipython-input-17-ce026a564156> in <module>
----> 1 location_df = pd.read_sql_query("SELECT Country, A06_Province, A07_district,
A08_ward, A09_village FROM Farms",con)
      2 location_df.head()
/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/sql.py in
read_sql_query(sql, con, index_col, coerce_float, params, parse_dates, chunksize)
               coerce_float=coerce_float,
   381
    382
                parse_dates=parse_dates,
--> 383
                chunksize=chunksize,
   384
           )
   385
/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/sql.py in
read_query(self, sql, index_col, coerce_float, params, parse_dates, chunksize)
  1725
  1726
                args = _convert_params(sql, params)
-> 1727
                cursor = self.execute(*args)
  1728
                columns = [col_desc[0] for col_desc in cursor.description]
/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/sql.py in
execute(self, *args, **kwargs)
                    ex = DatabaseError(f"Execution failed on sql '{args[0]}': {exc}")
   1692
-> 1693
                    raise ex from exc
  1694
   1695
            @staticmethod
DatabaseError: Execution failed on sql 'SELECT Country, A06 Province, A07 district,
A08_ward, A09_village FROM Farms': no such table: Farms
```

```
pd.read_sql_query("SELECT * FROM Farms LIMIT 10",con)
```

```
OperationalError
                                         Traceback (most recent call last)
/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/sql.py in
execute(self, *args, **kwargs)
  1680
               try:
-> 1681
                    cur.execute(*args, **kwargs)
  1682
                    return cur
OperationalError: no such table: Farms
The above exception was the direct cause of the following exception:
                                          Traceback (most recent call last)
DatabaseError
<ipython-input-18-f0266d39537f> in <module>
----> 1 pd.read_sql_query("SELECT * FROM Farms LIMIT 10",con)
/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/sql.py in
read_sql_query(sql, con, index_col, coerce_float, params, parse_dates, chunksize)
    381
                coerce_float=coerce_float,
    382
                parse_dates=parse_dates,
--> 383
                chunksize=chunksize,
   384
           )
   385
/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/sql.py in
read_query(self, sql, index_col, coerce_float, params, parse_dates, chunksize)
  1726
               args = _convert_params(sql, params)
               cursor = self.execute(*args)
-> 1727
                columns = [col_desc[0] for col_desc in cursor.description]
  1728
  1729
/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/sql.py in
execute(self, *args, **kwargs)
  1691
  1692
                    ex = DatabaseError(f"Execution failed on sql '{args[0]}': {exc}")
-> 1693
                    raise ex from exc
  1694
  1695
           @staticmethod
DatabaseError: Execution failed on sql 'SELECT * FROM Farms LIMIT 10': no such table:
Farms
```

```
food_df = pd.read_sql_query("SELECT G01_no_meals, G02_months_lack_food,
G03_no_food_mitigation FROM Farms LIMIT 5",con)
```

```
OperationalError
                                           Traceback (most recent call last)
 /opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/sql.py in
 execute(self, *args, **kwargs)
   1680
                 try:
 -> 1681
                     cur.execute(*args, **kwargs)
    1682
                     return cur
 OperationalError: no such table: Farms
 The above exception was the direct cause of the following exception:
                                           Traceback (most recent call last)
 DatabaseError
 <ipython-input-19-d3042728be92> in <module>
 ----> 1 food_df = pd.read_sql_query("SELECT G01_no_meals, G02_months_lack_food,
 G03_no_food_mitigation FROM Farms LIMIT 5",con)
 /opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/sql.py in
 read_sql_query(sql, con, index_col, coerce_float, params, parse_dates, chunksize)
     381
                 coerce_float=coerce_float,
     382
                 parse_dates=parse_dates,
 --> 383
                 chunksize=chunksize,
     384
             )
     385
 /opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/sql.py in
 read_query(self, sql, index_col, coerce_float, params, parse_dates, chunksize)
    1725
    1726
                 args = _convert_params(sql, params)
 -> 1727
                 cursor = self.execute(*args)
    1728
                 columns = [col_desc[0] for col_desc in cursor.description]
    1729
 /opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/sql.py in
 execute(self, *args, **kwargs)
    1692
                     ex = DatabaseError(f"Execution failed on sql '{args[0]}': {exc}")
 -> 1693
                     raise ex from exc
    1694
    1695
             @staticmethod
 DatabaseError: Execution failed on sql 'SELECT G01_no_meals, G02_months_lack_food,
 G03_no_food_mitigation FROM Farms LIMIT 5': no such table: Farms
food_df
                                          Traceback (most recent call last)
 <ipython-input-20-864faa46c3c2> in <module>
 ----> 1 food_df
```

