

The Open Digital Archaeology Textbook Environment

Shawn Graham, Neha Gupta, Jolene Smith, Andreas Angourakis, Michael Carter, & Beth Compton

2018-02-04

Contents

notice	5
About the Authors	7
Getting Started	9
How to use this text	9
How to contribute changes, or make your own version	10
How to access and use the computational environment	10
Colophon	11
Welcome!	13
1 Going Digital	15
1.1 So what is Digital Archaeology?	17
1.2 Project Management Basics	22
1.3 Github & Version Control	24
1.4 Open Notebook Research & Scholarly Communication	31
1.5 Failing Productively	39
1.6 The Ethics of Big Data in Archaeology	42
2 Making Data Durable	45
2.1 Designing Data Collection	46
2.2 Cleaning Data with Open Refine	47
2.3 Linked Open Data and Data Publishing	47
2.4 Introduction to Digital Libraries, Archives & Repositories	47
2.5 Command Line Methods for Working with APIs	47
3 Finding and Communicating the Compelling Story	49
3.1 Statistical Computing with R and Python Notebooks; Reproducible code	49
3.2 D3, Processing, and Data Driven Documents	50
3.3 Storytelling and the Archaeological CMS: Omeka, Kora	51
3.4 What is Web Mapping?	51
3.5 Place-based Interpretation with Locative Augmented Reality	52
3.6 Virtual Archaeology	52
3.7 Archaeogaming	53
3.8 Social media as Public Engagement & Scholarly Communication in Archaeology	53
4 Eliding the Digital and the Physical	55
4.1 3D Photogrammetry & Structure from Motion	55
4.2 3D Printing, the Internet of Things and “Maker” Archaeology	57
4.3 Artificial Intelligence in Digital Archaeology	57

5	Digital Archaeology's Place in the World	63
5.1	Marketing Digital Archaeology	63
5.2	Sustainability & Power in Digital Archaeology	63
6	On the Horizons: Where Digital Archaeology Might Go Next	65
7	Open Review	67
7.1	Contribute	67
	References	69

notice

This volume goes hand-in-glove with a computational environment built on the DHBox.

THIS IS A DRAFT VERSION; it's not even version 0.1 yet. It is filled with errors, omissions, and non-sequiturs.



Figure 1: A word cloud image of the original ODATE proposal, arranged to mimic a photograph of the temple of Athena Pronaos at Delphi



The online version of this book is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

About the Authors

Shawn Graham

Shawn Graham trained in Roman archaeology but has become over the years a digital archaeologist and digital humanist. He is currently an associate professor in the Department of History at Carleton University in Ottawa Canada. He keeps an open lab notebook of his research and experiments in digital history and archaeology at his research blog, [electricarchaeology](#). He can be found on Twitter at [electricarchaeo](#). His teaching explores methods and digital approaches at all levels, including seminars in the collaborative MA Digital Humanities program as well as in the MA Public History program. His research interests include augmented reality and virtual reality in the service of landscape archaeology, video games about the past, and agent based modeling.

Neha Gupta

Neha Gupta is an anthropologist and recent postdoctoral fellow at Memorial University. Her research specialties include geovisualization and GIS, landscape and settlement archaeology and the archaeology of India. She recently launched MINA | Map Indian Archaeology, a public digital project to promote the archaeology of India and to encourage collaboration in the development of digital tools and technologies appropriate for archaeology. Her projects regularly center on the themes of colonial practices, web maps, Indigenous peoples and archaeology's relationship with society.

Michael Carter

Michael Carter is an Assistant Professor in the School of Creative Industries at Ryerson University. He is also Director of Industry Relations - Master in Digital Media Program/Yeates School of Graduate Studies. His research is primarily centred around the Theory, Method and Practice of Virtual Archaeology, which flows naturally from his previous career in animation and visual special effects. More about his research can be explored on his [Research Gate](#) profile

Beth Compton

Beth Compton is a PhD student at the University of Western Ontario's Department of Anthropology. Her research examines critically the implications and assumptions of digital approaches to repatriation, and uses archaeological ethnographic approaches to evaluate artifact sharing policies. She study archaeological representation and replication, particularly the use of digital photography, 3D modeling, and 3D printing. She is interested in all aspects of meaning-making and knowledge sharing in digital archaeological heritage practice. She is a Co-Founder of the DH Maker Bus project and is an Ontario Trillium Scholar.

Jolene Smith

Brief Bio

Andreas Angourakis

Brief Bio

Editorial Board

Katharine Cook, University of Victoria

Ethan Watrall, Michigan State University

Daniel Pett, The British Museum

Eric Kansa, Open Context & The Alexandria Archive Institute

Kathleen Fitzpatrick, Michigan State University

Getting Started

We wrote this text with a particular kind of student in mind. We imagined this student as having already taken a first year introductory course to archaeology, of the kind that surveys the field, its history, and its principle methods and theoretical positions. Very few courses of that kind include any depth in digital methods and theory, which is understandable when we look at the incredible variety of archaeological work, skills, and interests! Digital work is every bit as diverse as other kinds of archaeology, but it also presents its own particular challenges. One of these is the anxiety that comes when one first approaches the computer for anything more complex than word processing or a bit of social media. ‘What happens if I break it?’, ‘I’m not techy!’, ‘If I wanted to do computers, I wouldn’t have gone into this!’ are all actual student concerns that we have heard in our various classrooms.

It’ll be ok.

We take a pedagogical perspective that focuses on the learning that happens when we make things, when we experiment or otherwise play around with materials and computing, and especially, when/if things break. It’s a perspective that finds value in ‘failing gloriously’, in trying to push ourselves beyond our comfort level. The only thing that you will need therefore to be successful in learning some of the basic issues around digital archaeology is a willingness to consider why things didn’t work the way they ought to have, and a web browser. We built this textbook with its very own digital archaeology computer built right in! There’s nothing you can break on your own machine, nor does your machine have to be very powerful. If you want, you can even download and install ODATE on a desktop computer to use as your very own digital archaeology laboratory.

How to use this text

Each section in this book is broken down into an overview or discussion of the main concepts, and then followed up with skill-building exercises. The computational environment, when you log into it, is set to expire after three hours - so you really ought to work through the sections on Github & Version Control and Open Notebook Research & Scholarly Communication so you’ll be able to get your work and data out of ODATE and onto space that you control. The best way to use this book is to make sure you have at least one hour blocked out to read through a section, and then two hours to go through the section again if you’re working on the exercises.

Do you notice that stripe down the side of the screen at the right? That’s a tool-bar for annotating the text, using a tool called **Hypothes.is**. If you highlight any text (go ahead, highlight that phrase right now by right-clicking and dragging your mouse!) a little pop-up will ask you if you want to annotate or highlight the text. If you choose annotate, a writing pane will open on the right. Using the Hypothesis tool requires a reader to create a login and account with Hypothesis, which is managed by the Hypothesis site, not us.

By default, such annotations are made public. Private annotations can only be viewed by the particular individual who made them. All annotations (both public and private) have their own unique URL and can be collated in various ways using the Hypothesis API (here’s an example). Please tag your annotation with `odate` to allow easy curating of the public annotations.

Please note that any public annotations can be read by any other reader. These can also be responded to, as well - which might make a great classroom activity! A class can create group annotations which are only visible to participants in that group (instructions here). Annotation is a tool for research; personal reaction to anything we've written in ODATE should be done via the reader's blog while leaving an annotation on ODATE linking to the blog piece. Because the API or 'application programming interface' for Hypothesis allows one to retrieve annotations programmatically, there is a growing world of scripts and plugins for managing or retrieving those annotations. Kris Shaffer has created a Wordpress plugin to pull annotations to a new Wordpress post (details are linked here), which might be another great option for a class blog working through ODATE.

Over time, the parts of the text that are heavily annotated will look as if someone has gone over them with a yellow highlighter. You can use this to help guide your reading - perhaps that's a part where many people had problems, or perhaps it's a part that sparked a really interesting discussion! Group annotation like this promotes 'active reading', which means that you're more likely to retain the discussion.

Finally, if you'd rather not read this as a web page, you can grab a pdf copy by pressing the download button above (the downwards-facing arrow icon) and printing out just the bits you want or need. If you'd rather read this text via an e-reader or iBooks or similar, the download link will also give you an ePub version. Individuals who use a screenreader or other assistive device might prefer to work with the pdf or epub versions. Please do let us know if there is any way we can make this text more accessible for users with particular needs. Since this text is fundamentally a series of plain-text files that we then manipulate to create these different outputs, it should be straightforward for us to adapt accordingly!

How to contribute changes, or make your own version

Perhaps your professor has assigned a portion of this text to your class, with the instruction to improve it. Do you see the edit button at the top of the screen (it looks like a little square with a pencil)? If you click on that, and you have an account on Github (and you're signed in), you will grab a copy of this entire text (but not the computational part; the source code for that is here instead) that you can then edit. If you want, you can also make a **pull-request** to us, asking us to fold your changes into our textbook. We welcome these suggestions! Since this book has a creative-commons license, you are welcome to expand and build upon this as you wish, but do cite and link back to the original version.

How to access and use the computational environment

The computational part of this textbook is a version of the DHBox created by a team from the CUNY Graduate Centre:

- Stephen Zweibel Lead Developer
- Patrick Smyth Developer, Writer
- Jojo Karlin Outreach Coordinator
- Matthew K. Gold Academic Advisor

The DHBox is a linux-powered computer accessible to anyone. It has a number of common digital humanities tools preinstalled for use, including command line access, a file manager, an R Studio Environment, and Brackets (for web development). It also includes Jupyter Notebooks, an environment for 'literate computing' (or when the analytical code and supporting scholarship are integrated into a document that allows the research to be reproduced). Our version of DHBox also includes a few more elements pre-installed explicitly for archaeological use, such as visual structure-from-motion photogrammetry and QGIS.

Open a new browser tab at <http://134.117.26.132:5000/> to access the ODATE box. **NB as this work is in development, that box is only currently available to users with VPN access to Carleton University servers. It will be made publically available in due course.**

Create a new user account; select the maximum amount of time available for your box. Each time you login, the box will tell you how much time remains for your account. You can use the file manager to download your work from the box to your own machine.

(Each account has a time limit so that our computing resources do not become overwhelmed with dormant accounts and data).

Colophon

This text was created using the Bookdown package for R Markdown. R Markdown is a variant of the simple Markdown format created by John Gruber. That is to say, at its core this text is a series of simple text-files marked up with simple markers of syntax like `#` marks to indicate headings and so on. R Markdown allows us to embed code snippets within the text that an interpreter, like R Studio, knows how to run, such that the results of the calculations become embedded in the surrounding discussion! This is a key part of making research more open and more reproducible, and which you'll learn more about in chapter one.

The sequence of steps to produce a Bookdown-powered site looks like this:

1. create a new project in RStudio (we typically create a new project in a brand new folder)
2. run the following script to install Bookdown:

```
install.packages("devtools")
devtools::install_github("rstudio/bookdown")
```

3. create a new textfile with `metadata` that describe how the book will be built. The metadata is in a format called YAML ('yet another markup language') that uses keys and values that get passed into other parts of Bookdown:

```
title: "The archaeology of spoons"
author: "Graham, Gupta, Carter, & Compton"
date: "July 1 2017"
description: "A book about cocleararchaeology."
github-repo: "my-github-account/my-project"
cover-image: "images/cover.png"
url: 'https://my-domain-ex/my-project/'
bibliography: myproject.bib
biblio-style: apa-like
link-citations: yes
```

This is the only thing you need to have in this file, which is saved in the project folder as `index.Rmd`.

4. Write! We write the content of this book as text files, saving the parts in order. Each file should be numbered `01-introduction.Rmd`, `02-a-history-of-spoons.Rmd`, `03-the-spoons-of-site-x.Rmd` and so on.
5. Build the book. With Bookdown installed, there will be a 'Build Book' button in the R Studio build pane. This will generate the static html files for the book, the pdf, and the epub. All of these will be found in a new folder in your project, `_book`. There are many more customizations that can be done, but that is sufficient to get one started.

The computational environment

The computational environment we are using is a fork of the DhBox project from CUNY, led by Stephen Zweibel, Patrick Smyth Developer, Jojo Karlin, and Matthew K. Gold. DHBox requires Ubuntu ≥ 14.04 and Python 2.7x. We run it at Carleton courtesy of the School of Computer Science, Andrew Pullin, Peter

Choynowski, and Doug Howe, via OpenStack and Docker. A version of DHBox that can be run from a USB stick can be found [here](#) and is worth exploring!

Welcome!

Digital archaeology as a field rests upon the creative use of primarily open-source and/or open-access materials to archive, reuse, visualize, analyze and communicate archaeological data. This reliance on open-source and open-access is a political stance that emerges in opposition to archaeology's past complicity in colonial enterprises and scholarship; digital archaeology resists the digital neo-colonialism of Google, Facebook, and similar tech giants that typically promote disciplinary silos and closed data repositories. Specifically, digital archaeology encourages innovative, reflective, and critical use of open access data and the development of digital tools that facilitate linkages and analysis across varied digital sources.

To that end, this document you are reading is integrated with a cloud-based digital exploratory laboratory of multiple cloud-computing tools with teaching materials that instructors will be able to use 'out-of-the-box' with a single click, or to remix as circumstances dictate. Part of our inspiration comes from the 'DHBox' project from CUNY (City University of New York, [link](#)), a project that is creating a 'digital humanities laboratory' in the cloud. While the tools of the digital humanities are congruent with those of digital archaeology, they are typically configured to work with texts rather than material culture in which archaeologists specialise. The second inspiration is the open-access guide 'The Programming Historian', which is a series of how-tos and tutorials ([link](#)) pitched at historians confronting digital sources for the first time. A key challenge scholars face in carrying out novel digital analysis is how to install or configure software; each 'Programming Historian' tutorial therefore explains in length and in detail how to configure software. The present e-textbook merges the best of both approaches to create a singular experience for instructors and students: a one-click digital laboratory approach, where installation of materials is not an issue, and with carefully designed tutorials and lessons on theory and practice in digital archaeology.

This is not a textbook about learning how to code. Rather, it is about instilling the habits of thought that will enable success when confronted with digital novelty, the habits of thought that will enable you to determine how to work with digital materials, and the habits of thought that permit you to see where and when digital approaches will make the difference in your research. Skills change; techniques evolve; new tools emerge. Habits of thought are hard to cultivate but have staying power!

Through this textbook, we (Graham, Gupta, Carter and Compton) aim to offer a learners'-perspective-view on digital methods in archaeology, that is, how we might think with, and through, digital sources of information, digital tools and technologies and their relationship with society. We are deeply aware of how rapidly both digital sources and technologies can change, particularly on the Web; we therefore present this e-textbook and open-learning environment as a guide to best practices when working with available digital data and digital tools, what kinds of analysis are possible, how to perform these analytical techniques, and how you might publish your data, making them re-usable for another scholar and ethics and ethical issues in doing digital archaeology.

Chapter 1

Going Digital

Digital archaeology should exist to assist us in the performance of archaeology as a whole. It should not be a secret knowledge, nor a distinct school of thought, but rather simply seen as archaeology done well, using all of the tools available to and in better recovering, understanding and presenting the past. In the end, there is no such thing as digital archaeology. What exists, or at least what should exist, are intelligent and practical ways of applying the use of computers to archaeology that better enable us to pursue both our theoretical questions and our methodological applications. (Evans, Daly, and MyiLibrary 2006)

While we agree with the first part of the sentiment, the second part is rather up for debate. We believe that there *is* such a thing as digital archaeology. Digital tools exist in a meshwork of legal and cultural obligations, and moreso than any other tool humans have yet come up with, have the capability to exert their own agency upon the user. Digital tools and their use are not theory-free nor without theoretical implications. There is no such thing as neutral, when digital tools are employed. This is why digital archaeology is - or should be - a distinct subfield of the wider archaeological project.

