STAT 243: Introduction to Statistical Computing
Fall 2018 (Paciorek)

August 21, 2018

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 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 not necessary a UNIX variant). Furthermore, I'm expecting you will know the basics of R, at the level of the material in the R bootcamp offered Aug. 18-19, 2019 [specifically the material in Modules 1-6, but don't worry about environments and scoping]. 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, we'll have a hands-on practice session, and the GSI can also provide assistance.

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;
- 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-related research.

Personnel

- Instructor:
  - Chris Paciorek
    e-mail: paciorek@stat.berkeley.edu; Room 495, Evans Hall; Phone: (510) 642-9056;
- GSI
  - Omid Solari
    email: solari@berkeley.edu
- **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 try to define/summarize what is going wrong or where you are getting stuck.

Course websites: Github, Google groups, and bCourses

Key websites for the course are:

- Github for course content: https://github.com/berkeley-stat243/stat243-fall-2018, including logistics info on the main Github page (scroll down below the files listing).
- Google groups (bConnected lists) for online course discussion, announcements and Q&A: https://groups.google.com/a/ classes-2188-stat-243-all or email to stat-classes-2188-stat-243-all@calgroups.berkeley.edu
- SCF tutorials for additional content: https://statistics.berkeley.edu/computing/training/tutorials

All course materials will be posted on Github. Class will follow a set of course notes with demonstrations. I will do my best to post the slides and demo code for class by 6 pm the day before class. I will not print out copies, so please bring your own copies if you want them in front of you. Note that each unit will have a single set of notes and demo code that I will add to as we move through the unit, so you may want to just print out the new pages. I'll provide PDF documents as well as the underlying LyX file I used to generate the document. If you want a plain LaTeX version, open the LyX file in LyX and do `File->Export->LaTeX(pdflatex)`. For some material, we will make use of tutorials provided by the SCF through joint work of Chris Paciorek and Jarrod Millman.

We will use the Google group for communication (announcements, questions, and discussion). You should ask questions about class material and problem sets through the Google group website or by sending email to the list. Please use this site for your questions so that either the GSI or I can respond and so that everyone can benefit from the discussion. I suggest you to modify your settings on Google groups so you are informed by email of postings. I strongly encourage you to respond to each other's questions as well, 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.

We will not use bCourses except for viewing grades and accessing the class screencasts.

Course material

Primary 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/10.1007/978-0-387-98144-4)
- Assorted documents provided by me.

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. On reserve as a paper book in the Math/Stat library (1-day reserve period)
- 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. If anyone cannot make either section time, please see me to discuss alternative arrangements. If you can't make your assigned time, please also see me.

Computing Resources

Most work for the course can be done on your laptop. You can also make use of the Statistical Computing Facility (SCF) network of Linux and Mac computers. Anyone not in Statistics who would like an SCF account is also welcome to get one for the semester - see Chris for an account form. The computer rooms in 342 and 432 Evans provide Mac desktops; you can also remotely log in to the SCF system from other campus computers or from home.

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 are in the 'howtos' directory of the Git repository. In particular, please see 'accessingUnixCommandline.txt' 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 and discuss. 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), 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, screencasts will be available through the bCourses Course Capture feature. Please login to your bConnected account and go to the Course Captures tab on the bCourses page for the class. Note that the screencasts will necessarily be more effective for the computer demos and will miss material that I write on the board.

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, 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, 30% on the exam, 15% on the project, and 5% on your participation in discussions on the Google group 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 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 the following:

   (a) A paper copy submitted to Chris **at the start of class** on the due date, and

   (b) An electronic copy of the PDF, code file, and Markdown/knitr document, following the instructions to be provided by the GSI.

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 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 component #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 the sum of the scores for the three components. If you turn in a PS late, I'll bump you down by two points. 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 should 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 gripes). If you prefer anonymity, you can leave a note in my mailbox or under my door.

Topics (in order with rough timing)

1. Introduction to UNIX, operating on a compute server, the bash shell and shell scripting, version control (4 days)
2. Data formats, data access, webscraping (2 days)
3. Debugging, good programming practices, reproducible research (2 days)
4. Programming concepts and advanced R programming: functions and variable scope, data and text manipulation, strings and regular expressions, environments, object oriented programming, efficient programming, computing on the language (9 days)
5. Computer arithmetic/representation of numbers on a computer (3 days)
6. Parallel processing (2 days)
7. Working with databases, hashing, and big data (3 days)
8. Numerical linear algebra (5 days)
9. Simulation studies and Monte Carlo (2 days)
10. Optimization (7 days)
11. Numerical integration and differentiation (1 day)
12. 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 2017. See https://github.com/berkele-stat243/stat243-fall-2017. Material will be quite similar though I'll be making modifications along the way and it's possible unit numbers will have changed.

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.