# STAT 243: Introduction to Statistical Computing
# Fall 2020 (Paciorek)

August 24, 2020

## Course Description

Statistics 243 is an introduction to statistical computing, taught using R. The course will cover both programming concepts and statistical computing concepts. Programming concepts will include data and text manipulation, data structures, functions and variable scope, regular expressions, debugging, and parallel processing. Statistical computing topics will include working with large datasets, numerical linear algebra, computer arithmetic/precision, simulation studies and Monte Carlo methods, numerical optimization, and numerical integration/differentiation. A goal is that coverage of these topics complement the models/methods discussed in the rest of the statistics/biostatistics graduate curriculum. We will also cover the basics of UNIX/Linux, in particular some basic shell scripting and operating on remote servers, as well as a bit of Python.

While the course is taught using R and you will learn a lot about using R at an advanced level, the focus of the course is statistical computing more generally. Also, this is not a course that will cover specific statistical/data analysis methods.

Informal prerequisites: If you are not a statistics or biostatistics graduate student, please chat with me if you're not sure if this course makes sense for you. A background in calculus, linear algebra, probability and statistics is expected, as well as a basic ability to operate on a computer (but I do not assume familiarity with the UNIX-style command line/terminal/shell). Furthermore, I'm expecting you will know the basics of R, at the level of the Modules 1-5 in the R bootcamp offered Aug. 10-13, 2020. If you don't have that background you'll need to spend time in the initial couple weeks getting up to speed. All the material from the bootcamp is available here (plus I can give you access to the bootcamp videos), we'll have a hands-on practice session in Lab 0 on Friday August 28, and the GSI can also provide assistance.

**DISCLAIMER: Given the evolving situation, we may need to adjust some of the approaches and policies stated here, but I will try to be very clear if/when I do this.**

## Objectives of the course

The goals of the course are that, by the end of the course, students be able to:

- operate effectively in a UNIX environment and on remote servers;
- program effectively in R with an advanced knowledge of R functionality and an understanding of general programming concepts and principles;
- be familiar with concepts and tools for reproducible research and good scientific computing practices; and
- understand in depth and be able to make use of principles of numerical linear algebra, optimization, and simulation for statistics- and data science-related analyses and research.

## Personnel

- Instructor:
  - Chris Paciorek (paciorek@stat.berkeley.edu)
- GSI
  - Zoe Vernon (zoe_vernon@berkeley.edu)
- We'll post office hours on the Github site README. I expect we'll have some hours where you can just show up and others where we have sign-up slots for individual-specific questions.
- **When to see us about an assignment**: We're here to help, including providing guidance on assignments. You don't want to be futilely spinning your wheels for a long time getting nowhere. That said, before coming to see us about a difficulty, you should try something a few different ways and define/summarize what is going wrong or where you are getting stuck.

## Class sessions

Given the virtual context, class sessions will be somewhat different than the usual university course.

Each unit has a set of notes and demo code that will form the basis of the instruction for that unit. For some material, we will make use of tutorials provided by the SCF (see link below).

- I will assign reading and pre-recorded videos for each unit. There will be a short assignment that will check your understanding of the materials.
- Some (but not all) class sessions will be live on Zoom, focused on presentation of topics benefiting from the live format, group work on problems, and discussion. All of the main sessions (but not breakout groups) will be recorded for anyone in the class (but not anyone else) to view later. In some class sessions you will be asked to work on a group problem and will need to turn in your answers.
- Those not able to attend live class sessions will have the option of watching the video separately but will still need to turn in work related to the class session.

My goal is to have the class sessions that we do hold live be an interactive environment. Part of this will involve group work where I will pose questions to the class to think about, respond to via Google forms, and discuss. During times where I am doing a demo or otherwise presenting, I encourage you to ask questions verbally and (likely) through the Piazza forum.

### Zoom guidance

- Microphones and questions: please keep your microphone muted, but feel free to interrupt me with a question or raise your virtual hand. I'll also try to remember to pause for questions regularly.
- We'll experiment with having text-based questions in Piazza during class sessions. Please post to the 'class' folder on Piazza.
- If possible please have your video on during class sessions. Unless absolutely necessary not to, please have your video on when working in a group or asking me a question.
- Please do your best to stay focused during class time and section group work (I know this can be hard).

  - I ask that you leave your phone elsewhere and keep other windows/apps on your computer closed or out of your direct view.

- If you have concerns about being recorded in the Zoom sessions, please let me know so we can discuss how you can avoid being recorded.