In a conversation initiated on Twitter on March 10, 2017, Graham asked the question (the thread for which discussion starts [here](#)), ‘is digital archaeology the same as using computers in archaeology?’ The resulting conversation ranged widely over everything from the topic of study to the ways in which computational power enables the researcher to ask questions that were not previously feasible to ask. Other researchers sounded a note of caution against the kind of ‘technological fetishism’ that digital work can often fall pray to, especially given the larger issues of gender and ‘solutionitis’ that emerge given the white, 20-35 year old demographic of many tech workers (for criticisms of technological solutionism or utopianism in archaeology, see the work of Colleen Morgan (2012) Joyce, Tringham, Morozov, Kansa). Others sounded a warning that to think of digital archaeology as something distinct from archaeology risks ‘going the way of DH’ and instead appealed for a holistic understanding.

Hanna Marie Pageau succinctly captured these issues, when over a series of tweets beginning [here](#) she wrote,

‘Digital archaeology has an obvious digital component. However, saying it’s simply using a computer is like saying being a computer scientist means you use a computer to do science. There is an implied addition [to the] topic of specific methods that brings you from an archaeologist using a computer to being an archaeologist who studies digital archaeology. I would argue that archaeogaming is the most straight forward example. Because while gaming is usually thought of as digital, it could study table top gaming and not technically be digital in nature. However if you’re studying ethics of representation in games you’re going from just using a computer as a tool to it being THE medium.’

In which case, an important aspect of digital archaeology that differentiates it from the use of computing power to answer archaeological questions is this question of purpose. In this section, we take up this question beginning with the question of *teaching* digital approaches. We progress by suggesting that digital archaeology

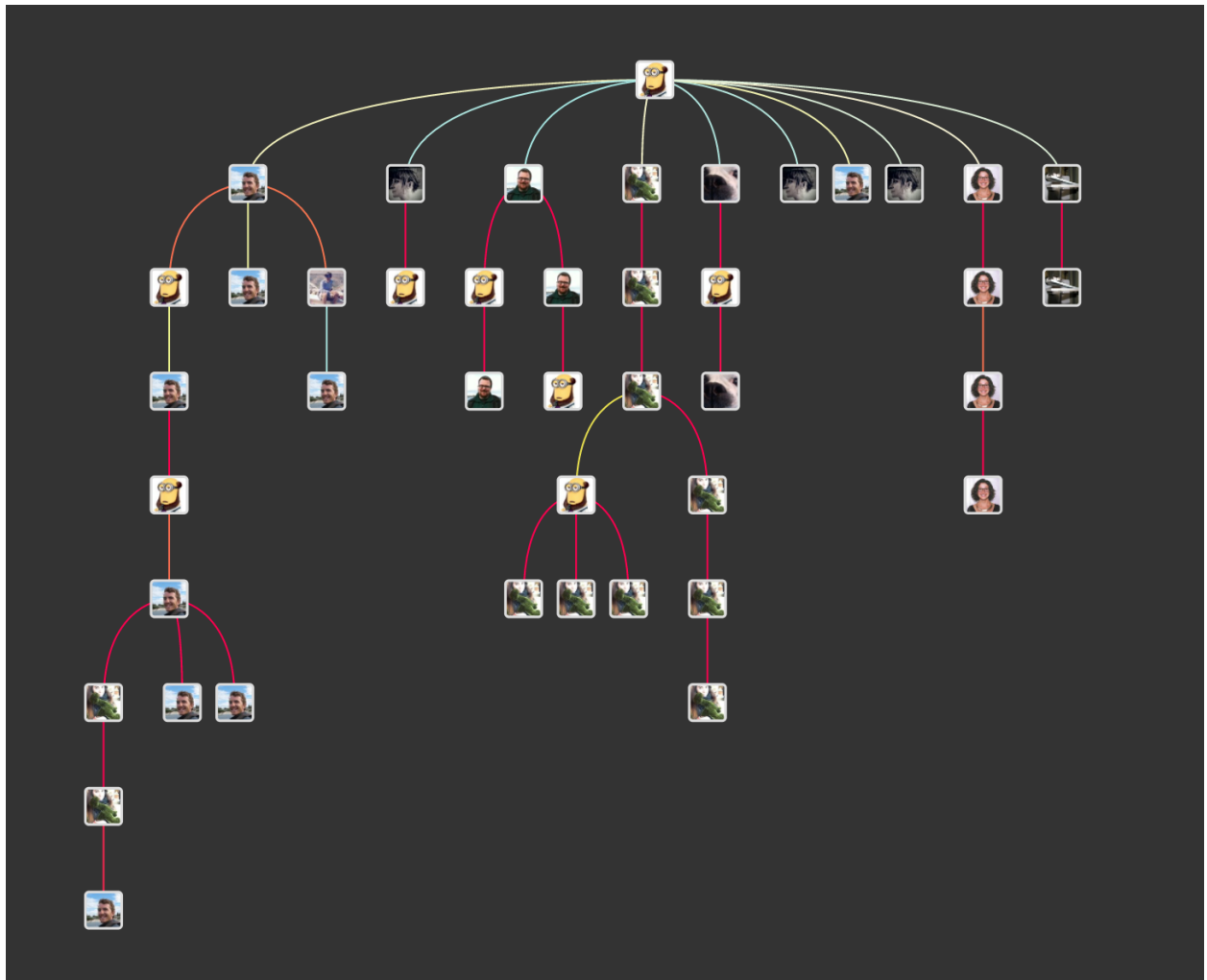


Figure 1.1: A visualization of the conversation tree on Twitter from Graham's query

is akin to work at the intersection of art and public archaeology and digital humanities. We provide you the necessary basics for setting up your own digital archaeological practice. Entrance into the world of digital archaeology requires organizational ability and facility with versioning files. It is allied with the practice of open notebook science, and it attempts to future-proof by using the simplest file formats and avoiding proprietary software where possible. These are the basics on which the rest of digital archaeological practice is founded.

1.1 So what is Digital Archaeology?

If you are holding this book in your hands, via a device or on paper, or looking at it on your desktop, you might wonder why we feel it necessary to even ask the question. It is important at the outset to make the argument that digital archaeology is not about ‘mere’ tool use. Andrew Goldstone in *Debates in the Digital Humanities* discusses this tension (Goldstone 2018). He has found (and Lincoln Mullen concurs with regard to his own teaching, (Mullen 2017)) that our current optimism about teaching technical facility is misplaced. Tools first, context second doesn’t work. Alternatively, theory first doesn’t seem to work either. And finally, for anything to work at all, datasets have to be curated and carefully pruned for their pedagogical value. We can’t simply turn students loose on a dataset (or worse, ask them to build their own) and expect ‘learning’ to happen.

Our approach in this volume is to resolve that seeming paradox by providing not just the tools, and not just the data, but also the computer itself. Archaeologically, this puts our volume in dialog with the work of scholars such as Ben Marwick, who makes available with his research the code, the dependencies, and sometimes, an entire virtual machine, to enable other scholars to replicate, reuse, or dispute his conclusions. We want you to reuse our code, to study it, and to improve upon it. We want you to annotate our pages, and point out our errors. For us, digital archaeology is not the mere use of computational tools to answer archaeological questions. Rather, it is to enable the audience for archaeological thinking to enter into conversation with us, and to *do* archaeology for themselves.

Digital archaeology is necessarily a public archaeology. This is its principal difference with what has come before, for never forget, there has been at least a half-century of innovative use of computational power for archaeological knowledge building.

Geospatial, digital and Web-based tools are now central to carrying out archaeological research and to communicating archaeological information in a globalized world. Until recently, the accumulation and effective management of digital archaeological data have been the primary focus of archaeologists (Evans and Daly 2006). Under this model, scholars emphasize the ‘integration’ into archaeology of computing technologies, and how, by utilizing current available computing memory and processor speed, one does archaeology, only better (Daly and Evans 2006: 1). This situation in turn demonstrates the ‘marriage between the two’, archaeology and computing (Daly and Evans 2006: 2).

For Evans and Daly (2006), writing in the first decade of the 21st century, digital archaeology was synonymous with the use of Information and Communication Technology or ICT, and reflected wider efforts at that moment to transform education through newly available digital tools. Some scholars and policy makers believed that digital technologies were the answer to pressing global social issues such as poverty, a point that we will discuss later.

More recently, in his inaugural editorial for the open-access journal, *Frontiers in Digital Humanities*, Costopoulos (2016) argues that ‘digital archaeology has been [here] a while’. Computing in archaeology, that is ‘doing archaeology digitally’ as (Costopolous 2016) remarks, constitutes a ‘state of normality’ in archaeological practice. This view places emphasis on the availability of digital tools and their use in institutional contexts, overlooking the highly structured nature of social groups that employ these tools, and where, how and why these technologies are created and used. While fruitful, these views tend to obscure broader developments in the social sciences and humanities, of which archaeology is a part, and underestimate the changing relationship between archaeology and society.

1.1.1 A distant view

Ethan Watrall has drawn the history of computational archaeology/digital archaeology all the way back to the pioneering work of James Deetz in the 1960s, who used computers at MIT to perform stylistic analyses of Arikara ceramics (Ethan Watrall 2017, Deetz (1965)). Most early interest in computation for archaeology was centred on the potential for computational databases, although ambition often out-stripped capability. By the 1970s, serious efforts were being put into work to build the infrastructural knowledge necessary to make and usefully query archaeological datasets. One can see this concern play out by considering a **topic model** (Shawn Graham 2014) of the early volumes of the *Computer Applications in Archaeology* (a topic model is a way of deducing latent patterns of discourse within text, based on patternings of words (See Graham, Weingart, and Milligan 2012)):

topic 1 – computer, program, may, storage, then, excavation, recording, all, into, form, using, retrieval, any, user, output, records, package, entry, one, unit

topic 6: but, they, one, time, their, all, some, only, will, there, would, what, very, our, other, any, most, them, even

topic 20: some, will, many, there, field, problems, may, but, archaeologists, excavation, their, they, recording, however, record, new, systems, most, should, need

The beginnings of the CAA are marked by hesitation and prognostication: what *are* computers for, in archaeology? There is a sense that for archaeologists, computation is something that will be useful insofar as it can be helpful for recording information in the field. By the 1980s desktop computing was becoming sufficiently widespread that the use of geographic information systems was feasible for more and more archaeologists. The other ‘killer app’ of the time was computer-aided design, which allowed metric 3d reconstructions from the plans drawn on site by excavators. Yet, computational resources were still limited enough that computing was not something that one could merely ‘play’ with. Software was costly, computation took time, and training resources were put into learning the proprietary packages that existed (rather than coding knowledge). By the 1990s, the introduction of the cd-rom and the shift in PC gaming technologies from primarily text-based to graphical based games led to teaching simulations for archaeology, most notably T. Douglas Price and Anne Birgitte Gebauer’s *Adventures in Fugawiland*. Watrall identifies the emergence of the web as being not so much a boon for computational archaeology as it was for public archaeology (although the pioneering journal *Internet Archaeology* was first published in 1996); nevertheless, the birth of the web (which it must be remembered is *distinct from* and *overlays* the internet) allowed for a step-change in the effectiveness of the dissemination of open-source software and code, including practices for remote collaboration on code that are now beginning to percolate into scholarly publication.

The 2000s have seen, insofar as digital archaeology is concerned, a replay of the earlier episodes of computational archaeology, concomitant with each subsequent web ‘revolution’ (ie, so-called web 2.0, web 3.0 etc). Works such as (Evans, Daly, and MyiLibrary 2006) and (E. C. Kansa, Kansa, and Watrall 2011) are broadly concerned more with questions of infrastructure and training, while the more recent *Mobilizing the Past* deal with problems of training, and the ethical issues that the emerging digital surveillance permitted by our networked society presents to the practice of archaeology (and public archaeology). Perhaps the most promising new digital technologies to emerge in recent years include methods for linking open archaeological data via the web (ie, freeing various ‘silos’ of disciplinary knowledge so that the semantic connections between them can be followed and queried) and various mixed-reality approaches (virtual reality, augmented reality, 3d printing, and the so-called internet of things or the practice of wiring everything that can be wired to the web). The 2000s have also seen a growing realization that our digital tools and their algorithmic biases not only permit interesting questions to be asked about the past, but also inhibit points of view or impose their own worldviews upon the past in ways that may damage communities and/or scholarship. This reflective critique of computation in the service of archaeology marks digital archaeology within the ambit of the digital humanities (despite the division between anthropological and humanistic archaeologies).

1.1.2 Is digital archaeology part of the digital humanities?

In recent years - certainly the last decade - an idea called ‘the digital humanities’ has been percolating around the academy. It is a successor idea to ‘humanities computing’, but it captures that same distinction between discussed above. Digital archaeology has developed alongside the digital humanities, sometimes intersecting with it (notably, there was a major archaeological session at the annual international Alliance of Digital Humanities Organizations (ADHO) DH conference in 2013).

The various component organizations of the ADHO have been meeting in one form or another since the 1970s; so too the Computer Applications in Archaeology Conference has been publishing its proceedings since 1973. Archaeologists have been running simulations, doing spatial analysis, clustering, imaging, geophysicing, 3d modeling, neutron activation analyzing, x-tent modeling, etc, for what seems like ages. Happily, there is no one definition of ‘dh’ that everyone agrees on (see the various definitions collected at <http://definingdh.org/>; reload the page to get a new definition). For us, a defining *characteristic* of DH work is that public use we discussed above. But, another characteristic that we find useful to consider is the *purpose* to which computation is put in DH work. This means that digital work also has to be situated in the contexts of power and access and control (which sometimes means that digital work is mis-characterised as being part of a ‘neo-liberal’ agenda to reduce knowledge work to base profit motifs, eg Brouillet; more thoughtful work about the confluence of the digital with neoliberalism may be found in Caraher xxxx and Kansa xxxx and Greenspan xxx. We discuss the ethical dimensions to digital work more fully in The Ethics of Big Data in Archaeology.)

For us, a key difference between the kind of computational archaeology of the last years of the twentieth century versus the emerging digital archaeology of the last decade lie in the idea of the purpose behind the computing power. Trevor Owens, a digital archivist, draws attention to the purpose behind one’s use of computational power – generative discovery versus justification of an hypothesis (tjowens 2012). Discovery marks out the digital humanist whilst justification signals the humanist who uses computers. Discovery and justification are critically different concepts. For Owens, if we are using computational power to deform our texts, then we are trying to see things in a new light, to create new juxtapositions, to spark new insight. Stephen Ramsay talks about this too in Reading Machines (Ramsay 2011, 33), discussing the work of Samuels and McGann, (Samuels and McGann 1999): “Reading a poem backward is like viewing the face of a watch sideways – a way of unleashing the potentialities that altered perspectives may reveal”. This kind of reading of data (especially, but not necessarily, through digital manipulation), does not happen very much at all in archaeology. If ‘deformance’ is a key sign of the digital humanities, then digital archaeologists are not digital humanists. Owen’s point isn’t to signal who’s in or who’s out, but rather to draw attention to the fact that:

When we separate out the the context of discovery and exploration from the context of justification we end up clarifying the terms of our conversation. There is a huge difference between “here is an interesting way of thinking about this” and “This evidence supports this claim.”

This is important in the wider conversation concerning how we evaluate digital scholarship. We’ve used computers in archaeology for decades to try to justify or otherwise connect our leaps of logic and faith, spanning the gap between our data and the stories we’d like to tell. We believe, on balance, that ‘digital archaeology’ sits along this spectrum between justification and discovery closer to the discovery end, that it sits within the digital humanities and should worry less about hypothesis testing, and concentrate more on discovery and generation, of ‘interesting way[s] of thinking about this’.

