XXX: Course title here

Fall 2024

Table of contents

1 Course information **2**

2 Course description **2**
2.1 Extended description 2

3 Instructor and teaching assistants **3**
3.1 Instructor 3
3.2 Teaching assistants 3

4 Course learning goals **4**
4.1 Course structure 5

5 Assignments and grading **6**
5.1 Problem sets (90 percent of total grade) 6
5.2 Participation (10 percent of total grade) 8
5.3 Grading scale 8

6 Course topics **8**

1 Course information

Resource	Link
Weekly meetings (online)	Fridays 9-11AM Pacific
Class website (public)	https://anyone-can-cook.github.io/rclass1/
Questions, discussion, announcements (private)	https://github.com/anyone-can-cook/rclass1_student_issues_f23
Class Zoom link	https://ucla.zoom.us/j/99687673673

2 Course description

The primary goals of this course are (1) to teach fundamental skills of "data management," which are important regardless of which programming language you use, and (2) to develop a strong foundation in the R programming language. The course is designed for students who never thought they would become programmers and no prior experience with R is required. For goal (1), most statistics courses teach you how to analyze data that are ready for analysis. In real research projects, data management – the process of cleaning, manipulating, and integrating datasets in order to create analysis datasets – is often more challenging than conducting analyses. For goal (2), R is a free, open-source, object-oriented programming language. R is the most popular language for statistical analysis and one of the most popular languages for "data science" applications (e.g., web-scraping, interactive maps, network analysis). Students will become proficient in data management and R programming through weekly problem sets, which will be completed in groups.

2.1 Extended description

Data management consists of acquiring, investigating, cleaning, combining, and manipulating data. Most statistics courses teach you how to analyze data that are ready for analysis. In real research projects, cleaning the data and creating analysis datasets is often more time consuming than conducting analyses. This course teaches the fundamental data management and data manipulation skills necessary for creating analysis datasets.

The course will be taught using R, a free, open-source programming language. R has become the most popular language for statistical analysis, surpassing SPSS, Stata, and SAS. What differentiates R from these other languages is the thousands of open-source "libraries" created by R users. R is one of the most popular languages for "data science" because R libraries have been created for web-scraping, mapping, network analysis, etc. By learning R you can be confident that you know a programming language that can run any modeling technique

you might need and has amazing capabilities for data collection and data visualization. By learning fundamentals of R in this course, you will be "one step away" from web-scraping, network analysis, interactive maps, quantitative text analysis, or whatever other data science application you are interested in.

The data management and programming skills you learn in this course will transfer to other object-oriented programming languages (e.g., Python).

The course primarily use data and examples from social sciences research and designed to teach skills that are important for social science research more broadly and also for computational research within the humanities. **We welcome students from across the university**.

Recommended prerequisites (encourage, but not required)

- One prior introductory statistics course (e.g., undergraduate-level stats course)
- Proficiency in general computer skills is helpful, e.g., downloading files from internet, renaming files, saving them to a folder of your choosing, finding this folder on your computer, etc

3 Instructor and teaching assistants

3.1 Instructor

Instructor Name

- Pronouns: he/him/his
- Office: Moore Hall, Room 3038
- Email:
- Office hours:
  - Zoom office hours: DAY TIME, zoom link
    - Note: different than class zoom link
  - And by appointment (afternoons)

3.2 Teaching assistants

Name

- Pronouns:
- Email:
- Office hours:
  - Zoom office hours:

  - And by appointment

Name

- Pronouns:
- Email:
- Office hours:
  - Zoom office hours:
  - And by appointment

4 Course learning goals

1. Understand fundamental concepts of object-oriented programming
   - What are the basic object types and how do they apply to statistical analysis?
   - What are object attributes and how do they apply to statistical analysis?
2. Become familiar with Base R approach to data manipulation and Tidyverse approach to data manipulation
3. Investigate data patterns
   - Sort datasets in ways that generate insights about data structure
   - Select specific observations and specific variables in order to identify data structure and to examine whether variables are created correctly
   - Create summary statistics of particular variables to diagnose errors in data
4. Create variables
   - Create variables that require calculations across columns
   - Create variables that require processing across rows
5. Combine multiple datasets
   - Join (merge) datasets
   - Append (stack) datasets
6. Manipulate the organizational structure of datasets
   - Summarize and collapse observations by group
   - Reshape and "tidy" untidy data
7. Learn guidelines and practical strategies for ensuring data quality when cleaning data and creating analysis variables
8. Become proficient at using GitHub – the industry standard platform used by programmers to collaborate on projects – to ask questions about course material and to collaborate with your classmates