- If you have trouble accessing a Zoom session or don't see me there, please check Piazza for updates.

Student backgrounds with computing will vary. For those of you with limited background on a topic, I encourage you to ask questions on Piazza and during class. For those of you with extensive background on a topic (there will invariably be some topics where one of you will know more about it than I do), I encourage you to pitch in with your perspective. In general, there are many ways to do things on a computer, particularly in a UNIX environment and in R, so it will help everyone (including me) if we hear multiple perspectives/ideas.

## Course websites: Github, Piazza, GradeScope, and bCourses

Key websites for the course are:

- Github for course content: https://github.com/berkeley-stat243/stat243-fall-2020, including logistics info on the main Github page (scroll down below the files listing).

- SCF tutorials for additional content: https://statistics.berkeley.edu/computing/training/tutorials

- Piazza site for discussions/Q&A (also linked from bCourses): https://piazza.com/berkeley/fall2020/stat243

- bCourses site for course recordings, Zoom links, and possibly some other materials:

  https://bcourses.berkeley.edu/courses/1497598.

- Gradescope for assignments (also linked from bCourses): https://www.gradescope.com/courses/166974

All course materials will be posted on Github except for pre-recorded videos and class Zoom recordings, which will be in bCourses.

We will use the course Piazza site for communication (announcements, questions, and discussion). You should ask questions about class material and problem sets through Piazza. Please use this site for your questions so that either Zoe or I can respond and so that everyone can benefit from the discussion. I suggest you to modify your settings on Piazza so you are informed by email of postings. I strongly encourage you to respond to or comment on each other's questions as well (this will help your class participation grade), although of course you should not provide a solution to a problem set problem. If you have a specific administrative question you need to direct just to me, it's fine to email me directly. But if you simply want to privately ask a question about content, then just come to an office hour or see me after class.

In addition, we will use Gradescope for viewing grades.

## Course material

- Primary materials: Course notes on Github and video recordings on bCourses.

- Back-up textbooks:

  - For bash: Newham, Cameron and Rosenblatt, Bill. Learning the bash Shell (available electronically through OskiCat: http://uclibs.org/PID/77225)

- For R:
  - Adler, Joseph; R in a Nutshell (available electronically through OskiCat: http://uclibs.org/PID/151634)
  - Wickham, Hadley: Advanced R: http://adv-r.had.co.nz/
- For statistical computing topics:
  - Gentle, James. Computational Statistics (available electronically through OskiCat: http://dx.doi.org/10.1007/0-387-98144-4)
  - Gentle, James. Matrix Algebra https://link-springer-com.libproxy.berkeley.edu/book/10.1007%2F978-3-319-64867-5 or Numerical Linear Algebra with Applications in Statistics https://link-springer-com.libproxy.berkeley.edu/chapter/10.1007/978-1-4612-0623-1_1
- Other resources with more details on particular aspects of R:
  - Chambers, John; Software for Data Analysis: Programming with R (available electronically through OskiCat: http://dx.doi.org/10.1007/978-0-387-75936-4)
  - Xie, Yihui; Dynamic documents with R and knitr. (available electronically through Oskicat)
  - Nolan, Deborah and Temple Lang, Duncan. XML and Web Technologies for Data Sciences with R. https://link.springer.com/book/10.1007%2F978-1-4614-7900-0
  - The R-intro and R-lang documentation. https://www.cran.r-project.org/manuals.html
  - Murrell, Paul; R Graphics, 2nd ed. http://www.stat.auckland.ac.nz/∼paul/RG2e/
  - Murrell, Paul; Introduction to Data Technologies. http://www.stat.auckland.ac.nz/∼paul/ItDT/
- Other resources with more detail on particular aspects of statistical computing concepts:
  - Lange, Kenneth; Numerical Analysis for Statisticians, 2nd ed. (first edition is available electronically through OskiCat: https://link.springer.com/book/10.1007%2Fb98850)
  - Monahan, John; Numerical Methods of Statistics (available electronically through OskiCat: http://dx.doi.org/10.1017/CBO9780511977176)

## Section

The GSI will lead section each week. Given the virtual format, our plan is to have each student sign up for a half-hour within the Friday 12-4 pm time window, with those in incompatible time zones accommodated separately. Discussion sections will generally involve group work on various topics (version control, debugging, testing, Python, etc.), discussion of problem set solutions, or discussion of a particular topic. Generally, the GSI will provide some written or pre-recorded video material to get you started, rather than doing live presentation. She will then circulate amongst the groups in Zoom breakout rooms.