Digital archaeology should be a prompt to make us ‘think different’. Let’s take a small example of how that might play out. It’s also worth suggesting that ‘play’ as a strategy for doing digital work is a valid methodology (see Ramsay (2011)). (And of course, the ability to play with computing power is a function of Moore’s law governing the increase in computing power time: computing is no longer a precious resource but something that can be ‘wasted’.)

1.1.3 Archaeological Glitch Art

Bill Caraher is a leading thinker on the implications and practice of digital archaeology. In a post on archaeological glitch art (Caraher 2012) Caraher changed file extensions to fiddle about in the insides of images of archaeological maps. He then looked at them again as images:

The idea . . . is to combine computer code and human codes to transform our computer mediated image of archaeological reality in unpredictable ways. The process is remarkably similar to analyzing the site via the GIS where we take the “natural” landscape and transform it into a series of symbols, lines, and text. By manipulating the code that produces these images in both random and patterned ways, we manipulate the meaning of the image and the way in which these images communicate information to the viewer. We problematize the process and manifestation of mediating between the experienced landscape and its representation as archaeological data.

Similarly, Graham’s work in representing archaeological data in sound (a literal auditory metaphor) translates movement over space (or through time) into a soundscape of tones (Graham 2017). This frees us from the tyranny of the screen and visual modes of knowing that often occlude more than they reveal (for instance, our Western-framed understanding of the top of the page or screen as ‘north’ means we privilege visual patterns in the vertical dimension over the horizontal (Montello et al. 2003)).

These playful approaches force us to rethink some of our norms of communication, our norms of what archaeology can concern itself with. It should be apparent that digital archaeology transcends mere ‘digital skills’ or ‘tool use’; but it also suffers from being ‘cool’.

1.1.4 The ‘cool’ factor

Alan Liu (Liu 2004) wondered what the role of the arts and humanities was in an age of knowledge work, of deliverables, of an historical event horizon that only goes back the last financial quarter. He examined the idea of ‘knowledge work’ and teased out how much of the driving force behind it is in pursuit of the ‘cool’. Through a deft plumbing of the history of the early internet (and in particular, riffing on Netscape’s ‘what’s cool?’ page from 1996 and their inability to define it except to say that they’d know it when they saw it), Liu argues that cool is ‘the aporia of information. . . cool is information designed to resist information. . . information fed back into its own signal to create a standing interference pattern, a paradox pattern’ (Liu 2004, 179). The latest web design, the latest app, the latest R package for statistics, the latest acronym on Twitter where all the digital humanists play: cool, and dividing the world.

That is, Liu argued that ‘cool’ was amongst other things a politics of knowledge work, a practice and ethos. He wondered how we might ‘challenge knowledge work to open a space, as yet culturally sterile (coopted, jejune, anarchistic, terroristic), for a more humane hack of contemporary knowledge?’ (Liu 2004, 9). Liu goes on to discuss how the tensions of ‘cool’ in knowledge work (for us, read: digital archaeology) also intersects with an ethos of the unknown, that is, of knowledge workers who work nowhere else somehow manage to stand outside that system of knowledge production. (Is alt-ac ‘alt’ partially because it is the cool work?). This matters for us as archaeologists. There are many ‘cool’ things happening in digital archaeology that somehow do not penetrate into the mainstream (such as it is). The utilitarian dots-on-a-map were once cool, but are now pedestrian. The ‘cool’ things that could be, linger on the fringes. If they did not, they wouldn’t be cool, one supposes. They resist.

To get that more humane hack that Liu seeks, Liu suggests that the historical depth that the humanities provides counters the shallowness of cool:

The humanities thus have an explanation for the new arts of the information age, whose inheritance of a frantic sequence of artistic modernisms, postmodernisms, and post-postmodernists is otherwise only a displaced encounter with the raw process of historicity. Inversely, the arts offer the humanities serious ways of engaging – both practically and theoretically- with “cool”. Together, the humanities and arts might be able to offer a persuasive argument for the humane arts in the age of knowledge work. (Liu 2004, 381).

In which case, the emergence of digital archaeologists and historians in the last decade might be the loci of the humane hacks – if we move into that space where we engage the arts. Indeed, the seminal anthropologist Tim Ingold makes this very argument with reference to his own arc as a scholar, ‘From Science to Art and Back Again’:

Revisiting science and art: which is more ecological now? Why is art leading the way in promoting radical ecological awareness? The goals of today’s science are modelling, prediction and control. Is that why we turn to art to rediscover the humility that science has lost?

We need to be making art. Digital archaeology naturally pushes in that direction.

1.1.5 Takeaways

- Digital archaeology is a public archaeology
- Digital archaeology is often about deformance rather than justification
- In that deformative practice, it is in some ways extremely aligned with artistic ways of knowing
- Digital archaeology is part of the digital humanities, and in many ways, presaged current debates and trends in that field.

All of these aspects of digital archaeology exist along a continuum. In the remainder of this chapter, we give you a ‘boot-camp’ to get you to the point where you can begin to wonder about deformation and the public entanglement with your work.

1.1.6 Exercises

The first steps in going digital are quite easy. They are fundamentally a question of maintaining some basic good habits. Everything else flows from these three habits:

1. **separate _what_ your write/create from _how_ you write it.**
2. **keep what you write/create under version control.**
3. **break tasks down into their smallest manageable bits**

Have you ever fought with Word or another wordprocessor, trying to get things just right? Word processing is a mess. It conflates writing with typesetting and layout. Sometimes, you just want to get the words out. Othertimes, you want to make your writing as accessible as possible... but your intended recipient can’t open your file, because they don’t use the same wordprocessor. Or perhaps you wrote up some great notes that you’d love to have in a slideshow; but you can’t, because copying and pasting preserves a whole lot of extra gunk that messes up your materials. Similarly, while many archaeologists will use Microsoft Excel to manipulate tabular data (artifact measurements, geochemistry data, and so on), Excel is well known for both corrupting data and for being impossible to replicate (ie, the series of clicks to manipulate or perform an analysis differ depending on the individual’s particular installation of Excel).

The answer is to separate your content from your tool, and your analytical processes separate from your data. This can help keep your thinking clear, but it also has a more nuts-and-bolts practical dimension. *A computer will always be able to read a text file.* That is to say: you’ve futureproofed your material. Any researcher will have old physical discs or disc drives or obsolete computers lying around. It is not uncommon for a colleague to remark, ‘I wrote this in Wordperfect and I can’t open this any more’. Graham’s MA thesis is trapped on a 3.5" disc drive that was compressed using a now-obsolete algorithm and it cannot be recovered. If, on the other hand, he had written the text as a .txt file, and saved the data as .csv tables, those materials would *continue* to be accessible. If the way you have manipulated or cleaned the data is written out as a **script**, then a subsequent investigator (or even your future self) can re-run the exact sequence of analysis, or re-write the script into the equivalent steps in another analytical language.

A .txt file is simply a text file; a .csv is a text file that uses commas to separate the text into columns. Similarly, a .md file is a text file that uses things like # to indicate headers, and _ to show where italicized text starts and stops. A script, in a play, tells you what to say and do. A **script** for a language like R or Python

does the same thing for the computer, and has the advantage that it is human-readable and annotatable as well, because its format *is still a simple text file*. Scripts you might encounter could have the `.r` or `.py` or `.sh` file extensions. You can open these in a text editor and see what the computer is being instructed to do. Annotations or comments in the script can be set off in various ways, and help the researcher know what is happening or is intended to happen at various points. Let's begin by creating some simple text files to document our research process, in the Markdown format.

1. A nice place to practice writing in markdown that shows you immediately how your text might be rendered when turned into html, pdf, or Word doc is Dillinger.io. Go there now to try it out. Write a short piece on why you're interested in Digital Archaeology.
 - a. Include a blockquote from the introduction to this book.
 - b. Include two links to an external site.
 - c. Embed an image.
2. In ODATE at the command line, you'll now write a markdown file using the built-in text editor `nano`. Everywhere in this book where you see the `$` symbol, we mean for you to type whatever follows *after* the `$` at the command line. The `$` is the *prompt*. Enter the command `$ nano my-first-md-file.md`. This tells the machine to use the nano text editor to make a new file called `my-first-md-file.md` and to allow you to edit it right away. (If you just wanted to create an empty file, you could use `$ touch my-first-md-file.md`). The screen that opens is a simple text editor. Re-write what you wrote for 1 but add subheadings this time. To save your work, hit `ctrl+x`. Nano will ask you if you want to save changes; select `y`. It will then prompt you for a file name, but `my-first-md-file.md` will *already* be inserted there, so just hit enter.
3. Make a new markdown file called 'todo list' or similar. Use bullet points to break down what else you need to do this week. Each bullet point should have a sub-bullet with an actual **ACTION** listed, something that you can accomplish to get things done.

** As you work through this book, we encourage you to write your thoughts, observations, or results in simple text files.** This is good practice whether or not you embark on a full-blown digital project, because ultimately, if you use a computer in your research, you *have* gone digital.

1.2 Project Management Basics

A digital project, whether in archaeology or in other fields, iterates through the same basic steps. There is

1. finding data
2. fixing data
3. analyzing the data
4. communicating the story in the data

Eighty percent of your time on any digital project will be invested in cleaning up the data and documenting what you've done to it. But in truth, a digital project begins long before we ever look at a data set (or are given data to work with, if we're part of a larger project). How do we formulate a research question or our exploration more generally? How do we translate a gut feeling or intuition or curiosity into something that is *operable*? REF Moretti on operationalizing things

The four steps we identified above are cyclical; at any one time you might be at a different stage of the process. Indeed, those four steps could easily be subsumed under what Simon Appleford and Jennifer Guiliano of devdh.org identify as the 'Best Practice Principles Of Designing Your First Project.' For Appleford and Guiliano, the outline of a project involves figuring out:

1. the question, problem, or provocation
2. sources (primary, secondary)
3. analytical activity
4. audience

5. product

Note that 4, audience, comes before 5, product. You must think of your reader/user!

Let us imagine that we were inspired by Allison Mickel's piece, 'Tracing Teams, Texts, and Topics: Applying Social Network Analysis to Understand Archaeological Knowledge Production at Çatalhöyük' (Mickel 2016).

We could frame a question: 'What role do social networks play in the development of knowledge production at my site?'

We could frame a problem: 'Mickel's exploration of social networks considered x, but not y.'

We could frame a provocation: 'Social Network Analysis promises to revolutionize our knowledge of the social contexts that underpin archaeological fieldwork, putting this power in the hands of everyone from the site director on down.'

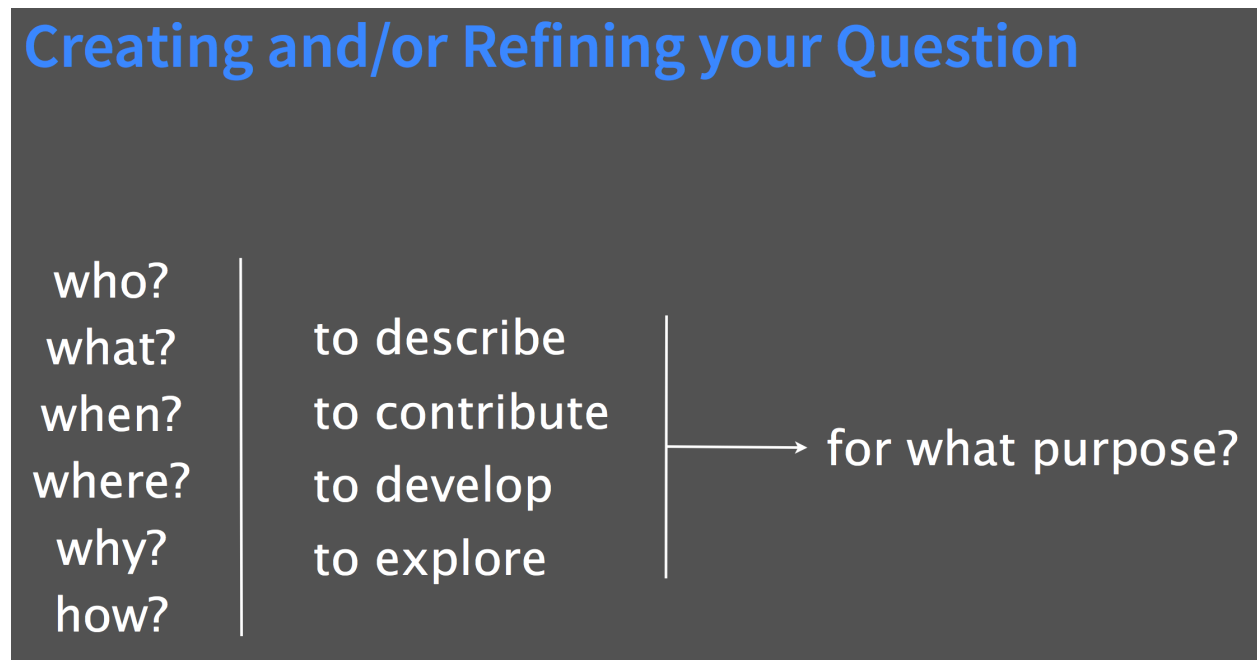


Figure 1.2: Creating and/or refining your research question, per DevDH.org

Following Appleford and Guiliano, we can refine our question, or our problem, or our provocation down to its essence in order to figure out the next parts of the the process. Knowing exactly what kind of question, problem, or provocation we're after, we then have a better sense of what to do when confronted with a mass of data (for instance, the excavation diaries from Kenen Tepe held in OpenContext.org, deposited by Parker and Cobb, 2012). Once the question is well-drawn out, questions 3 and 4 take care of themselves.

One other element that we might add is 'collaboration'. How do you plan to collaborate? While many digital archaeology projects are done by a single individual working in the quiet of their own space, most projects require many different skill sets, perspectives, and stakeholders. It is worth figuring out at the outset how you plan to work together. Will you use email? Will you use a private slack or messaging client? What about Kanban boards? (A Kanban board can be as simple as a whiteboard with three columns on it, marked 'to do', 'doing', and 'done'. Tasks are written on post-it notes and moved through the columns as necessary. A popular software implementation of a Kanban board is Trello.) We would also recommend that you write down the ideal division of labour and areas of responsibility for each of the participants, *along with a mechanism for resolving disputes*.

Finally, how much time would you have to work on your digital archaeology project? All of us have multiple demands on our time. Let's be realistic about how much time you have available. How many hours, total, do

you spend in class, at work, asleep, and socializing? Add that up for a week, then multiply by the number of weeks in your term. There are 384 hours in a 16 week term. Subtract to find out how many ‘spare’ hours you can devote to homework, this project, or a hobby.

Divide that by the number of weeks your course runs. That’s how many hours per week you can spend on all your non in-class course work. Then, divide those hours by the number of courses you have.

That’s how much time you have for your project. It’s not an awful lot, which means that the more energy you put into planning, the more effective your labour is going to be.

1.2.1 Take-aways

- be explicit about how collaboration will be managed
- be explicit about how your research goals intersect with your audience
- be brutally honest about your time and guard it jealously

1.2.2 exercises

1. (to be refined) Open a new textfile in the Archaeobox. Call it ‘initial-project-idea.md’. Using # to indicate headings, sketch out a question, problem, or provocation of your own that occurs to you as you browse the Kenen Tepe materials housed at OpenContext.org. Save that file.
2. create a project management plan. (to be fleshed out in more detail) - maybe provide a link or copy of an actual project-management document and direct the students to annotate (read others’ annotations), reflect on what the document does and where the trouble points are going to be?

