# 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.

# **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 <a href="mailto:rhodyprog4ds.github.io/BrownFall20/syllabus">rhodyprog4ds.github.io/BrownFall20/syllabus</a>. 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 <a href="mailto:repository">repository</a>. You can view the date of changes and exactly what changes were made on the Github <a href="mailto:commits">commits</a> page.

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

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 Brightspace site. 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 Zoom client on your computer. Please log in and configure your account. Please add a photo of yourself to your account so that we can still see your likeness in some form when your camera is off. You may also wish to use a virtual background and you are welcome to do so.

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

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

### 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

### Important

### TL:DR [1]

- check Brightspace
- Install Zoom
- Setup your URI Zoom Account
- Log in to Prismia Chat
- Make a GitHub Account
- Install Python
- · Install Git

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 ot 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.

#### 0 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.



🛕 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.

# **Detailed mechanics**

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

Level 3	Level 2	Level 1
15	15	15
10	15	15
5	15	15
0	15	15
0	10	15
0	5	15
0	0	15
0	0	10
0	0	5
0	0	3
	15 10 5 0 0 0 0	10 15 5 15 0 15 0 10 0 5 0 0 0 0

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



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

```
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'
        grade = 'F'
    return grade
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

### 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.



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







### Other Activities



A12 Not submitted

13 14 Attended, but all level 1 complete

Attended, but all level 1 complete

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

# Map to a B easily







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





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
- 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.

	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	${\sf Modeling, Naive\ Bayes, classification\ performance\ metrics}$	[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.

#### Note

On the BrightSpace calendar 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.

	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 stge 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.

	<b>A1</b>	<b>A2</b>	А3	<b>A4</b>	<b>A</b> 5	<b>A6</b>	<b>A7</b>	<b>A8</b>	A9	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	0	0	1	1	1	1	1	1	1	1	9
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 <a href="URI Microsoft 365">URI Microsoft 365</a> 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
- 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

# 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

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

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" None so there is no output from this cell

```
a = 9
```

But this one does display the value of the cell

```
a

9

b = 4
b

4

a
```

# 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

```
Docstring:
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')
```

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
```

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

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:



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

```
greeting('sarah')

hello sarah
```

A better version of that function might be:

• Tip
you can append strings
with +

# 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

# Grading and Assignment 1

- Solution function posted.
- note: not a sum
- read the rubric
- Brightspace will show grades as they're earned
- In class, respond on prismia
- Portfolio
  - 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

remember python indexes from 0

Indexing with a negative number counts from the end

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

Loops in python have similar syntax to the  ${\tt if}$  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
```

```
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.

```
pandas.read_csv()
```

```
TypeError Traceback (most recent call last)
<ipython-input-14-374ala6f9f7e> in <module>
----> 1 pandas.read_csv()

TypeError: read_csv() missing 1 required positional argument: 'filepath_or_buffer'
```