```
NameError: name 'food df' is not defined
```

```
parents_df = pd.read_sql_query("SELECT Id, B17_parents_liv FROM Farms WHERE
B17_parents_liv = 'yes'",con)
parents_df.head()
```

```
OperationalError
                                          Traceback (most recent call last)
/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/sql.py in
execute(self, *args, **kwargs)
  1680
               try:
-> 1681
                    cur.execute(*args, **kwargs)
  1682
                    return cur
OperationalError: no such table: Farms
The above exception was the direct cause of the following exception:
DatabaseError
                                          Traceback (most recent call last)
<ipython-input-21-4b3495032050> in <module>
----> 1 parents_df = pd.read_sql_query("SELECT Id, B17_parents_liv FROM Farms WHERE
B17_parents_liv = 'yes'",con)
      2 parents_df.head()
/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/sql.py in
read_sql_query(sql, con, index_col, coerce_float, params, parse_dates, chunksize)
   381
               coerce_float=coerce_float,
   382
               parse_dates=parse_dates,
--> 383
               chunksize=chunksize,
   384
           )
   385
/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/sql.py in
read_query(self, sql, index_col, coerce_float, params, parse_dates, chunksize)
  1725
  1726
               args = _convert_params(sql, params)
-> 1727
               cursor = self.execute(*args)
  1728
               columns = [col_desc[0] for col_desc in cursor.description]
  1729
/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/sql.py in
execute(self, *args, **kwargs)
  1692
                    ex = DatabaseError(f"Execution failed on sql '{args[0]}': {exc}")
-> 1693
                    raise ex from exc
  1694
  1695
           @staticmethod
DatabaseError: Execution failed on sql 'SELECT Id, B17_parents_liv FROM Farms WHERE
B17_parents_liv = 'yes'': no such table: Farms
```

Assignments

All assignments are due on Sunday at 11:59pm, via github unless otherwise noted.

Assignment TOC:

- Assignment 1 Due September 13
- Assignment 2 Due September 20
- Assignment 3 Due September 29
- Assignment 4 Due October 4
- Assignment 5 Due October 11

Assignment 1: Portfolio Setup, Data Science, and Python

Due: 2020-09-13

Objective & Evaluation

This assignment is an opportunity to earn level 2 achievements for the process and python and confirm that you have all of your tools setup, including your portfolio.

To Do

Your task is to:

1. Install required software

- Setup your portfolio, by <u>accepting the assignment</u> and following the instructions in the README file on your repository.
- 3. Add your own definition of data science to the introduction of your portfolio, in about/index.md
- 4. Add a Jupyter notebook called grading. ipynb to the about folder and write a function that computes a grade for this course, with the following docstring. Include:
- · 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. Uncomment the line # file: about/grading in your_toc.yml file.

```
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 possible modifier (+/-)
```

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+'
```

Submission Instructions

Create a Jupyter Notebook with your function and Add the notebook to your portfolio by uploading it to your repository, or adding to the folder off line and committing and pushing the changes.

View the gh-pages branch to see your compiled submission, as portfolio.pdf or by viewing your website.