Another broad goal of the course is for students to begin developing practical proficiency in "computational thinking." The California Computer Science Standards define computational thinking as "the human ability to formulate problems so that their solutions can be represented as computational steps or algorithms to be executed by a computer." This course will encourage students to work on the following elements of computational thinking:

- Before you start writing code to accomplish some task, write out the individual steps that must be completed to accomplish the task
- When a particular piece of code is not working, develop a problem-solving approach where you change one element of the code at a time in order to systematically isolate and fix the problem
- For when you conceptually understand what you need to do but you don't know the code to accomplish the task, develop a set of "go to" practices to help you figure it out, for example:
  - Ask Google
  - Post a question on the course GitHub "issues" page
  - Become proficient at searching the course lecture slides and course textbook for answers
  - When you know the right function, but not how to use it, become proficient at reading function documentation

4.1 Course structure

Overview. Course structure consists of weekly **asynchronous course materials** and weekly **synchronous meetings**. Each week we will focus on a particular topic (e.g., creating variables; writing functions). For each weekly topic, students will complete a problem set. Problem sets will be completed in groups and focus on practical application of concepts/skills from the topic of the week.

Asynchronous course materials. Asynchronous course materials will focus on the topic for that week (e.g., processing across rows). Course materials will consist of three types of resources:

1. Detailed lecture slides (PDF or HTML) with sample code
2. Pre-recorded video lecture of the instructor working through these slides
3. The ".Rmd" file that created the PDF/HTML lecture slides.
   - The .Rmd file will contain all "code chunks" and links to all data utilized in the lecture. Thus, students will "learn by doing" in that they will run *R* code on their own computer while they work through lecture materials on their own.

Synchronous meetings. Synchronous class meetings will be on Zoom. Attendance during the entire period is required, but students may ask instructor/TAs for exceptions due to scheduling conflicts.

During synchronous class time, students will have the option of (A) attending live lecture from the instructor or (B) working through lecture materials/problem sets in Zoom breakout rooms in small groups (e.g., problem set groups) or on their own. For the first three weeks of class, students will not have the option of working in Zoom breakout rooms.

For students who decide to work in Zoom breakout rooms, you will use this time to work through course materials (e.g., lecture slides, video lectures) and/or the associated problem set as you see fit. The synchronous workshops are also a great time to ask questions about course material or practical applications. TAs will be moving from one breakout room to the next, providing help. Each group can develop their own approach to how they want to use the synchronous workshop time. Some groups may work relatively independently, while others may work collaboratively. Some groups may agree to work through all asynchronous lecture materials beforehand so they can devote all workshop time to making progress on the problem set. The one requirement I make: do not do the problem set before working through the associated lecture material.

5 Assignments and grading

Course grade will be based on the following components:

- Weekly problem sets (90 percent of total grade)
- Participation (10 percent of total grade)

5.1 Problem sets (90 percent of total grade)

Students will complete 10 problem sets (the last one due during finals week). Problem sets are due by 9am each Friday, right before we start class. In general, each problem set will give you practice using the skills and concepts introduced in course materials for thate week. For example, after the lecture on joining (merging) datasets, the problem set for that week will require that students complete several different tasks involving merging data. Additionally, the weekly problem sets will require you to use data manipulation skills you learned in previous weeks. Link to problem set expectations and helpful resources HERE.

Problem set groups

- With the exception of the first problem set, students will complete problem sets in groups of 3. We highly encourage students who are abroad to form their own group to set a time to work on the problem sets together.

- Students have the option of not being part of a problem set group.
- We will form groups during the second synchronous class and you will keep the same group throughout the quarter. However, each student will submit their own assignment. You are encouraged to work together and get help from your group. However, it is important that you understand how to do the problem set on your own, rather than copying the solution developed by group members.
- Since you will be working together, it is understandable that answers for many questions will be the same as your group members. However, if I find compelling evidence that a student merely copied solutions from a classmate, I will consider this a violation of academic integrity and that student will receive a zero for the homework assignment.

