Stat 650/750 Course Description and Syllabus Fall 2017

Vital Info

Class Schedule	TR 12-1:30PM, GHC 4307
Instructors	Christopher Genovese 232E Baker Hall (x8-7836) genovese@cmu.edu Alex Reinhart areinhar@stat.cmu.edu
Office Hours	To be determined empirically and by appointment
TAs	Taylor Pospisil, Nicolas Kim, Justin Hyun
Web Page	`https://github.com/36-750` (See below for more on github.)
Text	Class notes and handouts

Course Objectives and Scope

Computing is an essential—and increasingly important—part of statistical practice and methodology, but computing's role in the typical graduate Statistics curriculum (including ours) has not been commensurate with its importance. For most students, the primary opportunity to develop computing skills comes during research projects, but this tends to put completion above learning and tends to cover ideas that are focused on the particular needs of the project. While students may be able to take a computing-related course here or there, it is rare for such a course to cover the fundamental concepts in computing and develop critical skills in a way that will pay off during later research, work experience, and beyond. This course, 36-650/750, aims to fill that gap.

The premise of this course is that building a broad and solid foundation in computing will pay significant dividends throughout a student's research career. We will focus on four main themes: (a) effective programming practices; (b) fundamental principles of software design; (c) important algorithms, data structures, and representations; and (d) essential tools and methods. Along the way, we will also consider a variety of applications and techniques that are important for statistical practice. However, the focus of the course is not on specific statistical techniques per se. Similarly, although you will practice new tools and approaches to programming, the purpose of this course is *not* teaching you to program.

A second premise of this course is that practice is the key to developing strong computing skills. To this end, the work will consist of many program-

ming tasks of various sizes, both between and during class. You will have access to a repository of exercises varying in complexity so that you can target your work to your experience level, and you will have an opportunity to work interactively with your peers. Prior programming experience is not a requirement, but if you have not programmed before (or much), you will be expected to work toward learning a chosen language. This course is, for the most part, language agnostic, and indeed, you will be asked to do several (simpler) tasks in a language outside your comfort zone to gain the valuable perspective that this offers.

By the end of this course, you should be better able to:

- develop correct, well-structured, and readable code;
- design useful tests at all stages of development;
- effectively use development tools such as editors/IDEs, debuggers, profilers, testing frameworks, and a version control system;
- build a small-to-medium scale software system that is well-designed and that facilitates code reuse and generalization;
- select algorithms and data structures for several common families of statistical and other problems;
- write small programs in a language new to you.

Class Mechanics

Classes will feature a combination of lectures, interactive discussions, and (single and group) programming activities. We will often discuss, edit, and run code from a variety of sources, and we will do a fair amount of real-time programming. Hence, ***you should bring your laptop to every class***.

We will have mechanisms for sharing code snippets interactively and turning in homeworks interactively. Further details are described below in Assignments. For this purpose, sign up for a (free) account at `github.com` and then visit `https://classroom.github.com/a/MMfm3qKz` to get your working repositories for the class. (You do not need to sign up for the educational discount, since it's all under our organization.)

Participation and attendance are important parts of the class and as such will constitute a nontrivial portion of your final grade.

Resources

Class notes will be made available on github shortly after class in the `Lectures` subdirectory of the `documents` repository. In addition, there will be occasional handouts and readings, all of which will be available within the same repository.

There is no primary text for the class, but several useful (though not

required) books are available for your use from Professor Genovese. This includes several general books on coding and algorithms along with some language-specific guides for particularly recommended languages. A list of these books along with other online resources will be available in the documents repository.

The instructors will hold regular office hours at times to be announced in class. The TAs will also be available to answer questions.

Assignments

The main work in this course consists of programming and related exercises. There will be three main types of exercises:

- Stand-alone. These are short, self-contained tasks that illustrate or build upon an idea we have discussed in class. Many of these involve writing a short program, but others will involve some related task (e.g., debugging, reasoning).
- Vignettes. These are groups of exercises around a central theme, data set, or idea. Working on these exercises in succession is intended to help you understand the central idea more deeply. Within a vignette, it is recommended that you do all the exercises that are not marked optional, though this usually takes place over several assignments.
- Challenges. These are larger programming tasks that integrate several different ideas and skills. There will be two challenges assigned over the semester.

All of these exercises will be available in the course repository on GitHub.

Short assignments will be graded on a simple rubric as either "Mastered" or "Not yet mastered". Challenges can also be graded as "Sophisticated". The criteria for these levels will be specified in rubrics that will be posted in the documents repository.

Using the facilities on github, the TAs will be able to review your submissions and provide feedback, with comments both in general and on specific lines or sections of your code. Because revision is an important skill for you to practice, you will have the opportunity to revise your assignments to address issues found during code review. Each assignment and project may be revised at most twice. We will describe in detail the procedure for requesting a review by the TAs of an original or revised submission. Note, however, that the TAs review is contingent on meeting the basic requirements of the submission. For example, if required tests or scripts are missing or if there are significant deficiencies with coding style and organization, the review will stop at noting that issue and you will have used a submission/revision for little gain.

