HIST 698 ❧ GEORGE MASON UNIVERSITY ❧ FALL 2014

Clio 3: Programming for Historians

Meets in TBD on Wednesday, 7:20–10:00 p.m.

Lincoln Mullen
E-mail: lincoln@lincolnmullen.com
Website: http://lincolnmullen.com
Office: TBD
Office hours: TBD and by appointment

DRAFT SYLLABUS

Course description

This class will teach you how to use computer programming for research in history. This class is not organized around the features of a particular programming language or the theoretical concepts of programming, as a class in a computer science department might be. Rather it is organized around three of the common practices (and four of the common programming languages) that digital historians use to do their work.

We will begin by learning the principles of the Unix operating system which undergirds almost all research programming. Our first major section will be on **data analysis**, in which you will use the **R language** to analyze historical data both quantitatively and geographically. You will also learn how researchers structure, manipulate, and clean their data. At the end of this section you will translate one of your visualizations into an interactive, online version using **Javascript**, the lingua franca for interacting with users over the web. Our next major section will be on **scripting for research**, using the elegant and expressive language **Ruby**. We will begin by using Ruby to access APIs and scrape web documents for research. Then we will use Ruby to create our own simple **web applications** and to interact with relational databases. Finally we will move on from Ruby to **PHP**, a commonly used language for web applications like Omeka and WordPress. We will use

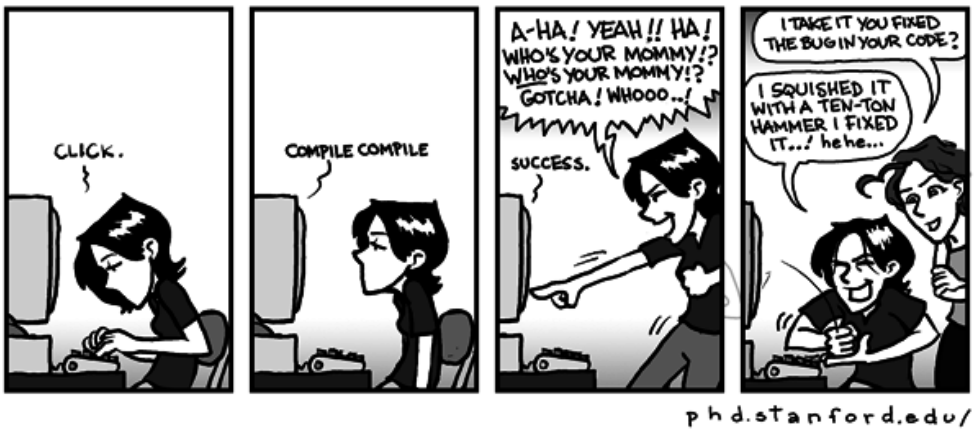
Omeka as our example of a large-scale digital humanities tool. You will learn how Omeka plugins work and make your own plugin as your final project.

This course emphasizes breadth for three reasons. First, digital humanists tend to be polyglot when it comes to tools and programming languages. Second, often the best way to gain a deeper understanding of one language is to learn a second language. By comparing multiple languages you will gain a knowledge of the basic structures of computer programming. Third, picking up several languages will teach you how to learn a computer language. In my experience, once you've learned three or four programming languages it becomes much easier to pick up the next language.

Since this is a research class, students are expected to come to the course

with an active research agenda and to know of or soon find a body of primary sources susceptible to computational methods. As much as possible, you should try to use the assignments in this course to advance your research, especially for your dissertation. You should expect to come out of this course a better-equipped historian. While this course alone will likely not be sufficient to turn you into a digital humanities software developer (for that you'll need a deep immersion in a particular language or two) you will gain a familiarity with the basics of computer programming as applied for humanities research. Using this base, you can go on to become more fluent as a digital humanities developer should you wish.

This course does not presume any prior experience with computer programming. However the course does assume that you have taken Clio 1 and Clio 2, and thus possess the skills taught in those courses. Make no mistake, this is an advanced digital history course and you will likely find the material very difficult, even alien at first. Nevertheless, I will make the commitment to get or give you all the help that you need, and (contrary to a common discourse about computer programming) there is no intrinsic reason why you should not do well in this course.



Learning goals

After taking this course, you will

- be familiar with R, Javascript, Ruby, and PHP, along with some of their most useful libraries, such as ggplot2, dplyr, jQuery, D3, Sinatra, and Nokogiri;

- understand the common concepts of computer programming across computer languages;
- be able to read documentation and navigate online aids such as Stack Overflow in order to learn programming languages for yourself;
- perform reproducible data analysis both quantitatively and geographically;
- programmatically access APIs and data for your research;
- understand the basics of how web applications are built; and
- contribute code back to digital humanities projects.

Assignments

Participation. Your most basic assignment is to come to class prepared. Each week you will have to read online tutorials or documentation about the programming technique we will be learning. When the syllabus says to "read" an assignment, that means to read the tutorial or documentation with a text editor and terminal open, doing your best to get the code to work. You are not expected to come to class having mastered the topic, but you are expected to come having tried your hand at it. No later than four hours before class, you should e-mail the course list about one thing you think you learned and one thing you didn't understand. I will use your e-mails to tailor my explanations in class. Often in class, I will ask you to work your way through some programming problem on your computer and submit your solutions to GitHub, though these solutions will not be formally graded. All of these kinds of participation will factor into your course participation grade.