1.3 Github & Version Control

It’s a familiar situation - you’ve been working on a paper. It’s where you want it to be, and you’re certain you’re done. You save it as ‘final.doc’. Then, you ask your friend to take a look at it. She spots several typos and that you flubbed an entire paragraph. You open it up, make the changes, and save as ‘final-w-changes.doc’. Later that day it occurs to you that you don’t like those changes, and you go back to the original ‘final.doc’, make some changes, and just overwrite the previous version. Soon, you have a folder like:

```
| -project
| -'finalfinal.doc'
| -'final-w-changes.doc'
| -'final-w-changes2.doc'
| -'isthisone-changes.doc'
| -'this.doc'
```

Things can get messy quite quickly. Imagine that you also have several spreadsheets in there as well, images, snippets of code... we don’t want this. What we want is a way of managing the evolution of your files. We do this with a program called Git. Git is not a user-friendly piece of software, and it takes some work to get your head around. Git is also very powerful, but fortunately, the basic uses to which most of us put it to are more or less straightforward. There are many other programs that make use of Git for version control; these programs weld a graphical user interface on top of the main Git program. It is better however to become familiar with the basic uses of git from the command line *first* before learning the idiosyncracies of these helper programs. The exercises in this section will take you through the basics of using Git from the command line.

1. Tell the story of your project
2. Travel back in time
3. Experiment with changes
4. Back up your work
5. Collaborate on projects

Figure 1.3: Alice Bartlett's summary of what Git does

1.3.1 The core functions of Git

At its heart, Git is a way of taking ‘snapshots’ of the current state of a folder, and saving those snapshots in sequence. (For an excellent brief presentation on Git, see Alice Bartlett’s presentation [here](#); Bartlett is a senior developer for the Financial Times). In Git’s lingo, a folder on your computer is known as a **repository**. This sequence of snapshots in total lets you see how your project unfolded over time. Each time you wish to take a snapshot, you make a **commit**. A commit is a Git command to take a snapshot of the entire repository. Thus, your folder we discussed above, with its proliferation of documents becomes:

```
|-project
  |-'final.doc'
```

BUT its commit history could be visualized like this:



Figure 1.4: A visualization of the history of commits

Each one of those circles represents a point in time when you the writer made a commit; Git compared the state of the file to the earlier state, and saved a snapshot of the **differences**. What is particularly useful about making a commit is that Git requires two more pieces of information about the git: who is making it, and when. The final useful bit about a commit is that you can save a detailed message about *why* the commit is being made. In our hypothetical situation, your first commit message might look like this:

Fixed conclusion

```
Julie pointed out that I had missed
the critical bit in the assignment
regarding stratigraphy. This was
added in the concluding section.
```

This information is stored in the history of the commits. In this way, you can see exactly how the project

evolved and why. Each one of these commits has what is called a **hash**. This is a unique fingerprint that you can use to ‘time travel’ (in Bartlett’s felicitous phrasing). If you want to see what your project looked like a few months ago, you **checkout** that commit. This has the effect of ‘rewinding’ the project. Once you’ve checked out a commit, don’t be alarmed when you look at the folder: your folder (your repository) looks like how it once did all those weeks ago! Any files written after that commit seem as if they’ve disappeared. Don’t worry: they still exist!

What would happen if you wanted to experiment or take your project in a new direction from that point forward? Git lets you do this. What you will do is create a new **branch** of your project from that point. You can think of a branch as like the branch of a tree, or perhaps better, a branch of a river that eventually merges back to the source. (Another way of thinking about branches is that it is a label that sticks with these particular commits.) It is generally considered ‘best practice’ to leave your **master** branch alone, in the sense that it represents the best version of your project. When you want to experiment or do something new, you create a **branch** and work there. If the work on the branch ultimately proves fruitless, you can discard it. *But*, if you decide that you like how it’s going, you can **merge** that branch back into your master. A merge is a commit that folds all of the commits from the branch with the commits from the master.

Git is also a powerful tool for backing up your work. You can work quite happily with Git on your own machine, but when you store those files and the history of commits somewhere remote, you open up the possibility of collaboration *and* a safe place where your materials can be recalled if -perish the thought- something happened to your computer. In Git-speak, the remote location is, well, the **remote**. There are many different places on the web that can function as a remote for Git repositories. You can even set one up on your own server, if you want. One of the most popular (and the one that we use for ODATE) is Github. There are many useful repositories shared via Github of interest to archaeologists - OpenContext for instance shares a lot of material that way. To get material *out* of Github and onto your own computer, you **clone** it. If that hypothetical paper you were writing was part of a group project, your partners could clone it from your Github space, and work on it as well!

You and Anna are working together on the project. You have made a new project repository in your Github space, and you have cloned it to your computer. Anna has cloned it to hers. Let’s assume that you have a very productive weekend and you make some real headway on the project. You **commit** your changes, and then **push** them from your computer to the Github version of your repository. That repository is now one commit *ahead* of Anna’s version. Anna **pulls** those changes from Github to her own version of the repository, which now looks *exactly* like your version. What happens if you make changes to the exact same part of the exact same file? This is called a **conflict**. Git will make a version of the file that contains text clearly marking off the part of the file where the conflict occurs, with the conflicting information marked out as well. The way to **resolve** the conflict is to open the file (typically with a text editor) and to delete the added Git text, making a decision on which information is the correct information.

1.3.2 Key Terms

- repository: a single folder that holds all of the files and subfolders of your project
- commit: this means, ‘take a snapshot of the current state of my repository’
- publish: take my folder on my computer, and copy it and its contents to the web as a repository at `github.com/myusername/repositoryname`
- sync: update the web repository with the latest commit from my local folder
- branch: make a copy of my repository with a ‘working name’
- merge: fold the changes I have made on a branch into another branch
- fork: to make a copy of someone else’s repo
- clone: to copy an online repo onto your own computer
- pull request: to ask the original maker of a repo to ‘pull’ your changes into their master, original, repository
- push: to move your changes from your computer to the online repo
- conflict: when two commits describe different changes to the same part of a file

1.3.3 Take-aways

- Git keeps track of all of the differences in your files, when you take a ‘snapshot’ of the state of your folder (repository) with the `commit` command
- Git allows you to roll back changes
- Git allows you to experiment by making changes that can be deleted or incorporated as desired
- Git allows you to manage collaboration safely
- Git allows you to distribute your materials

1.3.4 Further Reading

We alluded above to the presence of ‘helper’ programs that are designed to make it easier to use Git to its full potential. An excellent introduction to Github’s desktop GUI is at this Programming Historian lesson on Github. A follow-up lesson explains the way Github itself can be used to host entire websites! You may explore it here. In the section of this chapter on open notebooks, we will also use Git and Github to create a simple open notebook for your research projects.

You might also wish to dip into the archived live stream; link here from the first day of the NEH funded Institute on Digital Archaeology Method and Practice (2015) where Prof. Ethan Watrall discusses project management fundamentals and, towards the last part of the stream, introduces Git.

1.3.5 Exercises

1. How do you turn a folder into a repository? With the `git init` command. At the command line (remember, the `$` just shows you the prompt; you don’t have to type it!):
 - a. make a new director: `$ mkdir first-repo`
 - b. type `$ ls` (list) to see that the director exists. Then change directory into it: `cd first-repo`. (remember: if you’re ever not sure what directory you’re in, type `$ pwd`, or print working directory).
 - c. make a new file called `readme.md`. You do this by calling the text editor: `nano readme.md`. Type an explanation of what this exercise is about. The `.md` signals that you’re writing a text file that uses the markdown format of signalling things like headings, lists, tables, etc. (A guide to markdown syntax is here). Hit `ctrl+x` to exit, then `y` to save, leave the file name as it is.
 - d. type `$ ls` again to check that the file is there.
 - e. type `$ git init` to tell the Git program that this folder is to be tracked as a repository. If all goes correctly, you should see a variation on this message: `Initialized empty Git repository in /home/demonstration/first-repo/.git/`. But type `$ ls` again. What do you (not) see?

The changes in your repo will now be stored in that *hidden* directory, `.git`. Most of the time, you will never have reason to search that folder out. But know that the config file that describes your repo is in that folder. There might come a time in the future where you want to alter some of the default behaviour of the git program. You do that by opening the config file (which you can read with a text editor). Google ‘show hidden files and folders’ for your operating system when that time comes.

2. Open your `readme.md` file again with the nano text editor, from the command line. Add some more information to it, then save and exit the text editor.
 - a. type `$ git status`
 - b. Git will respond with a couple of pieces of information. It will tell you which **branch** you are on. It will list any untracked files present or new changes that are unstaged. We now will **stage** those changes to be added to our commit history by typing `$ git add -A`. (the bit that says `-A` adds any new, modified, or deleted files to your commit when you make it. There are other options or flags where you add *only* the new and modified files, *or* only the modified and deleted files.)
 - c. Let’s check our git status again: type `$ git status`
 - d. You should see something like this:

On branch master

Initial commit

Changes to be committed:

```
(use "git rm --cached <file>..." to unstage)
    new file:   readme.md```
```

- e. Let's take a snapshot: type `$ git commit -m "My first commit"`. What happened? Remember, Git keeps track not only of the changes, but *who* is making them. If this is your first time working with Git in the Archaeobox, Git will ask you for your name and email. Helpfully, the Git error message tells you exactly what to do: type `$ git config --global user.email "you@example.com"` and then type `$ git config --global user.name "Your Name"`. Now try making your first commit.
- f. The command above represents a bit of a shortcut for making commit messages by using the `-m` flag to associate the text in the quotation marks with the commit. Open up your `readme.md` file again, and add some more text to it. Save and exit the text editor. Add the new changes to the snapshot that we will take. Then, type `$ git commit`. Git automatically opens up the text editor so you can type a longer, more substantive commit message. In this message (unlike in markdown) the `#` indicates a line to be ignored. You'll see that there is already some default text in there telling you what to do. Type a message indicating the nature of the changes you have made. Then save and exit the text editor. DO NOT change the filename!

Congratulations, you are now able to track your changes, and keep your materials under version control!

3. Go ahead and make some more changes to your repository. Add some new files. Commit your changes after each new file is created. Now we're going to view the history of your commits. Type `$ git log`. What do you notice about this list of changes? Look at the time stamps. You'll see that the entries are listed in reverse chronological order. Each entry has its own 'hash' or unique ID, the person who made the commit and time are listed, as well as the commit message eg:

```
commit 253506bc23070753c123accbe7c495af0e8b5a43
Author: Shawn Graham <shawn.graham@carleton.ca>
Date:   Tue Feb 14 18:42:31 2017 +0000
```

Fixed the headings that were broken in the about section of `readme.md`

- a. We're going to go back in time and create a new branch. You can escape the `git log` by typing `q`. Here's how the command will look: `$ git checkout -b branchname <commit>` where `branch` is the name you want the branch to be called, and `<commit>` is that unique ID. Make a new branch from your second last commit (don't use `<` or `>`).
 - b. We typed `git checkout -b experiment 253506bc23070753c123accbe7c495af0e8b5a43`. The response: `Switched to a new branch 'experiment'` Check `git status` and then list the contents of your repository. What do you see? You should notice that some of the files you had created before seem to have disappeared - congratulations, you've time travelled! Those files are not missing; but they *are* on a different branch (the master branch) and you can't harm them now. Add a number of new files, making commits after each one. Check your `git status`, and check your `git log` as you go to make sure you're getting everything. Make sure there are no unstaged changes - everything's been committed.
4. Now let's assume that your `experiment` branch was successful - everything you did there you were happy with and you want to integrate all of those changes back into your `master` branch. We're going to merge things. To merge, we have to go back to the master branch: `$ git checkout master`. (Good practice is to keep separate branches for all major experiments or directions you go. In case you lose track of the names of the branches you've created, this command: `git branch -va` will list them for you.)
 - a. Now, we merge with `$ git merge experiment`. Remember, a merge is a special kind of commit that rolls all previous commits from both branches into one - Git will open your text editor and prompt you to add a message (it will have a default message already there if you want it). Save and exit and ta da! Your changes have been merged together.

5. One of the most powerful aspects of using Git is the possibility of using it to manage collaborations. To do this, we have to make a copy of your repository available to others as a **remote**. There are a variety of places on the web where this can be done; one of the most popular at the moment is Github. Github allows a user to have an unlimited number of **public** repositories. Public repositories can be viewed and copied by anyone. **Private** repositories require a paid account, and access is controlled. If you are working on sensitive materials that can only be shared amongst the collaborators on a project, you should invest in an upgraded account (note that you can also control which files get included in commit; see this help file. In essence, you simply list the file names you do not want committed; here's an example). Let's assume that your materials are not sensitive.
 - a. Go to Github, register for an account.
 - b. On the upper right part of the screen there is a large + sign. Click on that, and select **new public repository**
 - c. On the following screen, give your repo a name.
 - d. DO NOT 'initialize this repo with a readme.md'. Leave `add .gitignore` and `add license` set to NONE.
 - e. Click the green 'Create Repository' button.
 - f. You now have a space into which you will publish the repository on your machine. At the command line, we now need to tell Git the location of this space. We do that with the following command, where you will change `your-username` and `your-new-repo` appropriately:

```
$ git remote add origin https://github.com/YOUR-USERNAME/YOUR-NEW-REPO.git
```

- g. Now we push your local copy of the repository onto the web, to the Github version of your repo:

```
git push -u origin master
```

NB If you wanted to push a **branch** to your repository on the web instead, do you see how you would do that? If your branch was called **experiment**, the command would look like this:

```
$ git push origin experiment
```

- h. The changes can sometimes take a few minutes to show up on the website. Now, the next time you make changes to this repository, you can push them to your Github account - which is the 'origin' in the command above. Add a new text file. Commit the changes. Push the changes to your account.
6. Imagine you are collaborating with one of your classmates. Your classmate is in charge of the project, and is keeping track of the 'official' folder of materials (eg, the repo). You wish to make some changes to the files in that repository. You can manage that collaboration via Github by making a copy, what Github calls a **fork**.
 - a. Make sure you're logged into your Github account on the Github website. We're going to fork an example repository right now by going to <https://github.com/octocat/Spoon-Knife>. Click the 'fork' button at top-right. Github now makes a copy of the repository in your own Github account!
 - b. To make a copy of that repository on your own machine, you will now clone it with the `git clone` command. (Remember: a 'fork' copies someone's Github repo into a repo in your OWN Github account; a 'clone' makes a copy on your own MACHINE). Type:

```
$ cd..
```

```
$ pwd
```

We do that to make sure you're not *inside* any other repo you've made! Make sure you're not inside the repository we used in exercises 1 to 5, then proceed:

```
$ git clone https://github.com/YOUR-USERNAME/Spoon-Knife
```

```
$ ls
```

You now have a folder called ‘Spoon-Knife’ on your machine! Any changes you make inside that folder can be tracked with commits. You can also `git push -u origin master` when you’re inside it, and the changes will show up on your OWN copy (your fork) on Github.com. c. Make a fork of, and then clone, one of your classmates’ repositories. Create a new branch. Add a new file to the repository on your machine, and then push it to your fork on Github. Remember, your new file will appear on the new branch you created, NOT the master branch.