If at all possible, please have your cameras on during group work in section and when interacting with the GSI.

For those unable to make section (primarily a few of you in Asia), we will likely have a combined section/office hour time late in the day Wednesday (Pacific time) / early Thursday (Asia times).

## Computing Resources

Most work for the course can be done on your laptop. Later in the course we'll also use the Statistics department cluster. You can also use the campus Datahub to access a bash shell or run RStudio.

The software needed for the course is as follows:

- Access to the UNIX command line (bash shell)

- Git
- R (RStudio is recommended but by no means required)
- Python (later in the course)

Some tips for software installation (and access to Datahub) are in the 'howtos' directory of the Git repository. In particular, please see 'accessingUnixCommandLine.html' for options of how to access a bash shell.

## Course requirements and grading

### Course grades

The grade for this course is primarily based on assignments due every 1-2 weeks, a short exam in November (or possibly two quizzes - one in early October and one in mid-November), and a final group project. I will also provide extra credit questions on some problem sets. There is no final exam. 50% of the grade is based on the problem sets, 25% on the exam, 15% on the project, and 10% on your class participation (primarily the reading/video assignments and responses to the in-class Google forms questions, as well as participation in the Piazza discussions).

Grades will generally be As and Bs. An A involves doing all the work, getting full credit on most of the problem sets, showing competence on the exam, and doing a thorough job on the final project.

### Problem sets

We will be less willing to help you if you come to our office hours or post a question online at the last minute. Working with computers can be unpredictable, so give yourself plenty of time for the assignments.

There are several rules for submitting your assignments.

1. You should prepare your assignments using either LaTeX plus knitr or R Markdown. (If you're a Statistics student, I encourage you use LaTeX plus knitr).
2. Problem set submission consists of **both** of the following:
   (a) A PDF submitted electronically through Gradescope, generally (but not always) **by the start of class (10 am)** on the due date, and
   (b) An electronic copy of the PDF, code file, and R Markdown/knitr document pushed to your class Github repository, following the instructions to be provided by the GSI.

   **On-time submission will be determined based on the time stamp of when the PDF is submitted to gradeScope.**
3. Answers should consist of textual response or mathematical expressions as appropriate, with key chunks of code embedded within the document. Extensive additional code can be provided as an appendix. Before diving into the code for a problem, you should say what the goal of the code is and your strategy for solving the problem. **Raw code without explanation is not an appropriate solution.**
4. Any mathematical derivations may be done by hand and scanned with your phone if you prefer that to writing up LaTeX equations.

Note: knitr is a tool that allows one to embed chunks of code within LaTeX documents. It can also be used with the LyX GUI front-end to LaTeX. R Markdown is an extension to the Markdown markup language that allows one to embed R code within an HTML document. Please see the dynamics document tutorial on the SCF tutorials website; there will be additional information in the first section and on the first problem set.

### Problem set grading

The grading scheme for problem sets is as follows. Each problem set will receive a numeric score for (1) presentation and explanation of results, (2) technical accuracy of code or mathematical derivation, and (3) code quality/style and creativity. For each of these three components, the possible scores are:

- 0 = no credit,
- 1 = partial credit (you did some of the problems but not all),
- 2 = satisfactory (you tried everything but there were pieces of what you did that didn't solve or present/explain one or more problems in a complete way), and
- 3 = full credit.

For components #1 and #3, many of you will get a score of 2 for some problem sets as you develop good coding practices. You can still get an A in the class despite this.

Your total score for the PS is a weighted sum of the scores for the three components. If you turn in a PS late, I'll bump you down by two points (out of the available). If you turn it in really late (i.e., after we start grading them), I will bump you down by four points. No credit after solutions are distributed.

### Final project

The final project will be a joint coding project in groups of 3-4. I'll assign an overall task, and you'll be responsible for dividing up the work, coding, debugging, testing, and documentation. You'll need to use the Git version control system for working in your group.

### Rules for working together and the campus honor code

I encourage you to work together and help each other out. However, with regard to the problem sets, you should first try to figure out a given problem on your own. After that, if you're stuck or want to explore alternative approaches, feel free to consult with your fellow students and with the GSI and me. You can share tips on general strategy or syntax for how to do individual tasks within a problem, but **you should not ask for and you should not share complete code or solutions** for a problem. Basically, you can help each other out, but no one should be doing the work for someone else. In particular, **your solution to a problem set (writeup and code) must be your own**, and you'll hear from me if either look too similar to someone else's. **You MUST note on your problem set solution any fellow students who you worked/consulted with. If you got a specific idea for how to do part of a problem from a fellow student, you should note that in your solution in the appropriate place**, just as you would cite a book or URL.