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

```
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')



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

### 🥊 Tip

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
                     1
                                                    [admin, python review]
1
                     2
                                              Loading data, Python review
                                                Exploratory Data Analysis
2
                     3
3
                     4
                                                            Data Cleaning
4
                     5
                                            Databases, Merging DataFrames
5
                        Modeling, Naive Bayes, classification performa...
6
                                         decision trees, cross validation
7
                     8
                                                                Regression
                     9
8
                                                                Clustering
9
                    10
                                                     SVM, parameter tuning
                                                     KNN, Model comparison
                    11
11
                    12
                                                             Text Analysis
                                                            Topic Modelina
                    13
12
13
                    14
                                                            Deep Learning
                              skills
                  Unnamed: 2 level 1
0
                             process
1
        [access, prepare, summarize]
2
             [summarize, visualize]
3
     [prepare, summarize, visualize]
      [access, construct, summarize]
4
5
          [classification, evaluate]
6
          [classification, evaluate]
7
              [regression, evaluate]
8
              [clustering, evaluate]
9
                   [optimize, tools]
10
                    [compare, tools]
11
                      [unstructured]
12
               [unstructured, tools]
13
                   [tools, compare]
   Unnamed: 0_level_0
                                                                     skill \
               keyword
                                                       Unnamed: 1_level_1
               python
                                                    pythonic code writing
1
               process
                                       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
6
7
              prepare
                                                 prepare data for analysis
                                                      Apply classification
       classification
8
           regression
                                                          Apply Regression
9
            clustering
                                                                Clustering
                                                Evaluate model performance
              evaluate
11
             optimize
                                                Optimize model parameters
12
               compare
                                                           compare models
                                                  model unstructured data
13
         unstructured
14
                 tools use industry standard data science tools and w...
                                               Level 1
                                    Unnamed: 2_level_1
     python code that mostly runs, occasional pep8 ...
             Identify basic components of data science
1
2
     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 ...
     identify plot types, generate basic plots from...
     identify if data is or is not ready for analys...
    identify and describe what classification is, ...
8
    identify what data that can be used for regres...
                          describe what clustering is
    Explain basic performance metrics for differen...
    Identify when model parameters need to be opti...
                   Qualitatively compare model classes
    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 stge of the data scie...
     Load data for processing from the most common \dots
                                    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
                      can fit linear regression models
                                apply basic clustering
    Apply basic model evaluation metrics to a held...
    Manually optimize basic model parameters such ...
    Compare model classes in specific terms and fi...
    Apply at least one representation to transform...
13
    Solve semi-strucutred, completely specified pr...
```

```
Level 3
                                    Unnamed: 4_level_1
    reliable, efficient, pythonic code that consis...
    Compare different ways that data science can o...
    access data from both common and uncommon form...
         merge data that is not automatically aligned
    Compute and interpret various summary statisti...
    generate complex plots with pandas and plottin...
    apply data reshaping, cleaning, and filtering ...
    fit and apply classification models and select...
             can fit and explain nonlinear regression
    apply multiple clustering techniques, and inte...
    Evaluate a model with multiple metrics and cro...
    Select optimal parameters based of mutiple qua...
    Evaluate tradeoffs between different model com...
    apply mulitple representations and compare and \ldots
    Scope, choose an appropriate tool pipeline and...
   Unnamed: 0_level_0
                                    A1
              keyword Unnamed: 1_level_1 Unnamed: 2_level_1
               python
1
              process
2
               access
3
            construct
4
            summarize
                                                           0
            visualize
              prepare
7
       {\tt classification}
           regression
9
                                                           0
           clustering
                                                           0
10
             evaluate
11
             optimize
12
              compare
                                                           0
         unstructured
13
                                                           0
14
                tools
                                                          Α5
                   АЗ
                                       Α4
   Unnamed: 3_level_1 Unnamed: 4_level_1 Unnamed: 5_level_1
                    0
                                        0
1
2
                    1
                                                           0
                    0
                    1
                    0
                                                           1
                    0
                                                           0
                    0
                                                           0
10
                    0
                                                           0
11
                    0
12
                    0
                                                           0
                   Α6
                                       Α7
                                                          A8
   Unnamed: 6_level_1 Unnamed: 7_level_1 Unnamed: 8_level_1
                    0
                    0
                                        0
1
                    0
                                        0
                                                           0
3
6
                    0
                                                           0
                    1
                                                           0
                    0
                                                           1
                    0
                                                           0
10
                    0
                                                           0
11
                                                           0
12
                    0
13
                    0
14
                                       A10
                   Α9
                                                           A11
   Unnamed: 9_level_1 Unnamed: 10_level_1 Unnamed: 11_level_1
                    0
                                         0
                    0
                                         0
1
2
                    0
                                         0
3
                    0
                                         0
                    1
                                         1
6
                    0
                    0
                                         1
8
                    0
                    1
                                         0
                    0
11
                                         1
                                                             1
12
                    0
                                         0
                                                             1
```

```
14
                    0
                                         1
                                                              1
                   A12
                                        A13
                                                  # Assignments
   Unnamed: 12_level_1
                       Unnamed: 13_level_1 Unnamed: 14_level_1
                                          0
                     0
                                                               2
1
                                          0
2
                                                               3
                     0
                                          0
3
                     0
                                          0
                                                               2
4
                                                              11
                                          1
5
                     1
                                          1
                     0
                                                               2
6
                                          0
                                                               3
                     0
                                          0
8
                     0
                                          0
9
                     0
                                          0
                     0
10
                                          0
                                                               2
11
                     0
                                          0
12
                     0
                                          1
                                                               2
13
14
                                          1
   Unnamed: 0_level_0
                                                                   Level 3 \
              keyword
                                                       Unnamed: 1_level_1
0
               python
                        reliable, efficient, pythonic code that consis...
1
                       Compare different ways that data science can o...
              process
2
                       access data from both common and uncommon form...
               access
3
            construct
                             merge data that is not automatically aligned
4
            summarize
                       Compute and interpret various summary statisti...
5
            visualize
                       generate complex plots with pandas and plottin...
              prepare
                       apply data reshaping, cleaning, and filtering ...
7
       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
                       Select optimal parameters based of mutiple qua...
             optimize
                       Evaluate tradeoffs between different model com...
12
              compare
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
1
                     0
2
                    1
3
                                                            0
                                                                               0
                    1
4
                    1
                                                            0
                                                                               0
5
                    1
                                                            0
                                                                               0
                    0
                                                            1
8
                    0
                                                            1
                                                                               0
9
                    0
                                                            1
10
                    0
                                                            1
                                                                               0
                     0
11
                    0
12
                                        0
                                                            1
13
                    0
                                        0
                                                            1
                                                                               1
14
                    0
                                        0
                                                            1
                                                                               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		Modeling, Naive Bayes, classification performa	[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
- $\bullet \ \ \mathsf{you} \ \mathsf{don't} \ \mathsf{need} \ \mathsf{me} \ \mathsf{to} \ \mathsf{give} \ \mathsf{you} \ \mathsf{answers} \ \mathsf{about} \ \mathsf{how} \ \mathsf{code} \ \mathsf{works}, \mathsf{the} \ \mathsf{interpreter} \ \mathsf{will} \ \mathsf{tell} \ \mathsf{you}$
- 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	m
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-socialscifiles/master/data/SAFI\_clean.csv')
```