7. Now, you let your collaborator know that you’ve made a change that you want her to **merge** into the original repository. You do this by issuing a **pull request**. But first, we have to tell Git to keep an eye on that original repository, which we will call **upstream**. You do this by adding that repository’s location like so:

- a. type (but change the address appropriately):

```
$ git remote add upstream THE-FULL-URL-TO-THEIR-REPO-ENDING-WITH-.git
```

- b. You can keep your version of the remote up-to-date by fetching any new changes your classmate has done:

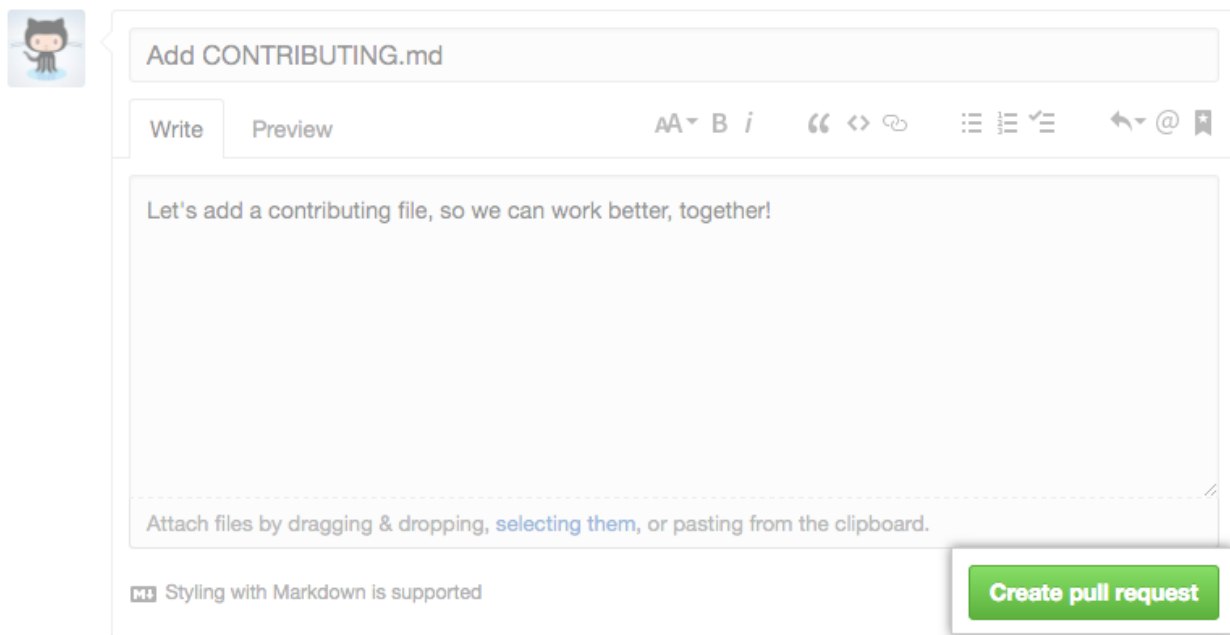
```
$ git fetch upstream
```

- c. Now let’s make a **pull request** (you might want to bookmark this help document). Go to your copy of your classmate’s repository at your Github account. Make sure you’ve selected the correct branch you pushed your changes to, by selecting it from the Branches menu drop down list.

- d. Click the ‘new pull request’ button.

- e. The new page that appears can be confusing, but it is trying to double check with you which changes you want to make, and where. ‘*Base Branch*’ is the branch where you want your changes to go, ie, your classmate’s repository. ‘*head branch*’ is the branch where you made *your* changes. Make sure these are set properly. Remember: the first one is the TO, the second one is the FROM: the place where you want your changes to go TO, FROM the place where you made the changes.

- f. A pull request has to have a message attached to it, so that your classmate knows what kind of change you’re proposing. Fill in the message fields appropriately, then hit the ‘create pull request’ button.



The screenshot shows the GitHub interface for creating a pull request. At the top left is the GitHub logo. The main title of the pull request is 'Add CONTRIBUTING.md'. Below the title are two tabs: 'Write' (active) and 'Preview'. The 'Write' tab contains a rich text editor with the text 'Let's add a contributing file, so we can work better, together!'. Above the text editor is a toolbar with various formatting options like bold, italic, link, and list. Below the text editor is a dashed line indicating where to attach files. At the bottom left, there is a note 'Styling with Markdown is supported'. At the bottom right, there is a green button labeled 'Create pull request'.

Figure 1.5: A pull request message; Image from Github.com

8. Finally, the last bit of work to be done is to accept the pull request and **merge** the changes into the original repository.
 - a. Go to your repository on your Github account. Check to see if there are any ‘pull requests’ - these will be listed under the ‘pull requests’ tab. Click on that tab.
 - b. You can merge from the command line, but for now, you can simply click on the green ‘merge pull request’ button, and then the ‘confirm merge’ button. The changes your classmate has made have now been folded into your repository.
 - c. To get the updates on your local machine, go back to the command line and type


```
$ git pull origin master
```

1.3.6 Warnings

It is possible to make changes to files directly via the edit button on Github. Be careful if you do this, because things rapidly can become out of sync, resulting in conflicts between differing versions of the same file. Get in the habit of making your changes on your own machine, and making sure things are committed and up-to-date (`git status`, `git pull origin master`, `git fetch upstream` are your friends) before beginning work. At this point, you might want to investigate some of the graphical interfaces for Git (such as Github Desktop). Knowing as you do how things work from the command line, the idiosyncracies of the graphical interfaces will make more sense. For further practice on the ins-and-outs of Git and Github Desktop, we recommend trying the Git-it app by Jessica Lord.

For help in resolving merge conflicts, see the Github help documentation. For a quick reminder of how the workflow should go, see this cheat-sheet by Chase Pettit.

1.4 Open Notebook Research & Scholarly Communication

Digital archaeology necessarily generates a lot of files. Many of those files are data; many more are manipulations of that data, or the data in various stages of cleaning and analysis. Without any sort of version control or revision history (as detailed in the previous section), these files quickly replicate to the point where a project can be in serious danger of failing. Which file contains the ‘correct’ data? The correct analysis? Even worse, imagine coming back to a project after a few months’ absence. Worse still, after a major operating system update of the kind foisted on Windows users from Windows 7 to Windows 10. The bad news continues: magnetic storage can fail; online cloud services can be hacked; a key person on the project can die.

Even if the data makes it to publication, there is the problem of the data not being available to others for re-interrogation or re-analysis. Requests for data from the authors of journal articles are routinely ignored, whether by accident or design. Researchers may sit on data for years. We have all of us had the experience of working on a collection of material, and then writing to the author of the original article, requesting an explanation for some aspect of the data schema used, only to find out that the author has either died, kept no notes, left the field entirely, or simply doesn’t remember.

There is no excuse for this any longer. **Open notebook science** is a gathering movement across a number of fields to make the entire research process transparent by sharing materials online as they are generated. These include everything from the data files themselves, to the code used to manipulate it, to notes and observations in various archives. Variations on this ‘strong’ position include data-publishing of the materials after the main paper has been published (see for instance OpenContext or the Journal of Open Archaeological Data). Researchers such as Ben Marwick and Mark E. Madsen are leading the field in archaeology, while scholars such as Caleb McDaniel are pushing the boundaries in history. The combination of simple text files (whether written text or tabular data such as .csv files) with static website generators (ie, html rather than

dynamically generated database websites like Wordpress) enables the live publishing of in-progress work. Carl Boettiger is often cited as one of the godfathers of this movement. He makes an important distinction:

This [notebook, not blog] is the active, permanent record of my scientific research, standing in place of the traditional paper bound lab notebook. The notebook is primarily a tool for me to do science, not communicate it. I write my entries with the hope that they are intelligible to my future self; and maybe my collaborators and experts in my field. Only the occasional entry will be written for a more general audience. [...] In these pages you will find not only thoughts and ideas, but references to the literature I read, the codes or manuscripts I write, derivations I scribble and graphs I create and mistakes I make. (Boettiger)

Major funding bodies are starting to require a similar transparency in the research that they support. Recently, the Social Sciences and Humanities Research Council of Canada published guidance on data management plans:

All research data collected with the use of SSHRC funds must be preserved and made available for use by others within a reasonable period of time. SSHRC considers “a reasonable period” to be within two years of the completion of the research project for which the data was collected.

Anecdotaly, we have also heard of work being denied funding because the data management plan, and/or the plan for knowledge mobilization, made only the briefest of nods towards these issues: ‘we shall have a blog and will save the data onto a usb stick’ does not cut it any more. A recent volume of case-studies in ‘reproducible research’ includes a contribution from Ben Marwick that details not only the benefits of such an approach, but also the ‘pain points’. Key amongst them was that not everyone participating in the project was on board using scripted code to perform the analysis (preferring instead to use the point-and-click of Excel), the duplication of effort that emerged as a result, and the complexities that arose from what he calls the ‘dual universes’ of Microsoft tools versus the open source tools. (MARWICK REF). On the other hand, the advantages outweighed the pain. For Marwick’s team, because their results and analysis can be re-queried and re-interrogated, they have an unusually high degree of confidence in what they’ve produced. Their data, and their results have a complete history of revisions that can be examined by reviewers. Their code can be re-used and re-purposed, thus making their subsequent research more efficient. Marwick goes on to create an entire ‘compendium’ of code, notes, data, and software dependencies that can be duplicated by other researchers. Indeed, we will be re-visiting their compendium in Section XXXXXXXXX.

Ultimately, McDaniels says it best about keeping open notebooks of research in progress when he writes,

The truth is that we often don’t realize the value of what we have until someone else sees it. By inviting others to see our work in progress, we also open new avenues of interpretation, uncover new linkages between things we would otherwise have persisted in seeing as unconnected, and create new opportunities for collaboration with fellow travelers. These things might still happen through the sharing of our notebooks after publication, but imagine how our publications might be enriched and improved if we lifted our gems to the sunlight before we decided which ones to set and which ones to discard? What new flashes in the pan might we find if we sifted through our sources in the company of others?

A parallel development is the growing practice of placing materials online as pre-prints or even as drafts, for sharing and for soliciting comments. Graham for instance uses a blog as a place to share longer-form discursive writing in progress; with his collaborators Ian Milligan and Scott Weingart, he even wrote a book ‘live’ on the web, warts and all (which you may still view at The Macroscopic). Sharing the draft in progress allowed them to identify errors and omissions as they wrote, and for their individual chapters and sections to be incorporated into class syllabi right away. In their particular case, they came to an arrangement with their publisher to permit the draft to remain online even after the formal publication of the ‘finished’ book - which was fortunate, as they ended up writing another chapter immediately after publication! In this, they were building on the work of scholars such as Kathleen Fitzpatrick, whose *Planned Obsolescence* was one of the first to use the Media Commons ‘comment press’ website to support the writing. Commentpress is a plugin for the widely used Wordpress blogging system, which allows comments to be made at the level of individual paragraphs. This textbook you are currently reading uses another solution, the hypothes.is plugin

that fosters communal reading and annotation of electronic texts. This points to another happy by-product of sharing one's work this way - the ability to generate communities of interest around one's research. The Kitz et al. volume is written with the Gitbook platform, which is a graphical interface for writing using Git at its core with markdown text files to manage the collaboration. The commit history for the book then also is a record of how the book evolved, and who did what to it when. In a way, it functions a bit like 'track changes' in Word, with the significant difference that the evolution of the book can be rewound and taken down different branches when desired.

In an ideal world, we would recommend that everyone should push for such radical transparency in their research and teaching. But what is safe for a group of (mostly) white, tenured, men is not safe for everyone online. In which case, what we recommend is for individuals to assess what is safest for them to do, while still making use of the affordances of Git, remote repositories, and simple text files. Bitbucket at the time of writing offers free private repositories (so you can push your changes to a remote repository without fear of others looking or cloning your materials); ReclaimHosting supports academic webhosting and allows one to set up the private 'dropbox' like file-sharing service Owncloud.

In this exercises below, we will explore how to make a simple open notebook via a combination of markdown files and a repository on Github. Ultimately, we endorse the model developed by Ben Marwick, of creating an entire 'research compendium' that can be installed on another researcher's machine, but a good place to start are with the historian Lincoln Mullen's simple notebook templates. This will introduce to you another tool in the digital archaeologist's toolkit, the open source R programming language and the R Studio 'IDE' ('integrated development environment').

Far more complicated notebooks are possible, inasmuch as they combine more features and ways of compiling your research. Scholars such as Mark Madsen use a combination of Github pages and the Jekyll blog generator (for more on using Jekyll to create static websites, see Amanda Visconti's Programming Historian tutorial.) A simple Github repository and Wordpress blog can be used in tandem, where the blog serves for the *narrative* part of a notebook, the part that tries to make sense of the notes contained in the repository. This aspect of open notebook science is critically important in that it serves to signal your *bona fides* as a serious scholarly person. Research made available online is *findable*; given the way web search works, if something cannot be found easily, it might as well not exist.

Ultimately, you will need to work out what combination of tools works best for you. Some of our students have had success using Scrivener as a way of keeping notes, where Scrivener writes to a repository folder or some other folder synced across the web (like Dropbox, for instance). In this workflow, you have one Scrivener file per project. Scrivener uses the visual conceit of actual 3 x 5 notecards. Within Scrivener, one would make one card per note, and keep them in a 'research' folder. Then, when it becomes time to write up the project, those notecards can be moved into the draft and rearranged as necessary so that the writing flows naturally from them.

1.4.1 How to Ask Questions

"I tried it and it didn't work", read the email. Tried what? What didn't work? What did the code actually say? What did you actually type? There is an art to asking questions when code or computing is involved. A lot of the issues involved in trying to get help ultimately stem from our natural reluctance, our natural reticence, to appear foolish or ignorant. Admitting in a public forum, or to a classmate or peer, or professor or colleague that you don't know how to x is intimidating. In which case, we invite you to reflect on this comic by Randall Munroe:

Learning 'how to ask questions' involves also learning 'how to answer questions'. This in turn is related to what Kathleen Fitzpatrick calls 'Generous Thinking':

Generous Thinking [begins] by proposing that rooting the humanities in generosity, and in particular in the practices of thinking with rather than reflexively against both the people and the materials with which we work, might invite more productive relationships and conversations not just among scholars but between scholars and the surrounding community - Kathleen Fitzpatrick



Figure 1.6: XKCD comic 'Ten Thousand'

This means that when your peer or colleague has a question or issue in their code or their process (or in their journey to become more digitally inflected in their work) you engage with them *at face value*, in the same spirit that the stickpeople in Munroe's cartoon do. Nothing will be more detrimental to the progression of digital archaeology than the appearance of gatekeepers and hidden knowledge.

When you run into a problem - when you first become aware that something is going awry - stop. Collect your breath. Do not click madly about in all directions, hoping that something might work.

1. Step away from your computer. Shut it down, put it to sleep, go outside. Sometimes, what you need more than anything else is just fresh eyeballs. Come back to the machine after a 30 minute break. You will be surprised at how often all that you really needed was a break.
2. If that doesn't solve the issue, your next step is to help other people **reproduce** what's happening on your machine. There are a variety of ways of doing this. To start, open your text editor and make a new 'date-my-problem.md' file with the following headings:

```
# what I wanted to do
- include links to any websites or articles you were reading
  # what I did
- include the history of commands you've typed by passing your command line history to a new text file
  # expected outcome
- describe what you think ought to be happening
- describe or list the software, the libraries, the packages you're working with
- share the actual code you're using
  # actual outcome
- what actually seems to be happening.
- copy all error messages
- take screenshots and put them online somewhere; include here the URL to the screenshot
- sometimes even a video can be handy; screen-cast-o-matic.com lets you take a video with narration o
```

With time, you will become better at describing the issue succinctly. You will also learn that many of your issues have been encountered before - one good strategy is to Google the actual text of your error message. You will discover Stack Overflow, who also have excellent advice on asking for help. There is even code to help you create code that shows other people what you're experiencing!

Your help query can also go into your open notebook. In this way, your open notebook becomes not just the record of your research, but also an invitation to others to join you.

