ACM Data Science Task Force Course Example

COMP 180: Computing and Data Analysis for the Sciences
Loyola University Chicago, Chicago IL
Ting Xiao
ting.xiao@unt.edu

Knowledge Areas that contain competencies (knowledge, skills, and dispositions) covered in the course

Knowledge Area	Total Number of Contact Hours
Data Visualization	7
Data Structure	2.5
Data Frames	2.5
Image Processing	2.5
Machine Learning	16

Where does the course fit in your undergraduate Data Science curriculum?
This course is part of a major but not compulsory. It doesn't have pre-requisites or required following courses. About 20 students take the course every semester. The distribution of majors in the course represents the role as a course for science students without traditionally extensive computing foundations, predominantly neuroscience and psychology. The distribution of student years was skewed toward junior and senior students by credit hours, although students were present in all four years. It is worth noting that this course had a majority of women in each semester with 63.5% female representation on average.

Is this course from or used in other curricula/majors?
This course was designed as a first introduction to computing for students interested in the physical or social science. It was adapted by University of North Texas as a graduate course in the Department of Computer Science and Engineering: CSCE 5300: Introduction to Big Data and Data Science. More than half of the materials of CSCE 5300 are from COMP 180.

What is covered in the course?
The course covers data visualization, data frames, image processing, classification and regression. Applications of these topics are not only presented in the form of assignments and quizzes, but also through projects, research presentations, and data analysis competitions. The initial focus on data visualization provides a more appealing demo-driven introduction to the course before the necessary survey of data structures afterward. Data frames are emphasized, since 2D tabular data structures are quicker for students to adopt on the time scale of this course. The image processing task is to classify images based on color histograms, first by hand using if/then statements and then later seeing the way in which k-nearest neighbors performs the same task. The first full lecture in the course is for exposure to the concepts and

practice of machine learning. Project selection is generally biased toward classification or regression to give more first-hand experience after their initial, relatively short exposure. After project work, advanced data analysis topic presentations by the instructor or colleagues are provided. Finally, the last two weeks are reserved for a class-wide automated predictive modeling competition in the style of Kaggle.com with real-time leaderboards and scoring of models. During the competition phase, exercises are given to recap material in the course in a format similar to what will be seen on the final.

What is the format of the course?
It is face to face with about 30 contact hours. It has lectures (first 10 minutes), lab sessions (last 40 minutes) and discussion classes. It is an application-oriented course. Reading materials and theory is minimal. Code and code-based tutorials is emphasized, as well as practice over theory, with assignments providing much of the content and expectations in the course. Students are encouraged to use search engines and given tips and techniques.

How are students assessed?
Students are evaluated in class on a weekly basis in a manner similar to assignments — data analysis tasks using a computer, but under time pressure and without help from others. Students are strongly encouraged to attempt to solve the tasks iteratively and incrementally — writing code that works first, but works poorly, and improving from there, rather than writing perfect code from beginning to end. When finished with a quiz or an exam, students are also encouraged to use the extra time to organize their materials to become more efficient at data analysis tasks similar to the assignments. All quizzes are due at the end of the class, with an analysis notebook submitted to the course management system.
A few weeks into the course students select among a small number of collaborative projects. This begins with a task for individual one-paragraph descriptions of desired projects which, after classification and regression is introduced, are often sufficiently complex for a multi-week project effort. Students are encouraged to reach out to people who may have data that could be analyzed on their behalf. Groups are 2–3 people, strictly not allowing individuals to work alone in order to encourage collaboration. After the projects are selected the progress is encouraged and tracked through initial group project proposals, in class oral updates, project presentations, and short, written final reports with the intention to provide a showcase for their efforts.

Course tools and materials
No textbook is required. All materials (readings, videos, tutorials, assignments, quizzes, and exams) will be accessible online and posted on the course calendar on the respective class day at the latest. The task-driven nature enabled the mutual development of resources for each of the top three requested languages, Python, R, and MATLAB, allowing students to concurrently complete the course in any of the three languages. Tutorials in all three languages were produced to prepare students for the language-independent task-driven assignments given each day. For consistency all tutorials and assignments use the Jupyter notebook cell-based workflow. Python and R are freely available. MATLAB is not free but a free version of MATLAB, Octave, is used in this course.

Why do you teach the course this way?
I am a firm believer in SMART (Specific, Measurable, Achievable, Relevant, Time-bound) goals. In my experience, my most intense learning efforts were a product of well-specified goals and measurable results along the way. I also aim to have my students experience the same benefits of intense, well-structured learning. I believe that frequent assessment and precise feedback are critical for an effective, long-term learning experience. Based on this belief, I designed COMP

180 from scratch. With a half-year of lead time, I was able to organize three students to create tutorial material in three data analysis languages (Python, R, and MATLAB) and have them report to myself and two other faculty in the process. The design worked well. In a class where many students admitted to having little computer literacy, I was admittedly a bit worried about having them on the computer every single day for tutorials, assignments, quizzes, and exams. It was challenging for them at first, but toward the end they all appreciated the "hands-on" approach. I think most of them now feel they can "speak" the programming language they chose. At the end of this course, students are well versed in the use of an interactive environment for data analysis (Python, R, or MATLAB), with an ability to manage the collection of data, create automated processes for analysis, use collaborative tools, and rapidly report quantitative findings.

Body of Knowledge coverage

KA	Sub-domain	Competencies Covered	Hours
AP	Visualization	Gain familiarity with the main strands of knowledge underpinning approaches to Analysis and Presentation Provide the range of skills and techniques that can be employed in addressing the challenges of analysis and presentation and creating efficient and effective interfaces	7
ML	Supervised learning	Appreciate the breadth and utility of machine learning methods, compare and contrast them, select appropriate methods for specific problems. Apply machine learning algorithms following appropriate training and testing methodology. Exhibit knowledge of methods to mitigate the effects of overfitting and curse of dimensionality in the context of machine learning algorithms. Provide an appropriate performance metric for evaluating machine learning algorithms/tools for a given problem.	16
PDA	Data structures	Write clear and correct code in a programming language that includes primitive data types, references, variables, expressions, assignments, I/O, control structures, functions, and recursion. Use standard libraries for a given programming language. Select appropriate data structures for a given problem.	2.5

Other comments

I have written a paper to describe this course in more detail.