research and for immersing students in primary sources from a period dominated by slave culture.

Walter L. Buenger
Texas A&M University
College Station, Texas

doi: 10.1093/jahist/jaw170

Franklin, http://www.fdrlibrary.marist.edu/archives/collections/franklin. Created and maintained by The Roosevelt Library and the National Archives. Reviewed Oct. 2015.

Run jointly by the Franklin D. Roosevelt Presidential Library and Museum and its parent organization, the National Archives, the digital archive *Franklin* describes itself as "a virtual research room and digital repository that provides free and open access to the digitized collections of the Roosevelt Library—to everyone, anywhere in the world."

At present, *Franklin* hosts only a small fraction of the papers of the Roosevelt Library, eight hundred thousand pages out of approximately 17 million, which represents slightly less than 5 percent of the total collection. Nonetheless, by selecting documents that are in regular demand by scholars, students, and popular researchers, the archivists have provided a highly valuable resource that is easily navigable and accessible.

Franklin provides access to several of Franklin D. Roosevelt's collections during his time as president, as well as Eleanor Roosevelt's correspondence from 1945 to 1947, the diaries of Treasury Secretary Henry Morgenthau Jr., the Records of the War Refugee Board (1944–1945), and the papers of Grace Tully, FDR's personal secretary after 1941. Also provided are selections of documents based on the topics of the Holocaust and World War II refugees, the atomic bomb, and the Pearl Harbor attack (1941). Each document can be viewed online in PDF format, which allows researchers the option to save copies to their own computers.

The provided documents are highly accessible and can be located either by browsing through the entire collection or via a simple search engine. A search for the term *Évian*, for example, brings up a large number of files related to the failed (and half-hearted) attempt by the United States in 1938 to bring about a resolution to the worsening refugee crisis that was exacerbated by Germany's seizure of Austria earlier that year. A note in the search results reminds users that they are being shown documents from across various collections. In this instance, the search results include links to folders of documents from the State Department, from the Myron C. Taylor Papers, and from the Sumner Welles Papers. Moreover, the documents (at least those that were originally typed) are searchable by keyword.

The archive also contains a collection of photos in the public domain. These are organized by subject, with categories such as FDR, Eleanor Roosevelt, the Great Depression, the New Deal, and World War II, or users can wade through the seemingly infinite number of thumbnails. Since there is no guide to the photos other than the pictures, users must use the search engine to tighten the focus. Curiously, the photo collections do not correspond to the document collections. A scholar searching for information on the "atomic bomb" comes up empty-handed. A search on the term *refugee* produces only two photographs.

If *Franklin* continues to expand its collection as promised, this limited but helpful resource will become increasingly valuable to researchers. Even in its current state, it is an excellent resource for instructors teaching future historians how to conduct archival research.

Barry Trachtenberg
University at Albany
State University of New York,
Albany, New York

doi: 10.1093/jahist/jaw171

The Programming Historian, http://programminghistorian.org/. Edited by Adam Crymble, Fred Gibbs, Allison Hegel, Caleb McDaniel, Ian Milligan, Miriam Posner, and William J. Turkel. Reviewed Dec. 2015–Jan. 2016.

Learning to code is a charged topic in digital history. Some scholars, leaning on the poles of digital history's "big tent" ethos, insist that learning to program is an unnecessary

barrier to entry and rightly point out that one can do digital history without writing a single line of code. Yet the field also holds up coding as a gold standard, the pinnacle of technical achievement. One can easily find software and Web sites, as well as research in the field that used to be called humanities computing, to show how historians who program have influenced the profession. Such debates have a much broader context. Scientists, too, are sorting out how programming fits in their research and graduate programs. Learning to code is also an issue in K–12 education, with even President Barack Obama claiming that "everybody's got to learn how to code early" and backing it up by writing a few lines of JavaScript in the Oval Office. It is no surprise, then, that coding instruction has proliferated, with most of it aimed at the tech industry.