1.4.2 discussion

Questions for discussion:

1. Search the archaeological literature (via jstor or Google Scholar) for examples of open notebook science 'in the wild'. Are you finding anything, and if so, where? Do there seem to be impediments *from the journals* regarding this practice?
2. What excites you about the possibilities of open notebook archaeology? What are the advantages?
3. What frightens you? What are the disadvantages?
4. Search online for the 'replicability crisis in science'. Is there any such thing in archaeology?
5. Study Marwick's paper REF and compare it to its supporting Github repository. What *new* questions could be asked of this data?
6. In what ways are terms like 'open access', 'open source', and 'open science' synonyms for a similar approach, and in what ways are they different?

1.4.3 Take-aways

Keeping an open notebook (or if necessary, a closed notebook; see below) is a habit that must be cultivated. As a target to aim for, try to have

- each experiment|project in its own folder
- each experiment|project with regular pattern of subfolders **data** and **figures** and **text** and **bib** etc
- the experiments|projects under version control.
- a plan for data publishing. One option is to submit the repository to zenodo or similar to obtain digital object identifiers (DOIs) for the repository
- a plan to write as you go, on a fail log or blog or what-have-you. Obtain a DOI for this, too.

We haven't mentioned DOIs in this section, but when your notebook and your narrative about your research has a DOI, it becomes easier for your colleagues to cite your work - even this work in progress!

1.4.4 Further Reading

Baker, James. 'Preserving Your Research Data', *The Programming Historian* <http://programminghistorian.org/lessons/preserving-your-research-data>

1.4.5 On Privilege and Open Notebooks

While we argue for open notebooks, there may be circumstances where this is not desirable or safe to do. Readers may also want to explore an Evernote alternative, Laverna which stores your notes in your web-browser's cache hence making them private, but also allows sync to services such as Dropbox (versioning and backup are still absolutely critical). If you work primarily on a Mac computer, nvAlt by Brett Terpstra is an excellent note-taking application that can sync remotely. Another possibility is Classeur a web app that integrates with various blogging platforms, allows for syncing and collaboration, the choice of what to make public and what to keep private, and includes the ability to sort notes into various notebooks. It does *not* save locally, so be warned that your information is on their servers. There is an API (application programming interface) that allows you to download your materials (for more on APIs, see Introduction to Digital Libraries, Archives & Repositories).

A final word on the privilege involved in keeping an open notebook is warranted. To make one's research available openly on the web, to discuss openly the things that worked, the things that haven't, the experiments tried and the dead ends explored, is at the current moment something that depends on the perceived race, class, and gender of the person doing it. What passes without comment when I (Shawn Graham, a white, tenured, professor) do something, could attract unwarranted, unwanted, and unfair attention if a woman of colour undergraduate tried. This is not to say this always happens; but disgracefully it happens far too often. It is important and necessary to fight back against the so-called 'internet culture' in these things, but it is not worth risking one's safety. To those who benefit from privilege, it is incumbent upon them to make things safe for others, to recognize that open science, open humanities, represents a net boon to our field. In which case, it is up to them to normalize such practices, to make it safe to try things out. We discuss more in the following section on what Failing Productively means, why it matters, and why it is integral not only to digital archaeology, but the culture of academic research, teaching, and outreach more generally.

1.4.6 exercises

In this series of exercises, we are going to take you through the process of setting up an open research notebook, where you control all of the code and all of the data. A good rule-of-thumb in terms of keeping a notebook is 'one notecard per thought', here adapted as 'one file per thought, one folder per project'.

1. Let us set up a template open-notebook based on Lincoln Mullen's Simple RmD Notebook. If you go to <https://lmullen.github.io/rmd-notebook/> you'll see a 'live' version of this template on the web. It is being served to us from a special branch in Mullen's Github account, called **gh-pages**. When you have a **gh-branches** branch in very nearly any repo associated with your own Github account, Github.com will treat that repository *not* as a repository but as an actual website. This allows us to update or experiment with changes on other branches, and show a 'polished' version to the world via the **gh-pages** branch. That branch will have a special address, in the form **your-account-name.github.io/your-repo**. Whenever you see **github.io** in a URL, you know that the source for that website will be found at **github.com/account-name/repo**. Do you see the difference? (For more on **gh-pages**, see the Github documentation).
- a. Begin sketching out on paper an idea for a digital archaeological project, perhaps the one you imagined at the end of our section on Project Management Basics. Imagine its file structure. In the top level folder are going to go all of your notecards. Sub-folders are going to hold diagrams that you create in the course of your reserch; source-data that you leave untouched; data that you've cleaned or manipulated; any helper code that you might create; and so on. You will use this structure to help organize your open notebook once we've installed it.
- b. Make a fork of our copy of Mullen's notebook (we've added `hypothes.is` to it). You can find our copy at <https://github.com/o-date/rmd-notebook>.
- c. **Clone** your copy to the ODATE environment at the command line. (Review Github & Version Control if necessary first).
- d. Type `ls` to make sure the `rmd-notebook` directory is present; then `cd rmd-notebook`.
- e. Check which branch you are on, and make sure it is the **gh-pages** branch. (Hint: check the status)
- f. Now you're ready to start adding notes! Remember, `.Rmd` files are just markdown files into which you can insert working R code. We're not ready to do that yet (but we will encounter it in due course), but for now, you can think of these files as simple notecards where one card = one idea. Note the existing `.rmd` files in this folder. Their filenames all begin with a number. Yours should be numbered as well. You can create a new card by typing `nano filename.rmd` where **filename** is whatever you want it to be. Your notecard can include images by using the markdown syntax, `![image title](path/to/image/filename.jpg)`; those images can be on the web or in an image folder. (A good place to practice markdown is at Dillinger.io).

Your note *must* contain some descriptive metadata at the top. This is good practice no matter what kind of note-taking system you use. In our case here, we use the **yaml** approach. This is what a minimum example looks like:

```
---
title: "First page of the notebook"
author: "Lincoln Mullen"
date: "December 3, 2015"
---
```

Title, author, and date. These will get passed to the `<meta>` tags in the eventual HTML we are going to generate. This information makes your site easier to find and to archive and to associate with you as a scholar.

- g. Now we make the public-facing website for your notebook. Mullen has bundled a series of commands into a **make** file, which acts as a short-cut for us and also ensures that the same sequence of operations is carried out everytime. (You can read more about makefiles [here](#).) At the command prompt type `$ make`.

SG `<- note pandoc and rmarkdown have to be bundled into the box before hand. Otherwise, the following c`
`$ wget https://github.com/jgm/pandoc/releases/download/1.19.2.1/pandoc-1.19.2.1-1-amd64.deb`
to get pandoc and then `$ sudo dpkg -i pandoc-1.19.2.1-1-amd64.deb`` to unzip and install it. Rmarkdown ha

- h. The make file is pushing all of your `.Rmd` files through a program called `pandoc`, and adding various styling options to make an entire website.

As an aside, Pandoc is an extremely useful piece of software for converting files from one format to another. A series of examples can be found at <http://www.pandoc.org/demos.html>. At the command line, type `$ pandoc` and then your source file, then indicate the kind of output you want using the `-o` flag, eg: `$ pandoc MANUAL.txt -o example1.html`. One of the outputs Pandoc can generate is a Word document - which means, your source text is kept in a very simple, future-proof format, and Word can be used just for typography.

Commit your changes and push them to your remote repository. Visit the `live` version of your repository - ie, the one at `github.io` *not* `github.com` to see your live open research notebook!

- i. Write a new note for your notebook recording your work, the problems you encountered, and the solutions you found. Save, `make`, commit, and push your note to the web.
- 2. Another approach involves writing markdown files, putting them online, and then using a kind of ‘helper’ file to manage their display as html. In this particular case, we are going to use something called `mdwiki`. Mdwiki involves a single html file which, when put in the same folder as a series of markdown files, acts as a kind of wrapper to turn the markdown files into pages of a wiki-style website. There is a lot of customization possible, but for now we’re going to make a basic notebook out of a mdwiki template.
 - a. Fork the minimal mdwiki template to your Github account; md wiki template is linked here
 - b. At this point, any markdown file you create and save into the `mdwiki-seed\11_CC\` folder will become a webpage, although *the .md extension should still be used in the URL*. If you study the folder structure, you’ll see that there are pre-made folders for pages, for pdfs, for images, and so on (if you clone the repo, you can then add or remove these folders as you like using the file manager). Remembering to frame any internal links as relative links. That is to say, if you saved a markdown file in `11_CC/pages/observation1.md` but wanted to link to `11_CC/pages/observation2.md`, it is enough to just add `[Click here](observation2.md)`. Because the `mdwiki-seed` you forked was *already* on a `gh-pages` branch, your notebook will be visible at `YOURUSERNAME.github.io/mdwiki-seed/`. But note: the page will reload and you’ll see `#!` or ‘hashbang’ inserted at the end of the URL. This is expected behaviour
 - c. Let’s customize this a bit. Via Github, click on the `11_CC` directory. One of the files that will be listed is `config.json`. If you click on that file, you’ll see:

```
{
  "additionalFooterText": "All content and images &copy; by Your Name Goes Here&nbsp;",
  "anchorCharacter": "#",
  "lineBreaks": "gfm",
  "title": "Your wiki name",
  "useSideMenu": true
}
```

Change the title so that it says something like `Your-name Open Research Notebook`. You can do this by clicking on the pencil icon at the top right of the file viewer (if you don’t see a pencil icon, you might not be logged into github). Scroll to the bottom and click on the ‘commit changes’ button when you’re done.

- d. Let’s add notes to this notebook. You can do this in two ways. In the first, you clone your mdwiki-seed via the command line, and use the text editor to create new pages in the appropriate folder (in this case, `11_CC\pages`), then `git commit`, `git add .`, and `git push` to get your changes live online. You can create a kind of table of contents by directing the `ls` command into a new file, like so:

```
$ ls > index.md
```

and then editing that file to turn the filenames into markdown links like so: `[display text for link](filename.md)`.

Alternatively, a more elegant approach to use in conjunction with `mdwiki` is to use Prose.io and keep your notebook live on the web. Prose.io is an editor for files hosted in Github. You log into Prose.io with your github credentials, and select the repository you wish to edit, in this case, `mdwiki-seed`. Then, click on the ‘new file’ button. This will give you a markdown text editor, and allow you to commit changes to your notebook! **Warning** do not make changes to `index.html` when using `mdwiki`. If you want a particular markdown file to appear as the default page in a folder, call it `index.md` instead. You could then periodically update your cloned copy on your own machine for back up purposes.

Either way, add some notes to the notebook, and make them available online.

1.5 Failing Productively

We have found that students are very nervous about doing digital work because, ‘what if it breaks?’ and ‘what if I can’t get it to work?’ This is perhaps a result of high-stakes testing and the ways we as educators have inculcated all-or-nothing grading in our courses. There is no room for experimentation, no room for trying things out when the final essay is worth 50% of the course grade, for instance. Playing it safe is a valid response in such an environment. A better approach, from a pedagogical point of view, is to encourage students to explore and try things out, with the grading being focused on documenting the *process* rather than on the final outcome. We will point the reader to Daniel Paul O’Donnell’s concept of the unessay; more details behind the link.

Our emphasis on open notebooks has an ulterior motive, and that is to surface the many ways in which digital work sometimes fails. We want to introduce to you the idea of ‘failing productively’ because there is such a thing as an unproductive failure. There are ways of failing that do not do us much good, and we need - especially with digital work - to consider what ‘fail’ actually can mean. In the technology world, there are various slogans surrounding the idea of ‘fail’ - fail fast; move fast and break things; fail better; for instance.

When we talk about ‘failing productively’ or failing better, it is easy for critics of digital archaeology (or the digital humanities; see Allington et al 2016 but contra: Greenspan 2016) to connect digital work to the worst excesses of the tech sector. But again, this is to misunderstand what ‘fail’ should mean. The understanding of many tech startup folks that valorizing failure as a license to burn through funding has caused a lot of harm. The tech sector failed to understand the humanistic implication of the phrase, and instead took it literally to mean ‘a lack of success is itself the goal’. But where does it come from?

The earliest use of the ‘fail better’ idea that we have found outside the world of literature and criticism seems to occur during the first tech boom, where it turns up in everything from diet books to learning to play jazz, to technology is in *The Quest for a Unified Theory of Information*. CITATION (Wolfgang Hofkirchner) That book according to Google Scholar at the time of writing has been cited 13 times, but the things that cite it have themselves been cited over 600 times. We are here merely speculating on where this mantra of the fast fail, fail better, comes from and how it spreads, but it would be a very interesting topic to explore.

Perhaps a better understanding of what ‘fail’ should mean is as something akin to what Nassim Taleb called ‘antifragility’. The fragile thing breaks under stress and randomness; the resilient thing stays the same; and the anti-fragile thing actually gets stronger as it is exposed to randomness. Kids’ bones for instance need to be exposed to shocks in order to get stronger. Academia’s systems are ‘fragile’ in that they do not tolerate fail; they are to a degree resilient, but they are not ‘antifragile’ in Taleb’s sense. The idea that ‘fail’ can break that which is ‘fragile’ is part of the issue here. So silicon valley really means ‘fail’ in the sense of ‘antifragile’ but they frequently forget that; academia sees ‘fail’ as the breaking of something fragile; and so the two are at loggerheads. Indeed, the rhetorical moves of academe often frame weak results - fails - as actual successes, thus making the scholarship fragile (hence the fierceness of academic disputes when results are challenged, sometimes). To make scholarship anti-fragile - to extract the full value of a fail and make it be productive, we need remember only one thing:

A failure shared is not a failure.

Not every experiment results in success; indeed, the failures are richer experiences because as academics we are loathe to say when something did not work – but how else will anybody know that a particular method, or approach, is flawed? If we try something, it does not work, and we then critically analyze why that should be, we have in fact entered a circle of positive feedback. This perspective is informed by our research into game based learning. A good game keeps the challenges just ahead of the player’s (student’s) ability, to create a state of ‘flow’. Critical failure is part of this: too hard, and the player quits; too easy, and the player drops the controller in disgust. The ‘fails’ that happen in a state of flow enable the player to learn how to overcome them. Perhaps if we can design assessment to tap into this state of flow, then we can create the conditions for continual learning and growth (see for instance Kee, Graham, et al. 2009). As in our teaching, so too in our research. Presner writes,

Digital projects in the Humanities, Social Sciences, and Arts share with experimental practices in the Sciences a willingness to be open about iteration and negative results. As such, experimentation and trial-and-error are inherent parts of digital research and must be recognized to carry risk. The processes of experimentation can be documented and prove to be essential in the long-term development process of an idea or project. White papers, sets of best practices, new design environments, and publications can result from such projects and these should be considered in the review process. Experimentation and risk-taking in scholarship represent the best of what the university, in all its many disciplines, has to offer society. To treat scholarship that takes on risk and the challenge of experimentation as an activity of secondary (or no) value for promotion and advancement, can only serve to reduce innovation, reward mediocrity, and retard the development of research. PRESSNER 2012 cite

1.5.1 A taxonomy of fails

There are fails, and then there are fails. Croxall and Warnick identify a taxonomy of four kinds of failure in digital work:

1. Technological Failure
2. Human Failure
3. Failure as Artifact
4. Failure as Epistemology

... to which we might add a fifth kind of fail:

5. Failing to Share

The first is the simplest: something simply did not work. The code is buggy, dust and grit got into the fan and the hardware seized. The second, while labeled ‘human failure’ really means that the *context*, the framework for encountering the technology was not erected properly, leading to a failure to appreciate what the technology could do or how it was intended to be used. This kind of failure can also emerge when we ourselves are not open to the possibilities or work that the technology entails. The next two kinds of failure emerge from the first in that they are ways of dealing with the first two kinds of failure. ‘Failure as Artifact’ means that we seek out examples of failures as things to study, working out the implications of why something did not work. Finally, ‘Failure as Epistemology’ purposely builds the opportunity to fail into the research design, such that each succeeding fail (of type 1 or type 2) moves us closer to the solution that we need. The first two refer to what happened; the second two refer to our response and how we react to the first two (*if* we react at all). The key to productive failure as we envision it is to recognize when one’s work is suffering from a type 1 or type 2 fail, and to transform it to a type 3 or type 4. Perhaps there should be a fifth category though, a failure to share. For digital archaeology to move forward, we need to know where the fails are and how to move beyond them, such that we move forward as a whole. Report not just the things that work, but also the fails. That is why we keep open research notebooks.