We will give you as much latitude as possible in selecting which exercises to do from the repository. The exercises in the repository cover a wide range, both in topic and complexity, and we want you to challenge yourself in selecting exercises that expand your knowledge and skill. At times, however, specific exercises will be assigned or exercises will be constrained to be selected from a particular subset of those available to fit with work we are doing in class. In addition, the exercises will be grouped into four broad categories, and *roughly* one-quarter of your assignments should be drawn from each of these categories.

Finally, this course is (nearly) language agnostic: you can use as your primary programming language any reasonable one that the instructors or TAs have worked with, which covers a very wide range. (If you have any questions as to suitability, ask Chris or Alex.) In addition, we believe strongly that getting experience with multiple languages helps broaden your perspective on thinking about problems, so we require that you do *at least two* assignments (simpler ones, if desired) in a language that is new to you. We will offer recommendations in class and will describe how to indicate the assignments that meet this requirement in your submissions.

Grading

Course grading is based on the rubric mentioned above. Rather than a points-based scheme, we will use a simple set of requirements to set your grade. The semester will have two scheduled Challenge projects; assignments can be submitted ***at most twice per week***, including revised assignments, so do not procrastinate. (There are fifteen weeks, so you have 28 opportunities to submit, not including the first week. You will be working on the Challenges for several of these weeks.) The TAs will grade the first two assignments or revisions submitted each week, leaving any excess submissions to be graded in subsequent weeks.

Your final grade is calculated by combining a base grade with three possible adjustments. The base grade is calculated from how many assignments and challenges you receive Mastered and Sophisticated on. See Table 1 below.

The possible adjustments are:

1. The grade can be adjusted upwards a step for students who demonstrate outstanding participation in class; it can be adjusted downwards up to two steps for poor participation or attendance.

2. The grade will be adjusted downwards one step for those not meeting the second-language requirement.

3. The grade will be adjusted downwards one step for those not achieving a rough balance of assignments across the four broad categories.

Grade	Requirements
R	Fail to meet requirements for D
D	Earn Mastered on at least 8 assignments
C	Earn Mastered on at least 10 assignments
B	Earn Mastered on at least 10 assignments **and** one Challenge
B+	Earn Mastered on at least 12 assignments **and** one Challenge, **or** earn Mastered on at least 10 assignments **and** earn Sophisticated on one Challenge
A-	Earn Mastered on at least 12 assignments **and** both Challenges
A	Earn A- **and** earn Sophisticated on one Challenge
A+	Earn A- **and** earn Sophisticated on both Challenges

Table 1. How the base grade is calculated

Policies

Email. When sending an email, please put either "[650]" or "[750]" at the beginning of the subject line, so that we can easily identify the message. Also, please be advised that merely sending email ***does not eliminate your responsibility for completing assignments on time***.

Collaboration, Cheating, and Plagiarism. Discussing assignments with your fellow students is allowed and encouraged, but it is important that every student get practice working on these problems. This means that ***all the work you turn in must be your own***. You must devise and write your own solutions and carry out your own tests. The general policy on homework collaboration is:

1. You must first make a serious effort to solve the problem.
2. If you are stuck after doing so, you may ask for assistance from another student. You may discuss strategies to solve the problem, but you ***may not look at (or have dictated to you) their code***.
3. Once you have done so, you must write your own solution individually.

This also applies in reverse: if someone approaches you for help, you must not provide it unless they have already attempted to solve the problem, and you ***may not share your code***.

In addition, you ***should not consult on-line or other sources*** that discuss solutions to problems related to those in the homework. You may refer to programming language documentation and resources for how to

write your code, but you must independently solve the problem assigned, *not* translate a solution presented online or elsewhere.

Please talk to us if you have any questions about this policy. Any form of cheating is typically grounds for course failure. We are obliged in these situations to report the incident to the appropriate University authorities. Please refer to the `University Policy on Academic Integrity (link)`.

Disability Resources. If you have a disability and have an accommodations letter from the Disability Resources office, we encourage you to discuss your accommodations and needs with us as early in the semester as possible. We will work with you to ensure that accommodations are provided as appropriate. If you suspect that you may have a disability and would benefit from accommodations but are not yet registered with the Office of Disability Resources, we encourage you to view the online guidelines (link) and/or to email `access@andrew.cmu.edu`.

Wellness. Course work at this level can be intense, and we encourage you to take care of yourself. Do your best to maintain a healthy lifestyle this semester by eating well, exercising, avoiding drugs and alcohol, getting enough sleep and taking some time to relax. This will help you achieve your goals and cope with stress.

All of us benefit from support during times of struggle. If you are having any problems or concerns, do not hesitate to come speak with either of us. There are also many resources available on campus that can provide help and support. Asking for support sooner rather than later is almost always a good idea.

If you or anyone you know experiences any academic stress, difficult life events, or feelings like anxiety or depression, we strongly encourage you to seek support. Counseling and Psychological Services (CaPS) is here to help: call 412-268-2922 and visit their website at `http://www.cmu.edu/counseling/`. Consider also reaching out to a friend, faculty member, or family member you trust to help get you the support you need.

Policy Updates. Updates to policies and course information will be posted in updated versions of this syllabus and announced in the course `ANNOUNCEMENTS` file in the github `documents` repository.