This is an excerpt from the **SAFI** dataset.

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)

🥊 Tip

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	memb_
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)

memb_	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

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 = pd.DataFrame({'animal': ['alligator', 'bee', 'falcon', 'lion',
...
'monkey', 'parrot', 'shark', 'whale', 'zebra']})
    >>> df
          animal
    0 alligator
              hee
    1
    2
          falcon
    3
            lion
          monkey
    5
          parrot
    6
           shark
           whale
    8
           zebra
    Viewing the first 5 lines
    >>> df.head()
          animal
    0 alligator
    1
              bee
    2
           falcon
    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()
```

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

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

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

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']
 0
        2016-11-17T00:00:00Z
        2016-11-17T00:00:00Z
 2
        2016-11-17T00:00:00Z
 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
 129
        2017-06-04T00:00:00Z
        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)
```

```
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.

```
{\tt safi\_df.dtypes}
 key_ID
                            int64
 village
                           object
                           object
 interview_date
 no_membrs
                            int64
 years_liv
                            int64
 respondent_wall_type
                          object
 rooms
                           int64
 memb_assoc
                           object
 affect_conflicts
                           object
                           int64
 liv_count
 items owned
                          object
 no_meals
                           int64
 months_lack_food
                           object
 {\tt instanceID}
                           object
 dtype: object
```

Note that it uses int64 and object as the types.

```
safi_df.head(2)
```

memb_	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

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



how many surveyed farms have all type  ${\tt mauddaub}$ ?

# 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



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)

key\_ID village interview\_date no\_membrs years\_liv respondent\_wall\_type roc 2016-11-0 0 1 God 3 4 muddaub 17T00:00:00Z 2016-11-1 1 God muddaub 17T00:00:00Z safi\_df.head(2)

 
 key\_ID
 village
 interview\_date
 no\_membrs
 years\_liv
 respondent\_wall\_type
 rooms
 memb\_

 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 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.

🥊 Tip

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.

#### key\_ID village interview\_date no\_membrs years\_liv respondent\_wall\_type rooms memb\_ 2016-11-0 1 God muddaub 1 17T00:00:00Z 2016-11-1 muddaub 1 God 1 17T00:00:00Z 2016-11-2 3 God 10 15 burntbricks 1 17T00:00:00Z

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