There will be a pull request on your repository that is made by GitHub classroom, $\frac{\text{request a review}}{\text{reduest on your repository}} from the team <math display="block">\frac{\text{reduest a review}}{\text{reduest on your repository}} from the team <math display="block">\frac{\text{reduest a review}}{\text{reduest on your repository}} from the team <math display="block">\frac{\text{reduest a review}}{\text{reduest on your repository}} from the team <math display="block">\frac{\text{reduest a review}}{\text{reduest on your repository}} from the team <math display="block">\frac{\text{reduest a review}}{\text{reduest on your repository}} from the team <math display="block">\frac{\text{reduest a review}}{\text{reduest on your repository}} from the team <math display="block">\frac{\text{reduest a review}}{\text{reduest on your repository}} from the team <math display="block">\frac{\text{reduest a review}}{\text{reduest on your repository}} from the team <math display="block">\frac{\text{reduest a review}}{\text{reduest on your repository}} from the team <math display="block">\frac{\text{reduest a review}}{\text{reduest on your repository}} from the team <math display="block">\frac{\text{reduest on your repository}}{\text{reduest on your repository}} from the team <math display="block">\frac{\text{reduest on your repository}}{\text{reduest on your repository}} from the team <math display="block">\frac{\text{reduest on your repository}}{\text{reduest on your repository}} from the team <math display="block">\frac{\text{reduest on your repository}}{\text{reduest on your repository}} from the team <math display="block">\frac{\text{reduest on your repository}}{\text{reduest on your repository}} from the team <math display="block">\frac{\text{reduest on your repository}}{\text{reduest on your repository}} from the team <math display="block">\frac{\text{reduest on your repository}}{\text{reduest on your repository}} from the team <math display="block">\frac{\text{reduest on your repository}}{\text{reduest on your repository}} from the team <math display="block">\frac{\text{reduest on your repository}}{\text{reduest on your repository}} from the team <math display="block">\frac{\text{reduest on your repository}}{\text{reduest on your repository}} from the team <math display="block">\frac{\text{reduest on your repository}}{\text{reduest on your repository}} from the team <math display="block">\frac{\text{reduest on your repository}}{\text{reduest on your repository}} from the team of t$

Solutions

One solution is added to the **Detailed Mechanics** part of the Grading section of the syllabus.

Assignment 2: Practicing Python and Accessing Data

Due: 2020-09-20

Objective & Evaluation

This assignment is an opportunity to earn level 1 or 2 achievements in python, process and access and begin working toward level 1 in summarize.

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/fall20instructors.

To do this click Issues at the top, the green "New Issue" button and then type away.

Warning

your function can have a different name than compute_grade, but make sure it's your function name, with those parameter values in your tests.



when the value of the expression after assert is True, it will look like nothing happened. assert is used for testing Accept the assignment on <u>GitHub Classroom</u>. It contains a notebook with some template structure (and will set you up for grading). The template will also convert notebooks that are added to markdown, which makes reading on GitHub for easier grading. If you want to incorporate feedback you receive back into a notebook file, <u>Jupytext</u> can do that.

To work with this notebook you can either:

- download the repository as .zip from the green code button, unzip, and re-upload, OR
- clone the repository with git and the push your changes. See Git/GitHub help on cloning, committing, and pushing, for
 example this <u>tutorial on git</u> to learn more about git.

Accessing Data with Python and pandas

(for python and access)

Find 3 datasets of interest to you that are provided in 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. Complete a dictionary for each with the url, a name, and what function should be used to load the data into a pandas. DataFrame.

Use a list of those dictionaries to iterate over the datasets and build a table that describes them, with the following columns ['name', 'source', 'num_rows', 'num_columns', 'source_file_name']. The source column should be the url where you loaded the data from or the source if you downloaded it from a website first The source_file_name should be the part of the url after the last /, you should extract this programmatically. Display that summary table as a dataframe and save it as a csv, named dataset_summary.csv.



For one dataset (must include nonnumerical data):

- display the heading with the last seven rows
- make and display a new data frame with only the non numerical columns
- was the format that the data was provided in a good format? why or why not?



For a second dataset:

- display the heading and 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)?

For the third dataset:

• display the first 5 even rows of the data for three columns of your choice

For any dataset:

• try reading it in with the wrong read_function. If you had done this by accident, how could you tell?

Data Science Process

(for the process skill)

Make a list of a data science pipeline and denote which types of programming might be helpful at each staged. Include this in a markdown cell in the same notebook with your analysis.

🥊 Tip

Urls are strings. The string class in python has a lot of helpful methods for manipulating strings, like split.

Note

If you download the datasets (or find them as .zip and need to) you can use the local path instead of the url, but include a markdown cell with links to where you got your data from.

9 Tip

You can create a pandas
DataFrame using the
constructor and you can build
lists (or lists of lists) using the
append method

Tip

Remember that this will be graded based on the rubric and that it should reflect your understanding, not be simply copied from a source. Also, always cite your sources, informal linking is ok.

To make a link in markdown

[text to display](http://url.com/of/the/site/your/are/linking/)

Assignment 3: Exploratory Data Analysis

Due: 2020-09-27

Objective & Evaluation

This assignment is an opportunity to earn level 1 or 2 achievements in summarize, visualize, or access. You can earn level 2 in python.