Tutorial. You will write one programming tutorial that meets the submission guidelines for *The Programming Historian 2*. This tutorial should explain how to perform some research technique in history, using any of the languages or technologies that we will learn. You should post this tutorial to your own blog. After revision you may wish to submit it to the *Programming Historian 2* to go through its peer-review process. This assignment will be due by the end of the semester, though you can and probably should submit it earlier.

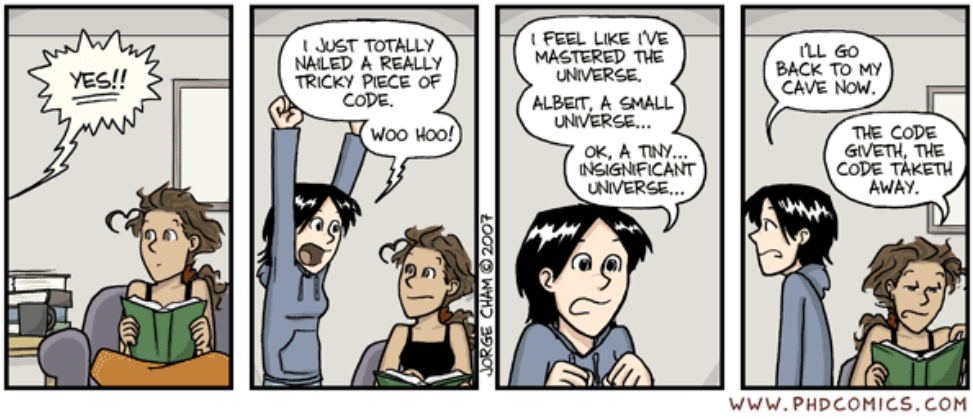
Lab notebook. At the end of our data analysis section you will submit a lab notebook in R that performs some kind of analysis of use to your historical research. This notebook should perform an exploratory data analysis, mining everything of possible interest from your data. For example, you should try mapping, visual analysis, charts, and summary statistics. This notebook

should also contain prose about the historical insights that you have learned.

Web visualization. You will create an online visualization in Javascript and D3 of some of the fundamental insights gained in the lab notebook section. This visualization will take the form of a website with interactivity and written analysis to go along with the visualization.

API script. You will submit code in Ruby which uses a web API to gather sources into usable data.

Omeka plugin. You will spend the second half of the semester creating a plugin for Omeka. This plugin's README should contain a brief essay explaining the scholarly purpose for the code. (At your option, you may substitute a comparably advanced program in any language for any purpose that would be more useful for your work. For example, you could create a web application or a WordPress plugin, or write Ruby gem, or create a more advanced web visualization. Please make arrangements with me well in advance.)



Evaluation

The assignments for this course will receive these weights.

assignment	weight
class participation/coding	40%
tutorial	10%
lab notebook in R	10%
web visualization in Javascript	10%

assignment	weight
API script in Ruby	10%
Omeka plugin in PHP	20%

A few notes

This course assumes that you are programming in a Unix-like environment, namely some Linux distribution or the Unix underpinnings of Mac OS X. You're free to use Windows if you like, but I won't support you in figuring out the differences, and your code will have to run on my machine.

Configuration is the bane of all computer projects, and to that end all code must run on a standard machine. This machine image will be available to you using Vagrant, and you'll be able to run it as a virtual machine on your own computer. You should test all code in the virtual machine before submitting it.

To submit your code, you will need a GitHub account, and must let me know your user name. (Your account can be anonymous/pseudonymous to the world, if you wish, just not to the people in this course.) Each assignment for the course should have its own repository, and each repository should have a tag, `submitted`, for the version of the assignment you wish me to grade.

This course will not teach you the tips and tricks of the innumerable tools that surround programming, such as IDEs (Integrated Development Environments), text editors, syntax checkers, environment managers and so on, though I can provide suggestions if you need them. Use whatever you want. If you don't have a preferred text editor, you could do worse than Sublime Text or TextWrangler. I prefer Vim.

The culture around programming can be extraordinarily generous, with many people sharing their work and expertise for free. It can also be extraordinarily toxic, especially for women and minorities. Part of this course will be learning to help yourself in the culture of programming, including indispensable sites such as Stack Overflow. If you get stuck in the more nefarious parts of the culture, come ask for help.

In any case, come talk with me early and often!

Course Schedule

Week 1 Unix

Read Mike Gancarz, *Linux and the Unix Philosophy*.

Read Jerry D. Peek, *Learning the Unix Operating System*, chs. 1, 3–5.

Experiment with your terminal, Bash, and Unix conventions.

Week 2 Git and GitHub

Try GitHub's online, interactive tutorial for Git. (This is probably the most fun way to learn the basics of Git.)