```
safi_df['village']
 0
             God
 1
             God
 2
             God
 3
             God
 4
             God
 126
           Ruaca
 127
           Ruaca
 128
           Ruaca
 129
        Chirodzo
 130
        Chirodzo
 Name: village, Length: 131, dtype: object
```

To index rows, we can use loc

```
safi_df.loc[3]
 key_ID
 village
                                                                        God
 interview_date
                                                       2016-11-17T00:00:00Z
 no_membrs
 years_liv
                                                                          6
 respondent_wall_type
                                                                burntbricks
 rooms
 memb assoc
                                                                        NaN
 affect_conflicts
                                                                        NaN
 liv_count
                         bicycle; radio; cow_plough; solar_panel; mobile_phone
 items_owned
 no_meals
 months_lack_food
                                                           Sept;Oct;Nov;Dec
                                  uuid:148d1105-778a-4755-aa71-281eadd4a973
 instanceID
 Name: 3, dtype: object
```

To select a range, use:

	Н
safi_df.loc[3:5]	Н
	ш

	key_ID	village	interview_date	no_membrs	years_liv	respondent_wall_type	rooms	memb_
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.loc[:4]

	key_ID	village	interview_date	no_membrs	years_liv	respondent_wall_type	rooms	memb_
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

m

	key_ID	village	interview_date	no_membrs	years_liv	respondent_wall_type	rooms	m
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 .

```
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 God 3 4 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	(

## 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
                             [admin, python review]
                         Loading data, Python review
2
                   3
                          Exploratory Data Analysis
3
                                      Data Cleaning
4
                      Databases, Merging DataFrames
                            skills
                Unnamed: 2\_level\_1
0
                           process
1
      [access, prepare, summarize]
           [summarize, visualize]
3
   [prepare, summarize, visualize]
4
    [access, construct, summarize]
   Unnamed: 0_{\text{level}}0
                                                                     skill \
              keyword
                                                       Unnamed: 1_level_1
                                               Evaluate model performance
             evaluate
11
             optimize
                                                Optimize model parameters
12
                                                           compare models
              compare
                                                  model unstructured data
13
         unstructured
14
                tools use industry standard data science tools and w...
                                               Level 1 \
                                   Unnamed: 2_level_1
    Explain basic performance metrics for differen...
    Identify when model parameters need to be opti...
                 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
    Apply basic model evaluation metrics to a held...
11
    Manually optimize basic model parameters such ...
    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...
    Evaluate tradeoffs between different model com...
    apply mulitple representations and compare and...
    Scope, choose an appropriate tool pipeline and...
   Unnamed: 0_level_0
                                      A1
              keyword Unnamed: 1_level_1 Unnamed: 2_level_1
10
             evaluate
                                        0
             optimize
11
                                        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
                                        0
11
                    0
                                        0
                                                           Θ
12
                    0
                                        0
                                                           0
13
                    0
                                        0
                                                           0
14
                   A6
                                       Α7
   Unnamed: 6_level_1 Unnamed: 7_level_1
                                         Unnamed: 8_level_1
                    0
                                        0
11
                    0
                                        0
                                                           0
                    0
                                        0
12
                                                           0
13
                    0
                                        0
                                                           0
                   Α9
                                       A10
   Unnamed: 9_level_1 Unnamed: 10_level_1 Unnamed: 11_level_1
10
                    0
11
12
                    0
                                         0
                                                              1
13
                    0
                                         0
                                                              0
14
                    0
                                         1
                                        A13
                                                  # Assignments
   Unnamed: 12_level_1 Unnamed: 13_level_1 Unnamed: 14_level_1
10
                     0
                                          0
11
                     0
                                          0
                                                               2
12
                     0
                                          1
                                                               2
13
                      1
                                          1
14
                                          1
  Unnamed: 0_level_0
                                                                   Level 3 \
                                                       Unnamed: 1_level_1
```

10 evaluate Evaluate a model with multiple metrics and cro 11 optimize Select optimal parameters based of mutiple qua 12 compare Evaluate tradeoffs between different model com 13 unstructured apply multiple representations and compare and 14 tools Scope, choose an appropriate tool pipeline and					
P1	P2	Р3	P4		
Unnamed: 2_level_1	Unnamed: 3_level_1 Unnamed	: 4_level_1 Unnamed	I: 5_level_1		
10 0	1	1	0		
11 0	0	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()

memb_	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
	1	burntbricks	6	7	2016-11- 17T00:00:00Z	God	4	3
	1	burntbricks	40	7	2016-11- 17T00:00:00Z	God	5	4

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\_ass

## key\_ID

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

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

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')

<AxesSubplot:xlabel='no_membrs', ylabel='rooms'>

8
7
6
9
4
3
2
1
2.5 50 7.5 10.0 12.5 15.0 17.5
```