Accept the assignment on GitHub Classroom. The template will convert notebooks that are added to markdown, which makes reading on GitHub for easier grading. It will sync between .ipynb and .md style notebooks stored in your repository.

This week I encourage you to try working with git, but if you're not comfortable with that you can work via upload again.

Exploratory Data Analysis

This week your goal is to do a small exploratory data analysis for two datasets of your choice. One dataset must include at least two continuous valued variables and at least one categorical variable(d1). One dataset must include at least two categorical variables and at least one continuous valued variable(d2).

Use a separate notebook for each dataset, name them dataset_01.ipynb and dataset_02.ipynb.

For each dataset:

- 1. Include a markdown header with a title for your analysis
- 2. Load the data to a notebook as a DataFrame from url.
- 3. Explore the dataset in a notebook enough to describe its structure
 - o shape
 - o columns
 - o variable types
- 4. Write a short description of what the data contains and what it could be used for
- 5. Complete an exploratory analysis with statistics and plots. Your analysis should include markdown cells describing the results you see, not only what you did. Your analysis should be structured to follow the steps below, for the corresponding dataset.

For d1:

- 1. Display all of the summary statistics for a subset of 5 of your choice or all variables if there are fewer than 5 numerical values
- 2. Display all of summary statistics grouped by a categorical variable
- 3. For two continuous variables make a scatter plot and color the points by a categorical variable
- 4. Pose one question for this dataset that can be answered with summary statistics, compute a statistic and plot that help answer that exploratory question.

For d2:

- 1. Display two individual summary statistics for one variable
- 2. Group the data by two categorical variables and display a table of one summary statistic
- 3. Use a seaborn plotting function with the col parameter or a FacetGrid to make a plot that shows something informative about this data, using both categorical variables and at least one numerical value. Describe what this tells you about the data.
- ${\bf 4. \, Produce \, one \, additional \, plot \, of \, a \, different \, plot \, type \, that \, shows \, something \, about \, this \, data.}$

🥊 Tip

A continuous valued variable is one that can take on infinitely many values. Examples include temperatue, length, age, etc. A categorical value is one that can only take on one of a subset of specific values. These require a bit more context to define often. For example color could be recorded in a basically continuous way as RGB values in HEX, but color of the stoplight will only be red, green, or yellow. Town of birth in a dataset of celebrities would not be a good categorical variable, there might not be many duplicates to do analysis with, but metro area of current residence for celebrities would be because we'd see mostly major cities and could do analyses.

if you use a local copy of the dataset you can load from a <u>relative path</u> instead of a url. Please include the data in a data folder that's in the folder where your notebook is so that the path is something like: data/best-datasetever. csv and upload the data to your assignment repository

In total, in each of your notebooks you will have:

- · a loaded dataset
- a basic description
- at least two summary statistic calculations
- at least two plots

Assignment 4: Preparing Data for Analysis

Due: 2020-10-04

Objective & Evaluation

You can earn prepare level 2 by cleaning two datasets as outlined in the cs_degrees.ipynb and travel_time.ipynb notebooks.

To earn access level 2 you must clean a third dataset, following the airlines.ipynb or safi_full.ipynb notebooks. Note that this is the SAFI dataset we've been working with in class, but a messier version of it. You know what some of the fixes are because they're in the notes and some of what it will look like when it's done.

To earn python level 2, you must complete safi_full.ipynb. Specifically, you must successfully use list or dictionary comprehensions and conditional statements, beyond those provided as hints.

To earn level 2 for summarize and visualize, include additional analyses after cleaning the datasets. You'll need at least two types of plots and two different ways of using summary statistics and to interpret the results.

Instructions

For this assignment, your assignment is to clean datasets in notebooks and include narrative description of how you're making decisions about the data cleaning. At the end of each cleaning, save the cleaned dataset to the data folder with an informative file name.

When you <u>accept the assignment</u>, there will be template notebooks in the repository. These have significant hints to guide your efforts.

There is an issue type with a todo list, try that out to plan and track questions you have.