Please see the last section of this document for more information on the Campus Honor Code, which I expect you to follow.

### Feedback

I welcome comments and suggestions. In particular, many of you were in classes last spring and may have great suggestions for how to do things in a virtual environment - please let me know of these as they occur to you. Particularly good suggestions will count towards your class participation grade.

## Accomodations for Students with Disabilities

Please see me as soon as possible if you need particular accommodations, and we will work out the necessary arrangements.

## Scheduling Conflicts

Campus asks that I include this information about conflicts: Please notify me in writing by the second week of the term about any known or potential extracurricular conflicts (such as religious observances, graduate or medical school interviews, or team activities). I will try my best to help you with making accommodations, but cannot promise them in all cases. In the event there is no mutually-workable solution, you may be dropped from the class.

The main conflict that would be a problem would be the exam (or quizzes), whose date(s) is(are) TBD.

## Campus Honor Code

*The following is the Campus Honor Code. With regard to collaboration and independence, please see my rules regarding problem sets earlier in this document – Chris.*

The student community at UC Berkeley has adopted the following Honor Code: "As a member of the UC Berkeley community, I act with honesty, integrity, and respect for others." The hope and expectation is that you will adhere to this code.

Collaboration and Independence: Reviewing lecture and reading materials and studying for exams can be enjoyable and enriching things to do with fellow students. This is recommended. However, unless otherwise instructed, homework assignments are to be completed independently and materials submitted as homework should be the result of one's own independent work.

Cheating: A good lifetime strategy is always to act in such a way that no one would ever imagine that you would even consider cheating. Anyone caught cheating on a quiz or exam in this course will receive a failing grade in the course and will also be reported to the University Center for Student Conduct. In order to guarantee that you are not suspected of cheating, please keep your eyes on your own materials and do not converse with others during the quizzes and exams.

Plagiarism: To copy text or ideas from another source without appropriate reference is plagiarism and will result in a failing grade for your assignment and usually further disciplinary action. For additional information on plagiarism and how to avoid it, see, for example: http://gsi.berkeley.edu/teachingguide/misconduct/prevent-plag.html

Academic Integrity and Ethics: Cheating on exams and plagiarism are two common examples of dishonest, unethical behavior. Honesty and integrity are of great importance in all facets of life. They help to build a sense of self-confidence, and are key to building trust within relationships, whether personal or professional. There is no tolerance for dishonesty in the academic world, for it undermines what we are dedicated to doing – furthering knowledge for the benefit of humanity.

Your experience as a student at UC Berkeley is hopefully fueled by passion for learning and replete with fulfilling activities. And we also appreciate that being a student may be stressful. There may be times when there is temptation to engage in some kind of cheating in order to improve a grade or otherwise advance your career. This could be as blatant as having someone else sit for you in an exam, or submitting a written assignment that has been copied from another source. And it could be as subtle as glancing at a fellow student's exam when you are unsure of an answer to a question and are looking for some confirmation. One might do any of these things and potentially not get caught. However, if you cheat, no matter how much you may have learned in this class, you have failed to learn perhaps the most important lesson of all.

## Topics (in order with rough timing)

The 'days' here are class sessions under a non-virtual format, as general guidance.

1. Introduction to UNIX, operating on a compute server (1 day)
2. Data formats, data access, webscraping (2 days)
3. Debugging, good programming practices, reproducible research (1 day)
4. The bash shell and shell scripting, version control (3 days)
5. Programming concepts and advanced R programming: text processing and regular expressions, functions and variable scope, environments, object oriented programming, efficient programming (9 days)
6. Computer arithmetic/representation of numbers on a computer (3 days)
7. Parallel processing (2 days)
8. Working with databases, hashing, and big data (3 days)
9. Numerical linear algebra (5 days)
10. Simulation studies and Monte Carlo (2 days)
11. Optimization (7 days)
12. Numerical integration and differentiation (1 day)
13. Graphics (1 day)

If you want to get a sense of what material we will cover in more detail, in advance, you can take a look at the materials in the *units* directory of Github repository from when I taught the class in 2019. See https://github.com/berkeley-stat243/stat243-fall-2019.