# After Class Questions

How can we clean data? what other topics will we cover in this class?

How does the syntax on the question from prismia today work?

Where I can go to find a list of all the syntax.

no\_membrs

## 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' ]
topics

['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:
```

#### Announcements

topics[-2:]

- next assignment will go up today
- try practingin throughout the week

['dictionaries', 'pandas']

• Check Brightspace for TA office hours

# Why Exploratory Data Analysis (EDA) before cleaning?

Typically a Data process looks like one of these figures:



# THE DATA SCIENCE PROCESS



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.

## **EDA**

# a version of the SAFI data with more columns
data\_url = 'https://raw.githubusercontent.com/brownsarahm/python-socialscifiles/master/data/SAFI\_full\_shortname.csv'

## import pandas as pd

and pull in the data.

<pre>safi_df = pd.read_csv(data_url) safi_df.head(2)</pre>
--

	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

1		1
ı		
1	safi_df.describe()	
п		

	key_id	quest_no	years_farm	note2	no_membrs	members_count	years_l
count	131.000000	131.000000	131.000000	0.0	131.00000	131.00000	131.00000
mean	66.000000	85.473282	15.832061	NaN	7.19084	7.19084	23.05343
std	37.960506	63.151628	10.903883	NaN	3.17227	3.17227	16.91304
min	1.000000	1.000000	1.000000	NaN	2.00000	2.00000	1.00000
25%	33.500000	32.500000	8.000000	NaN	5.00000	5.00000	12.00000
50%	66.000000	66.000000	15.000000	NaN	7.00000	7.00000	20.00000
75%	98.500000	138.000000	20.500000	NaN	9.00000	9.00000	27.50000
max	131.000000	202.000000	60.000000	NaN	19.00000	19.00000	96.00000

 $8 \text{ rows} \times 25 \text{ columns}$ 

Then we remember that this includes <code>key\_id</code> as a variable tht we don't actually want to treat as a variable, it's an index. We can change that with <code>set\_index</code> and we can do it in memory (without assignment) using the <code>inplace</code> keyword.

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

	quest_no	years_farm	note2	no_membrs	members_count	years_liv	responden
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	

 $8 \text{ rows} \times 24 \text{ columns}$ 

## 1 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.

#### 1 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
count 131.000000
       15.832061
mean
       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. Pep8, 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()
```

# 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	17.28205	
God	43.0	81.720930	77.863839	1.0	14.5	40.0	168.5	202.0	43.0	14.88372	
Ruaca	49.0	107.061224	55.024013	23.0	72.0	113.0	152.0	194.0	49.0	15.51020	
3 rows × 19	2 colum	nns									

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	ws × 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/i
mage)

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 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
  1
      4
  2
      5
1
2 3 6
>>> df.rename({1: 2, 2: 4}, axis='index')
  A B
  1 4
      5
2
  2
4
  3
```

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	auest 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 max 105.468983 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
```

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.



	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

 $5 \text{ rows} \times 65 \text{ columns}$ 

```
type(safi_df_inplace.set_index('key_id',inplace=True))

NoneType
safi_df_inplace.head()
```

🥊 Tip

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

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

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
```

# 1 Note

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.