A general strategy I recommend for completing the problem sets is as follows: (1) after lecture, do the reading associated with that lecture; (2) try doing the problem set on your own; (3) communicate with your group to work through the problem set, with a particular focus on areas group members find challenging.

Grading policies

- For students working in a problem set group, one submission from each problem set group will be chosen at random. The grade on that problem set submission will be the grade for all members of the group.
  - If a member of a problem set group has not submitted the problem set by the time the TAs conduct grading, that submission will be grades separately once it is submitted
  - The lowest problem-set grade will be dropped from the calculation of your final grade.
- Students who are not part of a problem set group will have their problem sets graded individually. A random subset of 4 or 5 problem sets will be graded. For students who work individually, the lowest problem set grade will not be dropped from calculation of final grade.
- Weekly required participation on github will be part of your problem set grade
- Policy on late assignments
  - Problem sets submitted after 11:59PM on Friday will lose one percentage point (e.g., max grade becomes 99% instead of 100%)
  - Starting at 12AM Monday morning, problem sets will lose an additional percentage point for each week-day it is not submitted
    - e.g., for a problem set submitted at 10AM on Monday, the max grade becomes 98%
    - e.g., for a problem set submitted at 10AM on Tuesday, the max grade becomes 97%
  - For late submissions due to an unexpected emergency, you will not lose points. Please contact the instructor and/or TAs and we will work it out together.

5.2 Participation (10 percent of total grade)

Broadly, we expect students to participate by being attentive, supportive of classmates, by asking questions, and by answering questions posed by classmates.

Practically speaking, the vast majority of your participation grade will depend on weekly participation on Github. Each week, students are required to post one communication on Github. This could be asking a question about the problem set, answering a question posed by a classmate, or a post describing something you learned while working through the week's material/problem set. If you post at least one communication on Github each week, you will earn an "A" for participation for the quarter.

In addition, students can work towards an 100% participation grade for the quarter by asking/answering questions during synchronous lecture (e.g., zoom chat) or by consistently being helpful/supportive to your classmates on Github.

5.3 Grading scale

6 Course topics

Below is an overview of course topics. Topics and schedule are subject to change at the discretion of the instructor. Topics may be cut if we need to devote more time to learning the most central topics. It is unlikely that additional topics will be added. The official course schedule, including weekly required reading and optional reading, will be posted on the course website.

1. **Week 1**: Introduction to R
    - Introduction to R and R data structures
    - Execute R commands, understand R objects and data structures, use R functions
2. Investigating data patterns in Base R [two weeks]
    - Data investigation and manipulation using Base R
    - Investigate R object type and structure, isolate elements using Base R subset operators and the `subset()` function, create new variables in Base R
3. Enter the tidyverse: pipes, dplyr, and variable creation
    - Data investigation and manipulation using tidyverse
    - Select, filter, and sort data using `tidyverse` functions, chain functions together using pipes (`%>%`), create new variables conditionally using `if_else()`, `recode()`, and `case_when()`
4. Enter the tidyverse: processing across rows

- Calculate aggregate statistics from multiple rows of data
- Group rows of data using `group_by()`, create aggregate statistics using `summarize()`

5. Strings and dates

    - Work with strings and date/datetime objects
    - Understand string basics, manipulate strings using `stringr` functions, work with dates and times using the `lubridate` package

6. Attributes and class

    - Understand the class and attributes of R objects
    - Investigate R object class and attributes, work with factor variables, label variables and values of a dataframe using the `labelled` package

7. Data quality

    - Tools and guidelines for exploratory data analysis (EDA)
    - Learn common approaches for exploring and analyzing data, understand skip patterns in survey data

8. Tidy data

    - Understand tidy data structure and reshaping data
    - Define tidy data and how to reshape untidy data into tidy form, reshape data from wide to long using `pivot_longer()`, reshape data from long to wide using `pivot_wider()`, handle missing values during reshaping

9. Joining data

    - Combine data from multiple datasets using joins
    - Merge datasets using mutating joins, check quality of merge using filtering joins, append datasets by stacking rows