STAT 243: Introduction to Statistical Computing
Fall 2013 (Paciorek)

August 29, 2013

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, flow control, functions and variable scope, regular expressions, matrix manipulations, debugging, and parallel processing. Statistical computing topics will include numerical linear algebra, 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. Note that until 2011, the course was a C programming course; we will NOT cover C in this class.

Note that I aim to have the course be useful to those who already know a fair amount of R by (1) covering more advanced aspects of R and (2) through the extensive coverage of the statistical computing topics.

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. 24-25, 2013. If you don't have that background you'll need to spend time in the initial couple weeks getting up to speed. Screencasts from the bootcamp will be available, 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; and
- 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;
    Office hours (in Evans 495): TBD, or just drop by if my door is open, or schedule an appointment.
- GSI
  - Tessa Childers-Day
    email: tchilders@stat.berkeley.edu
    Office hours (location TBD): time TBD
- **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, Piazza, and bSpace

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 R code for class by 4 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 on an SCF machine and do `File->Export->LaTeX(pdflatex)`.

We will use Piazza for communication (announcements and questions). You should ask questions about class material and problem sets through the course Piazza website:
piazza.com/berkeley/fall2013/stat243/home. If you're not already automatically enrolled on the site, you should sign up for the site at piazza.com/berkeley/fall2013/stat243 if you are not already s. Please use this site for your questions so that either Tessa or I can respond and so that everyone can benefit from the discussion. I encourage you to respond to each other's questions as well. I suggest you to modify your settings on Piazza so you are informed by email of postings. Questions of a personal nature can marked Private on Piazza. If you have a specific question you need to direct just to me, it's fine to email me directly.

I have set up a course Wiki on Github, building off the Wiki from last year. The goal is to build up a useful set of webpages that provide tips on statistical computing and programming. So if you figure out a clever way to do something, find a useful R function or package, have a way to clean up the Wiki, or see a useful reference to link to, please contribute this to the Wiki. One way to get credit for class participation is to contribute to the Wiki.

We'll only make limited use of bSpace, in particular for submitting the electronic versions of your problem set solutions.

Course material

Primary textbooks:

- For R (Chambers is more abstract and Adler more hands-on):

  - Chambers, John; Software for Data Analysis: Programming with R (available electronically through OskiCat: http://dx.doi.org/10.1007/978-0-387-75936-4)
  - Adler, Joseph; R in a Nutshell (available electronically through OskiCat: http://uclibs.org/PID/151634)

- 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:

- The R-intro and R-lang documentation. 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: http://sunsite.berkeley.edu:8080/librarysurvey/library.survey.logic?refUrl=http%3A%2F%2Fsite%2Eebrary%2Ecom%

- Monahan, John; Numerical Methods of Statistics (available electronically through OskiCat: http://dx.doi.org/10.1017/CBO9780511977176)

And for bash:

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

Section

Tessa will lead a one hour discussion section each week (there are two sections). The discussion sections will vary in format and topic but material will include discussion of problem set solutions, discussion of relevant papers, and demonstrations on various topics. If anyone cannot make either section time, please see me to discuss alternative arrangements.

Computing Resources

All students will be provided with a computer account on the Statistical Computer Facility (SCF) network of Linux and Mac computers. 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. If you wish to do your assignments on some other computer, keep in mind that required programs/packages may be stored on the SCF system, and it is your responsibility to get the required resources working on the other computer. Also, assignments and topics are oriented towards the UNIX (i.e., Linux or Mac) operating system, so if you wish to use a non-UNIX computer, you should make sure that the necessary resources are available (and in some cases, you will need to use a UNIX-based system). I can try to help with Windows-related issues but this help will necessarily be limited by my limited experience with Windows.

Course requirements and grading

Course grades

The grade for this course is primarily based on computer assignments due every 2-3 weeks and a final group project. There is no final exam. 10% of the grade is based on class participation. I will also occasionally pose questions/challenges to the class for discussion at the next class period. I will in some cases call on students to respond to these.

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

Class participation

The class participation portion of the grade can be satisfied in one or more of the following ways:

1. asking and answering questions in class (including discussion related to the questions/challenges),
2. contributing to the course Wiki on Github, and/or
3. contributing to class discussion through the Piazza site.

I may also provide extra credit questions on the problem sets. So if you really don't feel comfortable contributing through the mechanisms above, doing enough extra credit can make up for that. Extra credit can also make up for points taken off on other problems.

Problem sets

Problem will sometimes be somewhat open-ended, so those coming in at different levels may explore things with more or less sophistication. I'm also open to you defining your own assignment for a given topic, if you are working on a specific problem. E.g., instead of working on a particular text manipulation problem I assign, you might work with your own text data. Check with me before forging ahead.

We will be less willing to help you if you come to our office hours or Piazza 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 Latex plus knitr, Sweave, or R Markdown plus knitr (the latter should not be used if you are a Statistics student).
2. Submit your primary solutions on paper, including all your code.
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. Raw R code without explanation is not an appropriate solution.
4. In addition to the paper submission, submit your Latex/Markdown documents with embedded R code and any code appendix electronically through bSpace.
5. Any mathematical derivations may be done by hand.

Note: knitr and Sweave are tools that allow one to embed chunks of code within Latex documents. They 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. I will provide some tips about this on the first problem set.

Problem set grading

The grading scheme for problem sets is:

- 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 one or more problems in a complete way
- 3 = full credit
- 3+ = extra credit

If you turn in a PS late, I'll bump you down a number. If you turn it in really late (i.e., after I start grading them), I may bump you down two levels. 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 a version control system such as Git 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, in the context of the following guidelines. In terms of questions/challenges posed in class that we'll discuss at the next class, anything goes. In terms of 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 Tessa 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 nor 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.

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

Class time

My goal is to have classes be an interactive environment. This is both more interesting for all of us (hopefully) 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 in advance of class.

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, so it will help everyone (including me) if we hear multiple perspectives/ideas.

Feedback

I welcome comments and suggestions (and gripes) and will solicit feedback via a survey partway through the class. Comments at any other time are welcome, and 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. Using R: functions and variable scope, data and text manipulation, strings and regular expressions, data input/output (5 days)
3. Debugging, good programming practices, reproducible research (2 days)
4. Advanced R programming: environments, object oriented programming, efficient programming, computing on the language (5 days)
5. Programming concepts: a comparison with Python (2 days)
6. Parallel processing (2 days)
7. Working with databases, hashing, and big data (2 days)
8. Computer arithmetic/representation of numbers on a computer (3 days)
9. Numerical linear algebra (5 days)
10. Simulation studies and Monte Carlo (2 days)
11. Numerical integration and differentiation (2 days)
12. Optimization (5 days)
13. Graphics (2 days)

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.