Statistical Computing and Learning

Goal: Provide PhD level statistical, computational and machine learning tools for research in core statistical methodology. The focus is on inferential issues through hands-on exercises.

Professor: Edoardo M. Airoldi (airoldi@fas.harvard.edu)
Lectures: Mondays & Wednesdays, 2:30–3:50pm, in Science Center 221.
Teaching Fellow: Zhirui Hu (zhiruihu@g.harvard.edu)
Section Hours: TBD
Office Hours: By appointment (Prof. Airoldi) and TBD (Hu)
Course Webpage: https://canvas.harvard.edu/courses/14516
Prerequisites: Linear algebra, Statistics 111 or 120, and working knowledge of the R computer programming language required; Statistics 220 (concurrently) recommended.
Note: Programming assignments will apply the methods discussed in class and beyond.

Tentative Topics: Multivariate analysis basics, random number generation, optimization methods, numerical integration, Monte Carlo methods including Metropolis-Hastings and Gibbs samplers, approximate inference techniques including Expectation-Maximization algorithms, Laplace approximation and variational methods, data augmentation strategies.
Recommended Readings: *Pattern Recognition and Machine Learning*, by Bishop, 2007. *Numerical Analysis for Statisticians*, by Lange, 2000. *Monte Carlo Statistical Methods*, by Robert & Casella, 2004 (2nd ed.). A series of articles will be posted online.

Grading Scheme: problem sets (70%), final project (30%).
Problem Sets: Problem sets serve multiple purposes. They aim at developing your intuition about concepts covered in class in the context of simple problems. They expose you to computational and technical issues that researchers face in the context of real problems. They will force you to think critically about published analyses as you attempt to replicate small portions of them. The problem sets include theoretical, programming, and data analysis problems. We anticipate 6 problem sets and a final project. Problem sets are graded largely on getting the right answer or getting the program to work. Questions asking for exploration and experimentation will be graded more subjectively; ideal answers are thoughtful, perceptive, critical, clear, and concise.
Project: The class project is essentially a structured take home exam, equivalent to 1–2 problem sets. The project is an opportunity for you to explore a problem that you have

have encountered in one of the problem sets more in depth. Instructor and TF will advise you throughout the project, but the final responsibility to define and execute details of the analysis is yours. Your project is worth 30% of the final grade, and consists of two deliverables: a written report (typically a minimum of 25 pages of text, plus figures and tables) and working code. Latex templates and coding guidelines will be provided. We will consider allowing teams of two (or more) students for the project, depending on enrollment.
Class Blog: We will have a class blog, which you will be able to access as soon as you are officially enrolled in the course. The blog is a placeholder for questions and answers to common problems. It is the policy of this course that every question be asked on the class blog. The effort of formulating the question correctly is usually beneficial, and the TF and I may engage students in discussing the formulation of the question before getting to the point when it is possible to answer. This process is fun and helps us maintain a record of the computational and conceptual issues students encounter in the learning process.

Late Days: All assignments are due at 11:59pm on the due date; email your solutions to the professor or to the TF. Each student will be allotted three late days which can be used to turn in homework assignments late without penalty. For instance, you might choose to turn in the first homework two days late, and the third homework one day late. Once your free days are used up, late homeworks will be penalized according to the following policy: (i) Homework is worth full credit if submitted on time; (ii) It is worth half credit for the next 24 hours; (iii) It is worth zero credit after that. *You must turn in all the problem sets, even if for zero credit, in order to pass the course.* Exceptions to these rules will of course be made for serious illness or other genuine emergency circumstances, and late days should not be used for these purposes; in these cases, please contact a professor as soon as you are aware of the problem. The final project cannot be turned in late.

Collaboration: The collaboration policy for this course is based on the overarching objective of maximizing your learning experience, that is, what you gain in knowledge, understanding and the ability to solve problems. Obviously, you do not learn anything by copying someone else's solution. Forbidding any and all discussion of course material, on the other hand, may deprive you of the opportunity to learn from fellow students. The middle ground between these two extremes needs to be defined with the following basic principle in mind. Before working with another student, you should ask yourself if you would gain more or less by working together or individually, and then act accordingly. Some specific guidelines:

1. You are encouraged to talk to others about the material and for general help with R.

2. Before working with someone else, you should first spend a substantial amount of time trying to arrive at a solution by yourself. Easier problems, including many or most of the written exercises, should be solved individually from start to finish.

3. Discussing harder problems or programming assignments with fellow students is allowed to the extent that it leads all participants to a better understanding of the problem and the material. Following such discussions, you should only take away your understanding of the problem; you should not take notes, particularly on anything that might have been written down. This is meant to ensure that you understand the discussion well enough to reproduce its conclusions on your own. *You should also note on your solution who you worked with.*

4. Needless to say, simply telling the solution to someone else is prohibited, as is showing someone a written solution or a portion of your code. Comparing code or solutions also is not generally permitted. However, comparing and discussing the results of experiments is okay if done in the spirit of the guidelines above.

In any event, all writing and programming must be done strictly on your own. Copying of any sort is not allowed. Unless instructed otherwise, you may not use code or solutions taken from any student, from the web, from prior year solutions, or from any other source.