Stat243: Final group project
Due Wednesday Dec. 12, 5 pm

November 13, 2018

The project will be done in groups of three, with students assigned randomly but balanced between statistics and non-statistics students.

A few comments. First, the project, when split amongst the group members, is not intended to be a huge undertaking. The goals of the project are to give you experience in working collaboratively and developing a well-designed, well-tested piece of software.

The project will be graded as a letter grade and will count for about as much as two problem sets in your final grade.

Problem

Your task is to implement an adaptive-rejection sampler, described in the Unit 11 notes and with details in Gilks et al. (1992) - the PDF is in the *project* directory on the class Github repository. I'd suggest you implement the tangent approach rather than the secant approach, but the latter is fine too. The result should be an R package that I can install and use.

1. Your solution should allow the user to provide reasonable inputs, including the number of points to sample, and should check the inputs for validity (e.g., using *assertthat*). The primary input should be an R function that calculates the (possibly unnormalized) density of the distribution of interest in a vectorized fashion (e.g., many of the "d" functions in R, such as "dnorm" are legitimate inputs). Your code should include numerical checks that catch cases of non-log-concave densities as the calculations proceed. (I.e., you do not need to come up with an overall test of the input density, but you should be able to do some checks that will catch cases where the upper and lower bounds are not actually bounding the density.)

2. Formal testing is required with a set of tests where results are compared to some known truth. You should have tests for the overall function and any modules that do anything complicated. Given the output is stochastic, how to do this will require some thought. The output of your testing should be clear and interpretable. I.e., when I run your tests I should see informative messages of what is being done and whether a given test was passed or failed. You should also have unit tests for individual functions that carry out the individual computations that make up the algorithm. Please use the *testthat* package to set up your tests. Note also that an important part of the grade will depend on how your code does on tests that I have prepared,

3. Your solution should involve modular code, with functions or OOP methods that implement discrete tasks. You should have an overall design and style that is consistent across the components, in terms of functions vs. OOP methods, naming of objects, etc.

4. In terms of efficiency, the algorithm is inherently sequential. However, you should try to vectorize as much as possible. One possibility in terms of the overall calculation is that you could generate a vector of samples based on the upper envelope. Then determine where are the points that require evaluation of $f(x)$. All points up to the first of those points can be generated before changing the envelope, at which point you would throw away the remaining points. How many points you generate at once might vary depending on how far along you are in generating the number of points requested by the user. Or you may think of other tricks.

5. Your solution should include help/manual information as follows:

   (a) Basic doc strings for any function you create (i.e., include comments at the beginning of the function), as well as commenting of code as appropriate.

   (b) An overall help page for the primary function only. This help page should be directly in the form of standard R documentation in a file called *ars.Rd*. You can either write *ars.Rd* by hand or you can have it generated based on using the *roxygen2* package (see http://adv-r.had.co.nz/Documenting-functions.html for an example); in the latter case you would have the documentation included in the R code files. You do not need help pages for your auxiliary functions.

Formatting requirements and additional information

Your solution to the problem should have two parts:

1. A PDF document describing your solution, prepared in LaTeX or R Markdown. The description does not need to be more than a few pages, but should describe the approach you took in terms of functions/modularity/object-oriented programming, and the testing that you carried out. It must include a paragraph describing the specific contributions of each team member and which person/people were responsible for each component of the work. **Please submit a paper copy of the document to me - either directly to me, under my door, or in my mailbox. On your paper solution, please indicate the Github user name of the group member in whose Github account the final version of the project resides.**

2. An R package named *ars*, including the .tar.gz file created by `R CMD build ars`, committed to the relevant Github repository. The package should include:

   (a) A primary function called *ars* that carries out the simulation, located in a file *ars.R* in the *R* directory of the package,

   (b) Formal tests you applied to your overall function, located in the *tests* directory of the package. I will run the testing using `library(testthat); test_package('ars')` (please check that you can run that successfully before submitting).

   (c) Auxiliary functions used by the primary function and testing function,

   (d) Help information for the main function, in a file *ars.Rd* in the *man* directory.

   I should be able to install the package using `devtools::install_github('student_name/ars')`.

For a given subtask of the problem, if you find good code available, you may use it as a modular component of your code provided it does not constitute too large a part of your solution and provided you test the code. Consult me with questions on this matter.

You should use Git and Github to manage your collaboration. Please have the project be a repository within the Github account of one of the project members.

You should start the process by mapping out the modular components you need to write and how they will fit together, as well as what the primary function will do. After one person writes a component, another person on the team should test it and, with the original coder, improve it.

A good starting place for information about R packages is Hadley Wickham's book: http://r-pkgs.had.co.nz/.