```
{\tt respondent\_wall\_type} \quad {\tt respondent\_floor\_type}
                                                       30
burntbricks
                         cement
                         earth
                                                       37
cement
                          cement
                                                       1
muddaub
                                                       3
                         cement
                                                       43
                         earth
sunbricks
                          cement
                                                       4
                                                       13
                          earth
Name: instanceID, dtype: int64
```

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

```
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 <a href="https://stimeit.nd/">https://stimeit.nd/<a href="https://stimeit.nd/">ht

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()

3.09 ms ± 33.6 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)

<TimeitResult : 3.09 ms ± 33.6 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



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 text book

```
%%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('_')
```

```
98.2 \mus \pm 1.93 \mus per loop (mean \pm std. dev. of 7 runs, 10000 loops each)

<TimeitResult : 98.2 \mus \pm 1.93 \mus 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
31.46101173869179
```

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, <u>let me know</u> 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 <u>better formatting</u>.

```
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

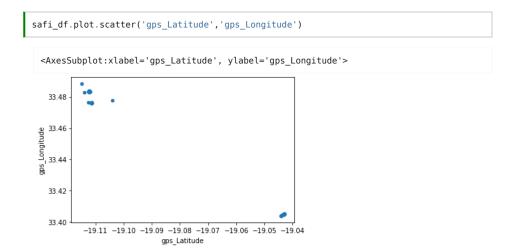
You can also try it on your computer and see how it compares. Or try other loops/ways of doing it in pandas.

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>

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

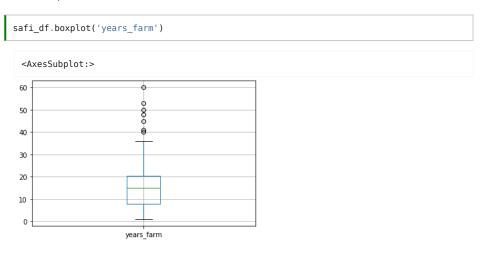
 $5 \text{ rows} \times 64 \text{ columns}$ 

We can make scatter plots as we saw Friday.



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.

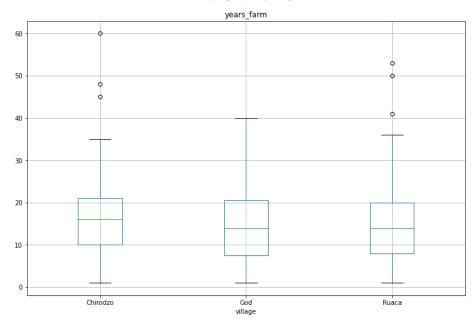


We can also make the figure larger  $\,$ 

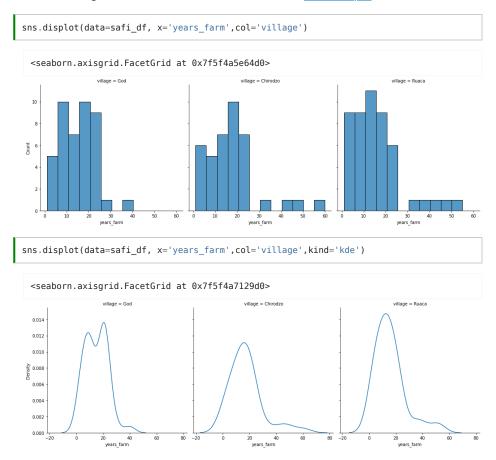
```
safi_df.boxplot('years_farm', by='village',figsize=(12,8))

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

#### Boxplot grouped by 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

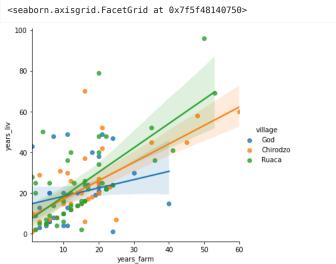
Then restart your notebook's kernel and re-run all cells

or, in a notebook, you can update with

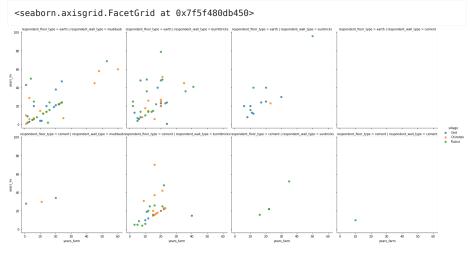
!pip install seaborn==0.11.0

# 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.



```
safi_df.groupby('poultry')['members_count'].mean()
```

```
poultry
no 5.894737
yes 7.720430
Name: members_count, dtype: float64
```