The Programming Historian Web site, now in its second edition, is aimed at teaching historians how to write code. It describes itself as offering "novice-friendly, peer-reviewed tutorials that help humanists learn a wide range of digital tools, techniques, and workflows to facilitate their research." The site offers over forty tutorials on how to accomplish specific tasks. For example, some of the tutorials explain how to work with Web Application Programming Interfaces (APIs) to access historical data, while another explains how to extract information, such as place names from that data. The site also includes a connected set of lessons, retained from the first edition of the site, explaining the basics of the Python programming language. Especially valuable is the tutorials' clear explanation of why and how a historian can use programming.

In one lesson, for example, titled "Data Mining the Internet Archive Collection," Caleb McDaniel explains how to download bibliographic data about the Boston Public Library's anti slavery collection. McDaniel explains how to identify the URLs at which the data is available, how to download it, and how to extract the pieces of interest. Along the way he teaches basic programming concepts such as for-loops (which allow code to be executed repeatedly). But McDaniel also explains that bibliographic data shows connections between objects and points users to other lessons on mapping and counting word frequencies for further analysis. Likewise, Vilja Hulden's tutorial on using naive Bayesian approaches to classifying documents (the most methodologically ambitious of the tutorials) connects machine-learning techniques to the digital archive for Old Bailey (London's central criminal court from 1674 to 1913). The tutorials are at their best when they connect technical skills to specific ways of doing research within the discipline.

To understand the aims of *The Programming Historian,* we should distinguish between "programming" and "scripting." "Programming" is writing code that solves some abstract problem. For instance, one could write a library for Python that implements a machine-learning technique useful for a broad range of problems. "Scripting" is writing code that solves some particular research question at hand. For instance, one might use that same machine-learning library to study nineteenth-century newspapers. That terminology is arbitrary, but the distinction is not. Because programming is necessarily abstract and thus less specific to the domain of history, *The Programming Historian* appropriately focuses on how to write scripts that use existing libraries and programs to solve historical problems. (Some of the lessons teach neither programming nor scripting but instead offer instruction in stand-alone tools such as QGIS, AntConc, and Omeka.)

This distinction points out one weakness in the project's pedagogical approach. What answer can be given to the oft-asked question: Why should a historian learn to code? A useful response may be that historians can use code to access APIs, download files, or visualize data, but that answer is insufficient because one can do all of those things using stand-alone tools. A more compelling answer is that by writing code a historian can escape a fundamental limitation of most digital tools: constrained historical practice. Most stand-alone tools are intended to take one kind of input and produce one kind of output. AntConc takes in text documents and outputs linguistic analysis. QGIS takes in spatial data and outputs maps or spatial analysis. QGIS cannot analyze texts, and AntConc cannot make maps, but a programming language used by historians, such as Python or R, can accept virtually unlimited inputs and produce virtually unlimited outputs.

Both languages have a wide variety of libraries available for analyzing texts, images, sounds, networks, and spatial data. A scholar working in such a language can write scripts that deal with any kind of historical problem amenable to computation. Scripting therefore better aligns with the practice of history because historians are omnivorous when it comes to sources. While other disciplines might be defined by their source base (poems, for example, or voting data), historians will use whatever they can access.

The stand-alone tutorials in *The Programming Historian* have difficulty showing users how to use a programming language as a tool capable of expressing almost any computational historical analysis instead of as a set of recipes that accomplish a set of discrete tasks. The first edition of the site, still available within the second edition, gave readers a stronger base by focusing on Python in a connected series of lessons. While *The Programming Historian* starts in the right place by giving users lessons in specific tasks, future editions could build on this base by showing how its lessons fit together.

The Programming Historian aims to influence how the digital history community invites contributors, reviews their work, and publishes it. Each tutorial has been peer reviewed. More than twenty authors and about forty reviewers have already contributed. The process of peer review is well thought out and, above all, highly public. Every prospective tutorial is visible in the lesson pipeline. The guidelines for contributing content are clear, and while the process of using GitHub might seem somewhat intimidating, the editors provide a step-by-step tutorial, and familiarity with version control software is a reasonable requirement for someone teaching programming. Every article is licensed using the Creative Commons CC-BY license, giving anyone freedom to use or adapt the tutorials as long as they cite the original version. *The Programming Historian* is a highly successful model of collaborative academic publishing on the Web.

Lincoln Mullen
George Mason University
Fairfax, Virginia

doi: 10.1093/jahist/jaw172