You can work with this assignment in 2 ways:

Git Workflow

- 1. Click the green code button and copy the url.
- 2. clone the repository as below where the url is the one you copied from GitHub. In your terminal on Linux or Mac or on the GitBash on Windows (<u>install instructions on tools section of syllabus</u>)

cd path/where/you/want/a/new/folder/for/this/assignment
git clone http://github.com/rhodyprog4ds/04-prepare-username

1. then launch your notebook from the newly created folder.

on Linux or Mac, in the same terminal (remember you can use tab complete)

cd 04-prepare-username
jupyter notebook

or on windows, in your anaconda prompt

cd path/where/you/want/a/new/folder/for/this/assignment/04-prepare-username jupyter notebook

- 1. work on your assignment, by opening the notebooks there and editing them.
- 2. commit your changes when you want to save a point in your progress. Either you have part workign and want to save that, or you want feedback, or you're done. On whichever terminal you used for the git clone command above

9 Tip

Recall how we used a single function to see many statistics in class, use that for dataset 1. For dataset 2, you can make your own combination of summary statistics with pandas. DataFrame .agg

🥊 Tip

You can download any of the files and open them with another program if you need to by pasting the url in your browser. You'll need to display relevant parts in a notebook for grading, but if looking another way helps, you can try that.

```
git add .
git commit -m 'description of current status of project'
```

1. push your changes when you want to share them on GitHub, (eg need help, want feedback or complete)

```
git push
```

1. if you will keep working after pushing, first pull down the .md conversion that was added by GitHub actions. If you don't you'll have to merge, it should be fine, but ask if you're not sure.

git pull

Download and upload workflow

- 1. Click the green code button
- 2. choose the download via zip option
- 3. unzip
- 4. launch a notebook in the folder where you unzipped
- 5. Upload only the files you changed into the repository to replace their previous versions with the add file via upload button on GitHub. Put the notebooks at the top level and the data files in the data folder.

Tips and hints

- Pandas can convert string to dates to be able to work with them more flexibly
- With python date objects you can get the day of the week from a date
- Pandas <u>read_excel</u> has:
 - o a header parameter can take a list as its value
 - o two parameters that allow you to skip rows when you read in the data
- list(range(7)) check out what this line does
- Pandas drop can work a lot of different ways

The goal of the plot for the CS degrees dataset is:

Degrees in computer and information sciences conferred by degree-granting institutions, by level of degree and sex of student: 1970-71 through 2010-11



Assignment 5: Constructing Datasets and Using Databases

Due: 2020-10-11

 $there\ will\ be\ no\ tolerance\ of\ late\ submissions\ on\ this\ assignment\ because\ it\ will\ be\ graded\ Monday,\ so\ that\ you\ can\ use\ that\ feedback\ to\ finish\ up\ your\ portfolio.$

Accept the assignment

Instructions

this will earn level 2 for construct

Your goal is to build and prepare two ready to analyze datasets. You will submit a notebook that describes how you built and prepared each dataset.

- Each finished dataset must be produced from two or more tables.
- At least one must come from an sqlite database, either by merging results from multiple queries or multiple tables.
- You should use at least two different merges and one concatenate

Your completed datasets should have:

- column names that are well formatted (only lowercase letters, numbers and _)
- an added column that is derived from one or more other columns (string operation or calculation)

For each dataset, pose one question that could not be answered from the input data files as provided and demonstrate how to answer it with the dataset you built. This could be something that can be answered with using only shape of the merged data, but if you need summarize and visualize level 2 achievements, you should use more statistics and plots.

Your notebooks must be in the top level of the repository, not in a subfolder.

Additional Achievements:

if you already earned prior achievements you can ignore the following

To earn level 2 for prepare, one of your analyses must use datasets with missing values and one must be provided as excel files with merged columns (for example from NCES). You may use one dataset with both merged columns and missing data or one of each. You must also use datasets that have column names that need repair.

For the merged column data, either before or after merging, you must additionally:

- create separate separate tables for original and aggregate values (eg percentages or sums that can be recovered from the other columns)
- unstack all levels of the data to create a single level index over the columns

To earn level 2 for summarize and/or visualize, include additional analyses after building the datasets. You'll need at least two types of plots for visualize and/or two different ways of using summary statistics for summarize and to interpret the results for either.

Portfolio

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 <u>syllabus</u>. 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 <u>formatting tips</u> for advice on how to structure files.

The first submission will be graded on the following criteria and due on October 14:

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 g	generate complex plots with pandas and plotting libraries and customize with matplotlib
prepare	apply data reshaping, cleaning, and filtering manipulations reliably and correctly by assessing data as received

On each chapter of your portfolio, you should identify which skills by their keyword, you are applying.

Formatting Tips

Your portfolio is a jupyter book. This means a few things:

- it uses myst markdown
- it will run and compile Jupyter notebooks

This page will cover a few basic tips.