# Questions after class

## More practice

Test out the parameters of the plotting functions to see what they do.  $\label{eq:parameters}$ 

## **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:

## **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 <u>launch a notebook</u>
- 2. restart and rerun notebooks
  - $\circ \;\;$  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, restart and rerun 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

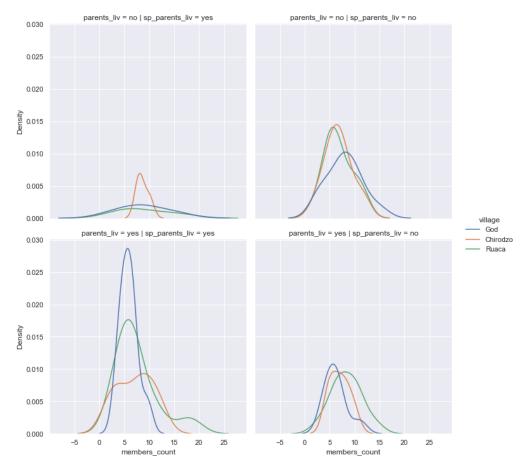
data_url = 'https://raw.githubusercontent.com/brownsarahm/python-socialsci-
files/master/data/SAFI_full_shortname.csv'
sns.set(font_scale=1.25)
```

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?

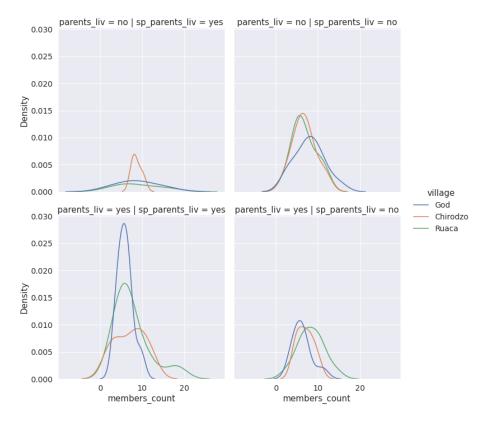


First, load the data

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

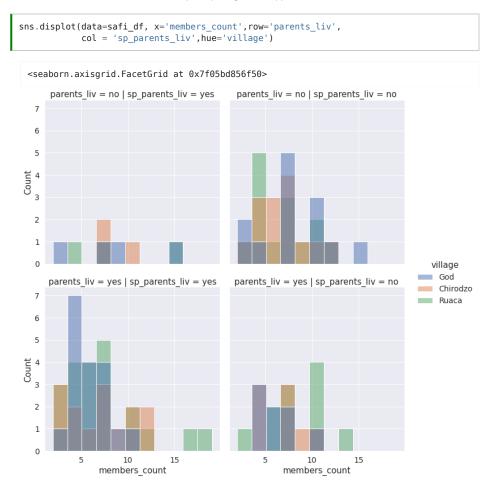
This is a <u>displot</u>.

<seaborn.axisgrid.FacetGrid at 0x7f05c0925d90>



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.

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 × 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()
```

```
['mobile_phone']
4
['radio']
4
['bicycle'; 'radio'; 'mobile_phone']
3
['bicycle'; 'radio'; 'cow_plough'; 'solar_panel'; 'solar_torch';
'mobile_phone'] 3
['bicycle'; 'radio'; 'cow_plough'; 'solar_panel'; 'table'; 'mobile_phone']
3
...
['motorcyle'; 'television'; 'radio'; 'solar_panel'; 'solar_torch'; 'table'; 'mobile_phone'] 1
['bicycle'; 'radio'; 'cow_plough'; 'mobile_phone']
1
['bicycle'; 'cow_plough']
1
['bicycle'; 'solar_torch'; 'table'; 'sofa_set'; 'mobile_phone']
1
['bicycle'; 'television'; 'radio'; 'cow_plough'; 'solar_panel'; 'solar_torch'; 'mobile_phone'] 1
['bicycle'; 'television'; 'radio'; 'cow_plough'; 'solar_panel'; 'solar_torch'; 'mobile_phone'] 1
Name: items_owned, Length: 95, dtype: int64
```