Read the documentation for GitHub and their tutorials

Use Scott Chacon, *Pro Git*, especially chs. 1–3, 5, for reference.

Read William J. Turkel and Adam Crymble, "What to Do If You Get Stuck," *The Programming Historian* 2.

Experiment with Git and GitHub.

Week 3 Data visualization in R

Watch Google Developers' introduction to R.

Read Winston Chang, *R Graphics Cookbook*, appendix A, chs. 1-4.

Browse ggplot2 documentation.

Experiment with ggplot2 in R Studio.

Week 4 Mapping in R

Read James Cheshire, "Introduction to Spatial Data and ggplot2," Spatial.ly, December 9, 2013.

Read David Kahle and Hadley Wickham, "ggmap: Spatial Visualization with ggplot2," *The R Journal* 5, no. 1 (June 2013): 144–61.

Browse documentation for RGDAL.

Experiment with spatial data and shapefiles with ggplot2/ggmap.

Week 5 Data cleaning and tidying in R

Read Hadley Wickham, "Tidy Data," *Journal of Statistical Software* (forthcoming).

Read Hadley Wickham, "Reshaping Data in R," *Statistical Computing and Graphics* 16, no. 2 (December 2005): 5–8.

Read Hadley Wickham, "The Split-Combine-Apply Strategy for Data Analysis," *Journal of Statistical Software* 40, no. 1 (April 2011): 1–29.

Watch Hadley Wickham, "Tidy Data and Tidy Tools," NYC Open Statistical Computing Meetup, December 2011.

Read Seth van Hooland, Ruben Verborgh, and Max De Wilde, "Cleaning Data with OpenRefine," *The Programming Historian* 2.

Browse documentation for reshape2, dplyr, plyr.

Experiment with reshaping, tidying, and cleaning your own data.

Week 6 Javascript for web interaction

Read Douglass Crockford, *Javascript: The Good Parts*.

Browse the beginning tutorials at jQuery Learning Center.

Experiment with jQuery in your browser.

Due: Lab notebook in R.

Week 7 Interactive web visualizations using D3

Read Scott Murray, *Interactive Data Visualization for the Web*.

Browse the D3 documentation.

Experiment with the D3 examples here and here.

Week 8 Object oriented programming in Ruby

Read David A. Black, *The Well-Grounded Rubyist*, chs. 1–4.

Due: Web visualization in Javascript.

Week 9 Flow control and data structures in Ruby

Read David A. Black, *The Well-Grounded Rubyist*, chs. 6–10.

Week 10 Using APIs in Ruby

Read David A. Black, *The Well-Grounded Rubyist*, ch. 11.

Read Jason Heppler, "Better Web Scraping with Nokogiri," October 12, 2012. You may find Jeri Wieringa, "Intro to Beautiful Soup," *The Programming Historian 2*, helpful on the same research technique using the equivalent library for Python.

Read Adam Crymble, "Downloading Multiple Records Using Query Strings"; Ian Milligan, "Automated Downloading with Wget"; and Kellen Kurschinski, "Applied Archival Downloading with Wget," all in *The Programming Historian 2*.

Browse documentation for Nokogiri, JSON module, REST wrapper.

Experiment with Ruby and an API for historical sources.

Week 11 Basic web applications using Sinatra in Ruby

Read Alan Harris and Konstantin Haase, *Sinatra: Up and Running*; you may also find *Sinatra: The Book* helpful.

Browse Sinatra documentation.

Experiment with creating a mini web application using Sinatra.

Due: API script in Ruby.

Week 12 Databases and SQL

Read *Sam's Teach Yourself SQL*, lessons 1–6, 9–10, 12–13.

Read "Singing with Sinatra" parts 1–3.

Browse sqlite3 gem documentation and Datamapper gem documentation

Experiment with sqlite3 and Ruby.

Week 13 PHP

Read all of "PHP Basic" and "PHP Database" from W3 Schools tutorials, and browse the remainder.

Experiment with PHP and a MySQL database in the development environment.

Week 14 Anatomy of an Omeka plugin

Read "Plugin Basics," "Essential Classes in Omeka," and browse the Omeka documentation.

Experiment with installing Omeka and the plugin of your choice, then make an attempt at understanding what the plugin does.

Week 15 Your first Omeka plugin

Come to class with as much of your Omeka plugin (or other project) working as possible. Be prepared to explain both your code and the scholarly rationale behind your plugin in our class workshop.

Last day of instruction

Due: Tutorial

Due: Omeka plugin

Policies

Assignments

I may change due dates or assignments. I will always give you plenty of notice of changes, which will always be intended for your benefit.

Students must satisfactorily complete all assignments (including participation assignments) in order to pass this course.

Boilerplate TBD

Acknowledgments

The following comic strips are used by permission:

- "Regular Expressions," *XKCD* 208.
- Jorge Cham, *PHD Comics*, April 28, 2000.
- Jorge Cham, *PHD Comics*, November 30, 2007.
- Scott Adams, *Dilbert*, June 24, 1995.

I am grateful to Stephen Ramsay for sharing his syllabus for "CS1 Humanities" with me.

License