STAT 243: Introduction to Statistical Computing
Fall 2021 (Paciorek)

August 23, 2021

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/testing, 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/computing for data science 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. 21-22, 2021. 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 an optional hands-on practice session during the second or third week of class, and the GSI can also provide assistance.

Covid considerations

We'll be following university policy through the semester. At the moment that means:

1. Class and section are in person. I will be recording class (but not section) via the room's course capture capabilities so if anyone needs to miss a class they should be able to catch up. There is in-class discussion and problem-solving so I expect students to attend class in general.

2. Masks are required at the moment. Hopefully the public health situation will improve to the point that masks will be optional later in the semester.

3. Vaccination is required with very limited exceptions.

Personally, I am feeling comfortable with having class under these circumstances, and I hope you are as well. If you have any concerns, please let me know; I'm happy to talk with you.

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 and compute clusters;
- 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
  - Andrew Vaughn (ahv36@berkeley.edu)
- We'll post office hours on the GitHub site README.
- **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.

Course websites: GitHub, Piazza, GradeScope, and bCourses

Key websites for the course are:

- GitHub for course content: https://github.com/berkeley-stat243/stat243-fall-2021, 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/fall2021/stat243
- bCourses site for course capture recordings and possibly some other materials:

  https://bcourses.berkeley.edu/courses/1507757.
- Gradescope for assignments (also linked from bCourses): [UNDER CONSTRUCTION]https://www.gradescope.com/c

All course materials will be posted on GitHub except for video content, 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 Andrew 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, SCF tutorials, and potentially pre-recorded videos 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 a two-hour discussion section each week (there are two sections). By and large, these will only last for about one hour of actual content, but the second hour may be used as an office hour with the GSI or for troubleshooting software during the early weeks. The discussion sections will vary in format and

topic, but material will include demonstrations on various topics (version control, debugging, testing, etc.), group work on these topics, discussion of relevant papers, and discussion of problem set solutions. The first section generally has more demand, so to avoid having too many people in the room, you should go to your assigned section unless you talk to me first.

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.

Class time

My goal is to have classes be an interactive environment. This is both more interesting for all of us and more effective in learning the material. I encourage you to ask questions and will pose questions to the class to think about, respond to via Google forms, and discuss (though I may have to adjust discussion given mask wearing and the public health situation). To increase time for discussion and assimilation of the material in class, before some classes I may ask that you read material or work through tutorials in advance of class. Occasionally, I will ask you to submit answers to questions in advance of class as well.

Please do not use phones during class and limit laptop use to the material being covered.

Student backgrounds with computing will vary. For those of you with limited background on a topic, I encourage you to ask questions during class so I know what you find confusing. 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 or have more real-world experience), 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.

Finally, class recordings for review or to make up for absence will be available through the bCourses Course Capture feature, available on the Course Captures tab on the bCourses page for the class.

Course requirements and grading

Course grades

The grade for this course is primarily based on assignments due every 1-2 weeks, two quizzes (likely in early-mid October and mid-late November, or possibly a single quiz/exam in 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 quizzes, 15% on the project, and 10% on your participation in discussions on Piazza, your responses to the in-class Google forms questions, as well as occasional brief questions that I will ask you to answer in advance of the next class.

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 quizzes, 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 R Markdown or LaTeX plus knitr.
2. Problem set submission consists of **both** of the following:
   (a) A PDF submitted electronically through Gradescope, **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 small 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 and concerns. 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 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 2020. See https://github.com/berkeley-stat243/stat243-fall-2020.