Practical Problems in Teaching Digital History

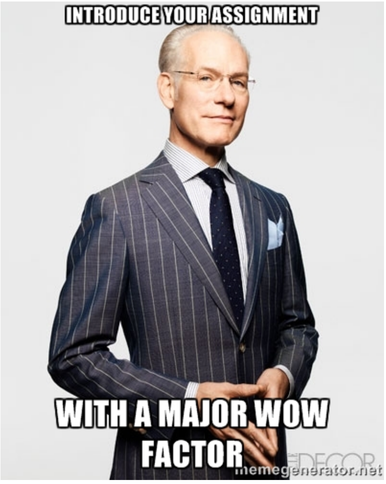
Lincoln Mullen
lincoln@lincolnmullen.com

Let's take it for granted that you know why you want to add a digital history assignment to your class. What specifications should your assignment have? What kind of problems are you likely to encounter in the classroom?

Characteristics of a good digital history assignment

A good assignment will have many of the following characteristics:

- It will result in historical insight tied in to the course material. This insight will probably come from an argument or a narrative that students build for themselves.
- It will ask students to mix traditional historical research with digital work.
- Students will have ownership over their part of the work, but their part will contribute to a sum that is greater than the whole.
- The assignment will start with a major wow factor, but the assignment will go on to teach mechanics of digital work and critical thinking about, e.g., data and visualizations.
- The assignment should be tool agnostic: students should learn the principles of digital work instead of which levers and knobs to operate on the Google dingus that is about to be canceled.
- The assignment must carefully distinguish between necessary complexity (which requires scholarly insight) and incidental complexity (putting up with the cruft around technology). Whatever incidental complexity can be eliminated, must be eliminated.
- Students will have the opportunity for safe, successful failure, possibly through scaffolding or repeated assignments that permit practice.
- The best assignments will be adapted from someone else who has already taught them several times.



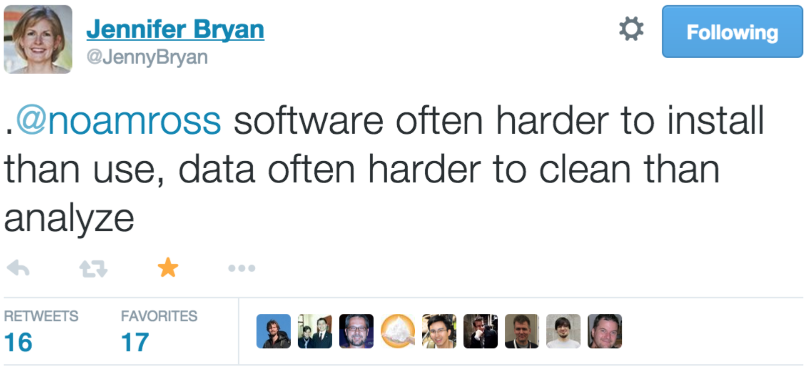
A excellent model for a DH assignment is Brian Croxall's "Mapping Mrs. Dalloway."[1] Also see the assignments for Jeff McClurken's many digitally-inflected courses.[2]

Potential problems in teaching digital history

These are some problems that I have run into. It is your responsibility to free your students from incidental complexity so that they can

[1] http://www.briancroxall.net/dh/assignments/mapping-mrs-dalloway/

[2] http://mcclurken.org/



Jennifer Bryan
@JennyBryan

.@noamross software often harder to install than use, data often harder to clean than analyze

Figure 1: The root of evil when teaching DH. https://twitter.com/JennyBryan/status/577977718907613185

focus on necessary complexity. Treat your students with endless respect. You will be amazed at how much better they are at learning higher-order concepts than at dealing with petty annoyances.

Students are not interested in technology

You cannot presume that students are interested in learning with technology. If anything, presume that they are slightly less interested in digital history than they are in history. It is your job to justify and motivate the assignment.

Suggestions: Interactive visualizations are crowdpleasers. It's even better if the interactive visualization explains something students already learned in class better than you explained it. Emphasize the fun parts.

Students are anxious about technology

Students are often anxious about working with technology in history, in part because tech is coded mathematical, male, and young, and in part because they haven't encountered DH before and don't know what you're expecting.

Suggestions: Stay calm in class, even when things go wrong. Plan on being even more of a cheerleader for digital assignments. Point out small wins. Plan a low-stakes practice assignment. Assign a blogging assignment where you can reward reflective failure.

Students cannot distinguish between incidental and necessary complexity

Students cannot distinguish between complexity you care about teaching (e.g., the interpretation of a map) and the complexity we put up with in order to get to the real work (e.g., how to install QGIS).

The boundary between incidental and necessary complexity depends on what you want to teach. In some classes, teaching data joins with geospatial data is necessary complexity; in most it will be incidental complexity.

They can, however, tell the difference between work that is getting them somewhere and work that is not.

Suggestions: Eliminate every bit of incidental complexity. No really, every little bit. Teach only using clean, tidy data. Avoid changing any options or preferences if at all possible. Avoid installing software. Don't encourage students to change things like WordPress themes. Don't demo anything you haven't explored before class.

WordPress.com sucks. Roll your own.

Students have different configurations

Students will come to class with a bewildering array of operating systems, tablets, and phones. More and more, technology companies treat their users like they are stupid. File systems are abstracted away on computers and completely inaccessible on tablets and phones. Operating systems no longer include a plain text editor suitable for working with HTML.

I'm looking at you, Apple Inc.

Suggestions: For most assignments, you will have to work with the lowest common denominator—the web browser. Thankfully, that's not much of a limitation. Always demo assignments using a technology stack as close as possible to what your students are using. For longer assignments, consider using a lab or a virtual machine. Vagrant[3] lets you script VM setup. Prefer professional-grade tools, which expose their inner workings and can usually be scripted instead. If you wouldn't use it to do real research, don't expect your students to get real work done with it.

In graduate classes I refuse to support Windows users or, for that matter, the non-Unix parts of Macs.

An example of the hierarchy of tools for mapping: R > QGIS > ArcGIS web > CartoDB > paper and pencil > Google Maps.

It is difficult to follow along in class

It is very hard to follow along with technology demos.

Suggestions: Demo everything at least twice. Don't forget to show the goal. Develop a plan for helping students who have run into difficulties or fallen behind without slowing down the entire class. I tend to follow this pattern:

1. Discussion.
2. Demo.
3. Group work (during which I help students individually).
4. Discussion.
5. Stay late after class.

Things go wrong

Things will go wrong.

Suggestions: Treat failures in class as an opportunity to teach the skill of troubleshooting. Troubleshooting might take two forms.

[3] `https://www.vagrantup.com/`

Teach students how to Google skillfully. Take the following error message as an example:

```
In file(file, "rt") :
  cannot open file ’my-class-assignment.csv’: No such file or directory
```

Students will have to be taught to first *read* the error message. Second they will have to be taught to Google for the error message after removing the part of the error message which is unique to their problem: `my-class-assignment.csv`. This is an extremely important skill which your students do not possess.

Second, students have to be taught how to test their mental models of how computers work. Explain to them that they have mental models. Then teach them to make incremental changes and predict what those effect those changes will have. Work like a scientist. For example, what does this line of HTML do?

```
This text is <b>important</b>.
```

What does this change do?

```
This text is <i>important</i>.
```

Concluding thoughts

Don’t let this list of potential problems discourage you. Instead, learn from my mistakes. Then find an assignment from a teacher you admire, and steal it.