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.

Structure of plain markdown

Use a heading like this:

```
# Heading of page
```

in the file and it will appear in the sidebar.

File Naming

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

To sync feedback received to your runnable notebook files, change the related GitHub Actions file: .github/workflows/In the step named convert that looks like:

```
- name: convert
run: |
    jupytext */*.ipynb --to myst
```

change it to:

```
- name: convert
run: |
    jupytext --set-formats ipynb,md */*.ipynb # Turn .ipynb into a paired ipynb/py notebook
    jupytext --sync */*.ipynb # Update whichever of .ipynb/notebook.md is
outdated
```

This means if you accept suggestion commits from the the .md file, the action will upate your .ipynb file. If you update your .ipynb file the action will update the .md file.

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 <u>text editor</u> 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.

Configurations

Things like the menus and links at the top are controlled as settings, in _config.yml

Show errors and continue

To show errors and continue running the rest, add the following:

```
# Execution settings
execute:
  allow_errors : true
```

Links

Markdown syntax for links

```
[text to show](path/or/url)
```

Reflective Prompts

These prompts are more reflective to help demonstrate your understanding of skills. These are more writing than new coding.

Correcting a Prior Assignment

Choose an assignment that you did not achieve the target level for. Write a blog style notebook analysis that corrects what you could have done better, what you learned, and addresses the misconception if applicable.

Data Science Pipeline

Like the day 1 activity, find two different sources that describe the data science pipeline or lifecycle. Write a blog style post that discusses their differences and hypothesizes about why they may be different? Are they for different audiences? Is one domain specific.

Podcast

Watch an episode of a high quality [1] podcast and write a blog style summary and review of the episode. Highlight what you learned and how it relates to things

Approved Podcasts:

• Pod of Asclepius, Fall Series: The Philosophy of Data Science

Annotate Class Notes

Annotate a whole session of class notes by submitting a PR to this repository. This applies after Dr. Brown's notes are uploaded, but not annotated. Use previously annotated notes as an example of what content should be added. In your portfolio, include a 1 paragraph reflection on what you learned by contributing the annotation and a link to your pull request.

[1] approved by Dr. Brown by creating a pull request to add it to the list on this page that is successfully merged. To create a PR, use the suggest an edit button at the top of this page.

Analysis Prompts

Loading Data

Look at all of the different ways pandas can load data. Consider some of the questions below and add a notebook that's styled like a report that answers a few of them with text and code.

- Which seem like good idea? are any dangerous?
- Which seem more or less common?
- Can you compare them on speed? Is it ever worth transforming a dataset before loading?
- How much can you repair a dataset using the parameters of the load functions?

CheatSheet

Make a cheatsheet with examples of the several different parameter settings for common operations for one topic.

This cheatsheet is an example, it's too broad, but it's the same idea. Yours should

Deeper Analysis

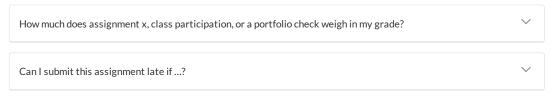
For one of the assignments, if there was something you were curious about. Try it out and investigate how to answer it. Vary parameters and document your investigation.

FAQ

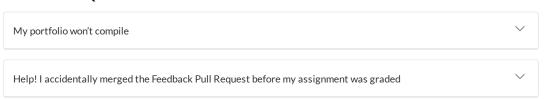
This section will grow as questions are asked and new content is introduced to the site. You can submit questions:

- via e-mail to Dr. Brown (brownsarahm) or Beibhinn (beibhinn)
- via Prismia.chat during class
- by creating an issue

Syllabus FAQ



GitHub FAQ



Resources

Note

It's okay if more than one person contributes annotations on the same day. If there's a pull request there already, try to add different, additional insights but if it's not there when you start working and appears before you submit, that's ok.

This section will compile resources for the course over time into various sections:

- <u>Data</u> sources of data to use for assignments
- Programming info on the tools and libraries we're using in class, background info, etc
- Reference example tips
- <u>Tips</u> general, semi-related information

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

References on Python

• Course Text

Data Sources

- Kaggle
- Google Dataset Search
- UCI Data Repository
- Json Datasets
- <u>Databases</u>

If you have others please share by creating a pull request or issue on this repo (from the GitHub logo at the top right).

By Professor Sarah M Brown © Copyright 2020.