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	distri
0	1	17 November 2016	1	2017-03- 23T09:49:57.000Z	2017-04- 02T17:29:08.000Z	Manica	Manic
1	2	17 November 2016	1	2017-04- 02T09:48:16.000Z		Manica	Manic
2	3	17 November 2016	3	2017-04- 02T14:35:26.000Z	2017-04- 02T17:26:53.000Z	Manica	Manic
3	4	17 November 2016	4		2017-04- 02T17:27:16.000Z	Manica	Manic
4	5	17 November 2016	5	2017-04- 02T15:10:35.000Z	2017-04- 02T17:27:35.000Z	Manica	Manic
			<b></b>				
126	127	18 May 2017	126	2017-05- 18T04:13:37.000Z	2017-05- 18T04:35:47.000Z	Manica	Manic
127	128	04 June 2017	193	2017-06- 04T09:36:20.000Z	2017-06- 04T10:13:32.000Z	Manica	Manic
128	129	04 June 2017	194	2017-06- 04T10:13:36.000Z	2017-06- 04T10:32:06.000Z	Manica	Manic
129	130	04 June 2017	199	2017-06- 04T10:33:55.000Z	2017-06- 04T10:52:22.000Z	Manica	Manic
130	131	04 June 2017	200	2017-06- 04T10:52:46.000Z	2017-06- 04T11:08:13.000Z	Manica	Manic

131 rows  $\times$  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

4

2

3

4

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.

```
farm1_items

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

We could try first casting it to a list.

```
list(farml_items)
```

```
['[',
'b',
'i',
'c',
'y',
'c',
   '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)
   7545
                    kwds=kwds,
   7546
-> 7547
                return op.get_result()
   7548
   7549
            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
            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)
    253
    254
            def apply_standard(self):
--> 255
                results, res_index = self.apply_series_generator()
    256
    257
                # wrap results
/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/core/apply.py in apply_series_generator(self)
    282
                        for i, v in enumerate(series_gen):
    283
                            # ignore SettingWithCopy here in case the user mutates
--> 284
                            results[i] = self.f(v)
    285
                            if isinstance(results[i], ABCSeries):
    286
                                # 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.

# **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 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



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 function name.

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

- 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.

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, <u>request a review</u> from the team rhodyprog4ds/Fall20instructors.

## 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.

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.

#### 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/fall20in structors.

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.

#### Note

when the value of the expression after assert is True, it will look like nothing happened. assert is used for testing 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 tutorial on git 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.



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/)



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



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.

₱ Tip

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

# 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 <u>GitHub Classroom</u>. 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
  - 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.
- 4. Produce one additional plot of a different plot type that shows something about this data.

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

## 🥊 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.

#### 🥊 Tip

if you use a local copy of the dataset you can load from a relative path 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-dataset-ever.csv and upload the data to your assignment repository as well.

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.

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	$generate\ complex\ plots\ with\ pandas\ and\ plotting\ libraries\ and\ customize\ with\ matplot lib$
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.

# 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.

## File Naming

It is best practice to name files without spaces.

1

🥊 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

## Configurations

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

## 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  $\boxed{1}$  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 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.

[<u>1</u>]

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 trasforming a dataset before loading?
- How much can you repair a dataset using the parameters of the load functions?

# Note It's all

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

# CheatSheet

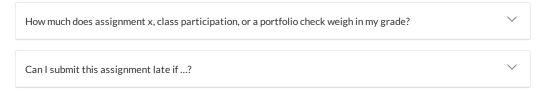
Make a cheatsheet with examples of the several different parameter settings for common operations for one topic.

# **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

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
- <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
- <u>UCI Data Repository</u>
- Json Datasets

If you have others please share by creating a pull request or issue on this repo