Lets consider some digital projects that we have been involved in, and categorize the kinds of fails they suffered from. We turn first to the HeritageCrowd project that Graham established in 2011. This project straddled community history and cultural history in a region poorly served by the internet. It was meant

to crowd-source intangible heritage via a combination of web-platform and telephony (people could phone in with stories, which were automatically transcribed and added to a webmap). The first write-up of the project was published just as the project started to get underway (CITATION GRAHAM ETAL). It's what happened next that is of interest here.

The project website was hacked, knocked offline and utterly compromised. The project failed.

Why did it fail? It was a combination of at least four distinct problems (GRAHAM CITATION):

1. poor record keeping of the *installation* process of the various technologies that made it work
2. computers talk to other computers to persuade them to do things. In this case, one computer injected malicious code into the technologies Graham was using to map the stories
3. Graham ignored security warnings from the main platform's maintainers
4. Backups and versioning: there were none.

Graham's fails here are of both type 1 and type 2. In terms of type 2, his failure to keep careful notes on how the various pieces of the project were made to fit together meant that he lacked the framework to understand how he had made the project vulnerable to attack. The actual fail point of the attack - that's a type 1 fail, but could have been avoided if Graham had participated more in the spirit of open software development, eg, read the security warnings in the developer forum! When Graham realized what had happened to his project, he was faced with two options. One option, having already published a piece on the project that hailed its successes and broad lessons for crowdsourcing cultural heritage, would have been to quietly walked away from the project (perhaps putting up a new website avverring that version 2.0 was coming, pending funding). The other option was to warn folks to beef up the security and backups for their own projects. At the time, crowdsourcing was very much an academic fashion and Graham opted for the second option in that spirit. In doing this, the HeritageCrowd project became a fail of type 3, an artifact for study and reflection. The act of blogging his post-mortem makes this project also an instance of type 5, or the communication of the fail. It is worth pointing out here that the *public* sharing of this failure is not without issues. As we indicated in the Open Notebook Research & Scholarly Communication section, the venue for sharing what hasn't worked and the lessons learned is highly contingent on many factors. Graham, as a white male tenure-track academic on the web in 2012, could share openly that things had not worked. As the web has developed in the intervening years, and with the increasing polarization of web discourse into broad ideological camps, it may well not be safe for someone in a more precarious position to share so openly. One must keep in mind one's own situation and adapt what we argue for here accordingly. Sharing fails can be done with close colleagues, students, specialist forums and so on.

If we are successful with ODATE, and the ideas of productive fail begin to permeate more widely in the teaching of digital archaeology, then a pedagogy that values fail will with time normalize such 'negative results'. We are motivated by this belief that digital archaeology is defined by the productive, pedagogical fail. It is this aspect of digital archaeology that also makes it a kind of public archaeology, and that failing in public can be the most powerful thing that digital archaeology offers the wider field.

We implore you to do your research so that others can retrace your steps; even a partial step forward is a step forward! When you find a colleague struggling, give positive and meaningful feedback. Be open about your own struggles, but get validation of your skills if necessary. Build things that make you feel good about your work into your work.

1.5.2 Exercises

What are the nature of your own fails? Reflect on a 'fail' that happened this past year. Where might it fit on the taxonomy? Share this fail via your open notebook, blog, or similar, with your classmates. How can your classmate convert their fails into types three or four?

1.6 The Ethics of Big Data in Archaeology

In its 2017 global survey of digital usage, We Are Social, a British marketing firm reported that mobile subscriptions in the Americas, Europe and the Middle East now outnumber their respective resident populations. Overall, the firm concluded that across 239 countries that were surveyed, Web usage and the number of social media users continues to grow, with many residents accessing Web content on smart phones and tablets rather than on personal computers. These are compelling statistics, and make clear that no region in the world is fully digital, and that across the ‘digital world’, there exists considerable unevenness. Yet, we are unable to discern real barriers to these tools and technologies, and the survey does not shed light on their place in social life.

For example, awareness that rural communities in the Global North (ref), as in the Global South (ref), often do not have equitable access to educational and health resources, facilities and infrastructure spurred initiatives to create low-cost, internet ready devices that can potentially address these shortcomings. One Laptop per Child launched in 2006, sought to ‘transform education’ by providing the world’s poorest children with a low-energy, rugged laptop for under \$100 (USD). Its current deployment ranges from schools in the United States (Miami, Florida and Charlotte, North Carolina), Indigenous youth in Canada, and students in the war-torn regions such as Nagorno Karabakh and Armenia, Rwanda, Gaza and Ramallah, and Afghanistan, as well as in Nicaragua, Peru, Paraguay, Uruguay, Kenya, Madagascar, India and Nepal.

In the same vein, in 2011, Aakash, a low-cost, Android-based tablet developed in India, by Indian engineers, and sponsored by the Indian government, was released into public space to bring the digital classroom into the hands of the most marginalized and remote communities in India (Phalkey and Chattapahyay 2016; Chattapadhyay and Phalkey 2016).

These well-intentioned, ambitious ICT projects have in common the belief that digital technologies solve social problems where ever they exist. The projects also share another element: digital tools are thought to be intuitive and empowering thus, they do not require ‘teachers’ or ‘teaching’. As Audrey Watters (2012) suggests, ‘parachute’ technologies i.e. devices that are dropped into school environments assume that children will ‘use [them], hack [them], and prosper’. Yet, these deployment efforts typically do not evaluate student academic achievement through time, nor do they seek to build upon, and expand existing educational infrastructure and resources (Warschauer and Ames 2010: 33-34).

We live, work and play in a globalized world that has serious inequalities in terms of wealth, basic necessities for human life such as clean water, food, housing, and safety, as well as access to education and health services and the opportunity to make a living and found a family (UN Declaration 1949). These are not new concerns; yet, they inform the international and national (often post-colonial) contexts within which archaeology is practiced. That these societal issues transcend the range and scope of any one discipline means that we cannot underestimate the influence of social and political factors on the practice of archaeology.

We believe that digital archaeology can offer insights into the social context of archaeology, and a deeper understanding of its the impact on the practice of archaeology. This in turn can guide us to how we can begin to address social inequalities in the discipline. We offer the following provocations (building on Graham, forthcoming):

- 1) the ethics of digital public archaeology are the ethics of archaeology
- 2) the ethics of making digital ‘things’ are the ethics of labour, power and access/control
- 3) a digital ‘thing’ is built and reflects the culture of its maker(s) and thereby invites critical study by anthropologists, archaeologists, historians, etc
- 4) digital things are entangled in the practices and ethics of contemporary society

.. concerning digital things & what they do, see Morgan 2011 https://www.academia.edu/2997156/Emancipatory_Digital_Archaeology esp p160

1.6.1 discussion

- include discussion of privacy
- ethics of crowdfunding
- ethics of crowdsourcing (& dubious ethics of using amazon's mechanical turk)
- ethics of public archaeology & social media

1.6.2 exercises

- maybe some sort of personal audit of what traces the student is leaving online
- some sort of case study?

Chapter 2

Making Data Durable

Elsewhere in this text we've covered approaches to creating digital archaeology data. Section 1.1 discusses three great habits to maintain while working digitally ref back to “The first steps in going digital are quite easy. They are fundamentally a question of maintaining some basic good habits. Everything else flows from these three habits:” section 1.1.6. These principles will help you and future researchers use your data, reproduce your conclusions and “futureproof” your digital work.

The concept of Paradata as implemented in events like the Heritage Jam link/reference can be applied broadly to analytical and technical digital projects to great effect. Originally documented in the London Charter for Computer-based Visualization of Cultural Heritage [source and link], paradata (literally data alongside the data) are narrative descriptions of process. No matter how well-structured your files and how reproducible.

Most archaeological researchers have encountered “bad” data in the past when trying to understand or reconstruct someone’s conclusions from field notes and excavation reports. What makes a dataset bad?

Please remind me never again to be a smart arse and give trench areas the names of people rather than code numbers. It's confusing and specialists must find it embarrassing. Why we called adjacent areas **Mervyn** and **Marvyn** (and not just A and B) I'll never know! @ [REDACTED]

FN10.06	Mervy	6030	
FN10.06	Mervy	6030	
FN10.06	Mervy	6030	
FN10.06	Mervyn	6030	
FN10.06	Mervy	6033	
FN10.06	Mervyn	6102	
FN10.06	mervyn	6123	
FN10.06		6110	

Figure 2.1: A Useful Warning from Kenny Brophy

It's not understandable. The dataset may be illegible (poor handwriting, or a bad scan of a hard copy

document). It might be made up of codes with no way to decipher their meaning. It might be a digital binary file that can no longer be read by available software (or on a physical format that's now obsolete). do

INCLUDE SOME IMAGES

It can't be easily accessed. The dataset might be saved in a format that only expensive, proprietary software can read. It might be comprised of a lot of fascinating information but challenging to extract from a format like PDF.

It's difficult to reuse. The dataset might use inconsistent terminology.

With these frustrations in mind, you have the tools to create good data. Good data are stable, human readable, and accessible. They can be rearranged and remixed. The same qualities that make data useful can also help to protect against the ravages of time.

You'll notice that quality data (however you define the term) take planning. The structure of your data and the practicalities of how you'll handle and store it should be woven into your research design. A reality to confront, however, is that there will be tradeoffs.

Up front, right at the beginning, when you're working out your research questions and goals, ask yourself some questions. What do I want my data to do? How much detail do I need to collect? Do I have a plan for curating datasets I've created? Is there space somewhere to do so? Will I need to be selective?

lots of references to Marwick paper(s) go here...

Perfection is unattainable. No dataset will ever be truly "complete." You could collect, measure, and produce data at finer and finer resolutions, ad infinitum. But you don't want to do that. And even if you think you're being comprehensive, all data you collect is only a specific, bounded representation of the real world surely there's a reference or a better way to put this.

2.1 Designing Data Collection

In teaching, there is a concept called 'backwards design', where you design your lesson from the end point you wish your students to achieve. The same is true of data. Knowing that archaeology is destructive, and that the processes of archaeology *creates* new kinds of artefacts (drawings, photos, record sheets, labels, and so on), we need to start at the end. Ideally, it is an end point that will allow someone else to reconsider the archaeological record that you have created. As Gavin Lucas (Lucas 2012), the idea of an 'archaeological record' is a three-fold, trinity-like concept-

- the material culture broadly understood
- the 'formation theory' of how that material culture came to be
- and the materialization of the construction of that material culture in the present: our archive

And so, we will begin this section on data collection by thinking through where the *things* are going to go.

Where to put your data

LOCKSS Choosing a repository Does it have room for your stuff? Is your stuff within the purview of its digital archivists? How long is this place going to be around? Does it have plans for migrating and maintaining your files? Choosing what you want to curate Simple formats = easier migration Convert databases to xml or csv tables with relationships Lossless files. TIFF over JPEG CAD- no archival alternative Geospatial- GeoJSON is text-based and human readable 3D- a new frontier in digital preservation (we don't know how to do it)

2.1.1 discussion

2.1.2 exercises

2.2 Cleaning Data with Open Refine

blahde blah blah

2.2.1 discussion

2.2.2 exercises

2.3 Linked Open Data and Data Publishing

yargble blarble floss

2.3.1 discussion

2.3.2 exercises

2.4 Introduction to Digital Libraries, Archives & Repositories

yadda linked open data in here?

- there should be a discussion here on the shape of data; not everything comes in flat files (explain what that means); also json, geojson,

2.5 Command Line Methods for Working with APIs

2.5.1 Working with Omeka

yadda

2.5.2 Working with tDAR

yadda

2.5.3 Working with ADS

2.5.4 Exercises

yadda

Chapter 3

Finding and Communicating the Compelling Story

blah blah blah

3.1 Statistical Computing with R and Python Notebooks; Reproducible code

When one of us (Graham) was a graduate student, he was tasked with teaching undergraduates how to do a chi-squared test of archaeological data. This was in the late 1990s. He duly opened up Excel, and began to craft a template there. While it was possible to use Excel's automatic chi-square statistical test on a table of data, simply showing the students how to use the function seemed undesirable. If he used the function, would the students really *know* what the test was doing? In any event, after several hours of working with the function, he determined that he couldn't actually figure out what exactly the function was doing. The dataset was from Mike Fletcher and Gary R. Lock's very clear *Digging Numbers: Elementary Statistics for Archaeologists* (spearheads and the presence/absence of a loop feature; Fletcher and Lock, 1991: 115-125). The answer Excel was providing was not the answer that Fletcher and Lock demonstrated. Was there some setting somewhere in Excel that was affecting the function? Time was ticking. Graham elected instead to build the spreadsheet so that each step - each calculation - in the statistic was its own cell. He had the students perform these steps on their own spreadsheet. When even that went awry, he saved his sheet onto a floppy disc, and loaded it on one computer at a time for the students. This time, working from the original spreadsheet, the results agreed with the text. The association between material and period was stronger than the association of material and presence of a loop.

Excel is a black box. When we use it, we have to take on faith that its statistics do what they say they are doing. Most of the time, this is not an issue - but Excel is a very complicated piece of software. Biologists, for instance, have known for years that Excel's automatic date-time conversions (that is, Excel detects the *kind* of data in a cell and changes its formatting or *the actual data itself*) could affect gene names (Zeeberg et al 2004) link. Marwick discusses other issues that occur when using Excel to manipulate our archaeological data. Archaeological data can seldom be used in the format in which they are first recorded. Much effort has to be expended to clean the data up into the neat and tidy tables that a spreadsheet requires. What decisions were made in the tidying process? What bits and pieces were left out? What bits and pieces were transformed as they were entered into the spreadsheet? If someone else were to try to confirm the results of the study - to reproduce it - and there is no record of the manipulations of the data (the specific transformations), reproducibility is unlikely. When researchers discuss a new method, without the data being available and the kinds of transformations and analyses not clearly and fully specified, the researchers have created a barrier to moving archaeology forward. As an exercise right now to understand just how difficult a problem this is,

make a table in Excel. Visualize the following data: 12 bronze spear heads, 18 iron spear heads; 6 of the bronze spear heads have a loop, 10 of the iron spear heads have a loop. Write down all of the steps you took to visualize this information. Save your work; close the spreadsheet. Now hand your list over to a peer. Have them follow the steps exactly, but do not clarify or otherwise explain the list.

Compare what they’ve created, with what you created. How close of a match is there? Did they reproduce your visualization? What elements or steps did you forget to include in your list?

Now you might begin to see some of the issues involved in relying on a point-and-click graphical user interface and a black-box like Excel with your archaeological data.

Marwick (2017) writes with regard to this problem,

This is a substantial barrier to progress in archaeology, both in establishing the veracity of previous claims and promoting the growth of new interpretations. Furthermore, if we are to contribute to contemporary conversations outside of archaeology (as we are supposedly well-positioned to do, cf. Kintigh et al. (2014)), we need to become more efficient, interoperative, and flexible in our research. We have to be able to invite researchers from other fields into our research pipelines to collaborate in answering interesting and broad questions about past societies.

Marwick lists four principles for reproducible archaeology. Note that ‘reproducible’ here means being able to re-do the analysis and obtain the same results; ‘replicable’ means using the same methods on a new data set collected and analyzed with the published methods of an earlier study. The principles that we should follow are:

- make the data and the methods that generated the results openly available
- script the statistical analysis (that is, no more point-and-click statistics). More on this below.
- use version control
- record the computational environment employed
- how R addresses these issues
- how R works with version control
- point to Marwick’s more complex book on R for archaeologists
- R markdown and jupyter notebooks : literate programming
- what goes in ‘discussion’ below?
- marwick’s compendium

3.1.1 discussion

3.1.2 exercises

3.2 D3, Processing, and Data Driven Documents

3.2.1 discussion

3.2.2 exercises

3.3 Storytelling and the Archaeological CMS: Omeka, Kora

blargle

3.4 What is Web Mapping?

A Web map is a geographic visualization that is supported by computational infrastructures. This form of mapping is fundamentally *powered* by the Web Axismap, 2018. It should come as no surprise that ubiquity of Web maps correlates with development of Web 2.0, a marked transformation in the way that that we interact with the Internet Cristina Aced, 2013.

As Darcy DiNucci remarked in 1999, content loading into a browser as ‘static screenfuls is only an embryo of the Web to come’. Web 2.0, thus is characterized by ‘interactivity’, ‘two-way communication’ often through devices that are ‘Internet-connected’ and a greater awareness amongst developers of ‘user experience’ and ‘interface’. Whereas the first iteration of the Web required advanced training to create a website, Web 2.0, as is often claimed, requires little or no expert knowledge to make a Web-ready ‘thing’ such as a website, a blog, or a Wiki. This is the context in which Web mapping, that is the Web-based visualization of geographic information became popularized.

One can argue that Web maps narrow the distance between professional map makers and non-specialist makers; yet some scholars have expressed concerns on the quality of geographic data that are now frequently and widely shared on the Web ref-name. Collaborative online platforms such as OpenStreetMap, for example, have drawn attention to the social dimension of knowledge making; how accurate is information that a lay-person or particular interest community has uploaded? Can we make real-world decisions based on those data? D Bégin, R Devillers, S Roche, 2013 suggest that volunteered geographic information reflects ‘contributors’ motivation and individual preferences in selecting mapped features and delineating mapped areas’, a situation that can enable scholars in examining and assessing the quality of those data.

For archaeologists, Web maps and publishing geographic information present challenges and opportunities. Archaeologists are aware that the data they collect through field studies often contain sensitive location information. They often express concern that publishing location information on sites of archaeological and historical interest can facilitate, if not result in, the destruction of those sites through looting. Looting and illegal trafficking of archaeological artefacts and human bones is an issue observed in many places ref-name

Recent developments in geovisual analytics that leverage the spatial dimension in data suggest that scholars can work with, and meaningfully analyze these data even where they contain sensitive location information Andrienko et al 2007. This situation opens enormous opportunities for archaeologists to develop tools that are appropriate for the publication of archaeological data. In the next section, we build upon the ethos of ‘openness’ and present an overview of map services, followed by a guide to making an interactive Web map with the Leaflet library. For an example of a Leaflet Web map, check out Open Context.

3.4.1 Overview of Map Services

Tiled map service and Web Map Service are two forms of Web-based mapping. A WMS is an interface that enables us (the clients) to request specific maps i.e. visual representations of geographic information from a geospatial database. The WMS server is called via a Universal Resource Link (URL) through an Internet-enabled desktop GIS. A request typically consists of the geographic layer (e.g. theme) and geographic area of interest. The response to a request results in *geo-registered map images* that are displayed and queried

within a browser. Because the map is dynamically drawn upon request, and because the server typically uses the most current information from several layers in the geospatial database, WMS maps tend to load slowly. [Toporama] (http://wms.ess-ws.nrcan.gc.ca/wms/toporama_en) is an example of a WMS server for Canadian topographic themes. [add image for WMS architecture?]

Tile map services such as TillMill Project, OpenStreetMap, CartoDB, and Stamen all use a rasterized image that is divided into 256 x 256 adjacent pixel images or ‘tiles’. Once generated, the set of tiles are stored on disk and thus they can be distributed rapidly to large numbers of simultaneous requests. Tiled maps shift attention to map aesthetics and smooth map navigation, trading in functionality such as layer order, map scale and projection. Alex Urquhart maintains a list of free tile services. [add image for tiles?]

3.4.2 Making a Web map with Leaflet

Leaflet is a JavaScript library developed by Vladimir Agafonkin for use with tiled maps. Launched in 2008, Leaflet has become widely used in tile Web mapping because it is easy to customize and develop interactivity with map elements, and because of its speed and simplicity when compared to a dynamically drawn WMS map.

To make your own interactive Web map with Leaflet, you need the following:

- 1) geocoded data saved as csv
- 2) GitHub account
- 3) tile map service

3.4.3 exercises

3.5 Place-based Interpretation with Locative Augmented Reality

yep.

3.5.1 discussion

3.5.2 exercises

3.6 Virtual Archaeology

What is Virtual Archaeology (VA)? ### Theory blah

3.6.0.1 Making, Wayfaring, Tacking and Archaeological Cabling

blah #### Materiality blah #### Phenomenology blah ### Method blah

3.6.0.2 The London & Seville Charters

blah #### Agency blah #### Authority blah #### Authenticity blah #### Transparency blah
 #### Paradata blah ### Practice blah #### Paul Reilly blah #### Anthony Masinton blah ####
 Grant Cox blah #### Richard Allen blah #### Daniel Pletinckx blah #### Nicole Beale blah ####
 Alice Watterson blah #### Costas Papadopoulos blah #### Ed Gonzalez-Tennant blah

3.7 Archaeogaming

Guest section by Andrew Reinhard, Meghan Dennis, Tara Copplestone, Colleen Morgan?

3.7.1 Discussion

blah ### Exercises blah

3.8 Social media as Public Engagement & Scholarly Communication in Archaeology

boo socmed

3.8.1 discussion

3.8.2 exercises

Chapter 4

Eliding the Digital and the Physical

yadda

4.1 3D Photogrammetry & Structure from Motion

vsfm

4.1.1 discussion

4.1.2 exercises

4.1.3 Going further

It is possible to make 3d models from archival film/video footage, although the quality of the resulting model may require a significant amount of sculpting work afterwards to achieve a desirable effect. It depends, really, on why one wants to build a 3d model in the first place. Archaeologists for instance might want to work with a 3d rendering of a building or site now lost.

The workflow has a number of steps:

1. obtaining the video (if it is on eg. youtube)
2. slicing the video into still images
3. adding camera metadata to the images
4. computing matched points across the images
5. triangulation from the matched points
6. surface reconstruction

The necessary software are all open-source or free-to-use programs 1. Youtube-dl <https://rg3.github.io/youtube-dl/> 2. avconv via libav-tools 3. exiftool <https://www.sno.phy.queensu.ca/~phil/exiftool/> 4. regard3d <http://www.regard3d.org/> 5. meshlab (for post-processing) <http://www.meshlab.net/>

At the command line in your instance of DH-Box, you can install the first three programs like so:

```
$ sudo pip install --upgrade youtube_dl
```

```
$ sudo apt-get install libav-tools
```

```
get exiftool
```

```
get and compile regard3d
```

```
wget https://downloads.sourceforge.net/projects/regard3d/files/Regard3D/0.9.5/Regard3D_src_0.9.5.7z
```

but you need `sudo apt-get install p7zip-full` to unzip it

SG - VSFM. Not regard3d. regard3d is gui based and can't do that in dhbox, least not on my end.

Step One Downloading from Youtube Archival or interesting footage of all kinds may be found on youtube and other video streaming services. Youtube-dl is a sophisticated program for downloading this footage (and other associated metadata) from youtube and some other sites. Find a video of interest. Note the url. Then: `youtube-dl https://www.youtube.com/watch?v=nSB2VeTeXXg` Try to find video that does not have watermarks (the example above has a watermark and probably is not the best source video one could use). Look for videos that are composed of long cuts, that sweep smoothly around the site/object/target of interest. You may wish to note the timing of interesting shots, as you can download or clip the video to those passages (see the youtube-dl documentation)

You might want to rename the downloaded file into something easier to work with. The youtube link above for instance is for a file called "Aerial views of the ancient city of Pompeii HD Stock Footage-nSB2VeTeXXg.mp4". Typing that out a few times will quickly become frustrating. We can rename the file using the `mv` command:

```
$ mv "Aerial views of the ancient city of Pompeii HD Stock Footage-nSB2VeTeXXg.mp4"
  "pompeiiifilm.mp4"
```

That is to say, 'move the "old file name" to "new file name"'. If you included file paths in the new file name, you'd actually move the file, too.

Step Two Slicing the Video into Stills

First, let's make a new directory into which we can put the still images that we are going to cut the video.

```
$mkdir frames
```

A quick search of the linux user community will give you an idea of the wide variety of things you can do with `avconv`, but for now, we are going to tell it to split the video into frames and put the frames into a new directory:

```
$ avconv -i name-of-your-video-here.mp4 frames/frame-%05d.jpg
```

The `-%05d` flag behind the word `frame` is a pattern for naming the output files: `frame-00001.jpg`, `frame-00002.jpg` and so on. Move into the `frames` directory and list the files to see the results. If you were able to use the same video of drone footage from Pompeii that we used, there will be 1914 images in the new folder. You might wish to explore the documentation for the `avconv` command (type `man avconv` to get the manual or visit the online documentation [here](#)). The command below will take a still every 5 seconds:

```
$ avconv -i name-of-your-video-here.mp4 -vsync 1 -r 5 -f image2 frames/frame-%05d.jpg
```

Step Three Adding Camera Metadata

We will be using `Regard3d` to stitch the images together. `Regard3d` needs to know the camera make, model, focal length (mm), and sensor width (mm). We are going to fudge this information with our best approximation. 'Sensor width' is the width of the actual piece of hardware in a digital camera upon which light falls. You'll have to do some searching to work out the best approximation for this measurement for the likely camera used to make the video you're interested in. Find the camera database that `Regard3d` uses (see the documentation for `Regard3d` for the location on your system). It is a csv file. Open it with a text editor (eg `Sublime Text` or `Atom`. not `Excel`, because `Excel` will introduce errors). Add the make, model, and sensor width information following this pattern: `make;model;width-in-mm` `Regard3d` reads the exif image metadata to work out which camera settings to use. Focal length is read from the exif metadata as well. We assign these like so, from the command line in your frames folder: `exiftool -FocalLength="3.97" .jpeg` `exiftool -Make="CameraMake" .jpeg` `exiftool -Model="CameraModel" *.jpeg` Note that the make and model must absolutely match what you put into the camera database csv file - uppercase, lowercase, etc matters. Also, Windows users might have to rename downloaded `exiftool` file to `exiftool.exe` and put it into their path variable (alternatively, rename it and then put it in the frames folder so that when you type the command,

your system can find it easily). Step Four Computing Matches Open Regard3d and start a new project. Add a photoset by selecting your frames directory. Note that when you used the exiftool, the original images were copied within the folder with a new name. Don't select those original images. As the images load up, you will see whether or not your metadata is being correctly read. If you get NaN under make, model, focal length, or sensor width, revisit step three again carefully. Click ok to use the images. Click on compute matches. Slide the keypoint density sliders (two sliders) all the way to 'ultra'. You can try with just the default values at first, which is faster, but using 'ultra' means we get as many data points as possible, which can be necessary given our source images. This might take some time. When it is finished, proceed through the next steps as Regard3d presents them to you (the options in the bottom left panel of the program are context-specific. If you want to revisit a previous step and try different settings, select the results from that step in the inspector panel top left to redo). The final procedure in model generation is to compute the surfaces. When you click on the 'surface' button (having just completed the 'densification' step), make sure to tick off the 'texture' radio button. When this step is complete, you can hit the 'export' button. The model will be in your project folder - .obj, .stl., and .png. To share the model on something like Sketchfab.com zip these three files into a single zip folder. On sketchfab, you upload the zip folder. Step Five Clean Up Double click on the .obj file in your project folder. Meshlab will open and display your model. The exact tools you might wish to use to enhance or clean up your model depends very much on how your model turned out. At the very least, you'll use the 'vertice select' tool (which allows you to draw a box over the offending part) and the 'vertice delete' tool. Search the web for help and examples for the effective use of Meshlab.

4.2 3D Printing, the Internet of Things and “Maker” Archaeology

yay

4.2.1 discussion

4.2.2 exercises

4.3 Artificial Intelligence in Digital Archaeology

4.3.1 Agent-based modeling (ABM)

Andreas: “Hello World!” Overview

4.3.1.1 Models, simulation, and the precursors of ABM

“Essentially, all models are wrong, but some are useful”

George Box, 1987

Empirical Model-Building and Response Surfaces, p. 424

- def system, model, formal model
 - The origins of computer simulation
 - Equation-based modeling, System Dynamics - ODE systems as models (mention Lorenz attractor, end with the idea of iteration of a process)
 - Algorithm-based modeling, Algorithms as behavioral models (mention cellular automata, Van Neumann, Schelling, etc)
 - The variety of algorithm-based models, much coincides with the variety of programming paradigm.
- a. Flow: spectrum between centralized (iteration of a single process, e.g. event) and distributed (multiple processes are iterated contextually by multiple entities).

- b. Schedule: spectrum between fully-scheduled (processes are initiated in a predetermined order) and event-driven (processes are initiated given a certain input). ABMs fall somewhere in-between these two spectra, though normally leaning towards distributed (e.g., using Monte Carlo methods) and fully-scheduled. Applications in several disciplines (see Wikipedia)

4.3.1.2 ABM in History and Social Sciences

- Emergence... Life game, deterministic chaos
- Saving the gap between atomism and holism, or individual and social structure
- Virtual laboratory for social sciences
- Examples
- Mention the cycles of enthusiasm and skepticism, and the tendency to develop closed circles
- Challenges and pitfalls

4.3.2 discussion

4.3.2.1 Why model with ABM?

- open with the idea that we are always modeling
- Mention review article ‘Why model?’

4.3.2.2 How to model with ABM?

- Parts of an ABM model (re-visit presentations)
- Define hypotheses, questions or simply the phenomenon of interest (identify the system)
- Define the elements and processes (you believe are) required to address the system (model the system). mention paradox Occam’s razor vs. emergence
- Express the chosen elements and processes as computer code (implement the model)
- Modeling is a process of constant iteration. Each stage is an iteration loop that is often repeated several times before jumping into the next step. The end result of development and exploration of a model may be relatively stable, often in the form of publications. However, this product potentially feeds a new modeling effort (and that is actually the main usefulness of models in the long run).
- ABM in python, Java, C#, or C++? in R? in NetLogo? (Any language can do it, really). Pros and cons, talk in terms of higher/lower level programming languages (still, they are all quite high-level).
- remember to talk about <https://www.openabm.org/>

4.3.3 exercises

- The Pond Trade (coastal settlements of variable size around a pond; cultural vectors; trade ships travel between the settlements, re-evaluating afterwards if the trip is worse or better than other previous trips (make table); they carry economic value and cultural traits from between the base and destination settlements)
- Pond Trade model repository: <https://github.com/Andros-Spica/PondTrade>

```
library(DiagrammeR)
grViz("diagrams/boxes.dot")
```

SG: code chunks didn’t render correctly through the bookdown generation process, which is baffling. check the original bookdown book for how to sort this out. removed the executable code chunk, put it into a different script, ran it, copied the output here. this is not optimal.

- Implement the model based on general definition (agents, variables, parameters)

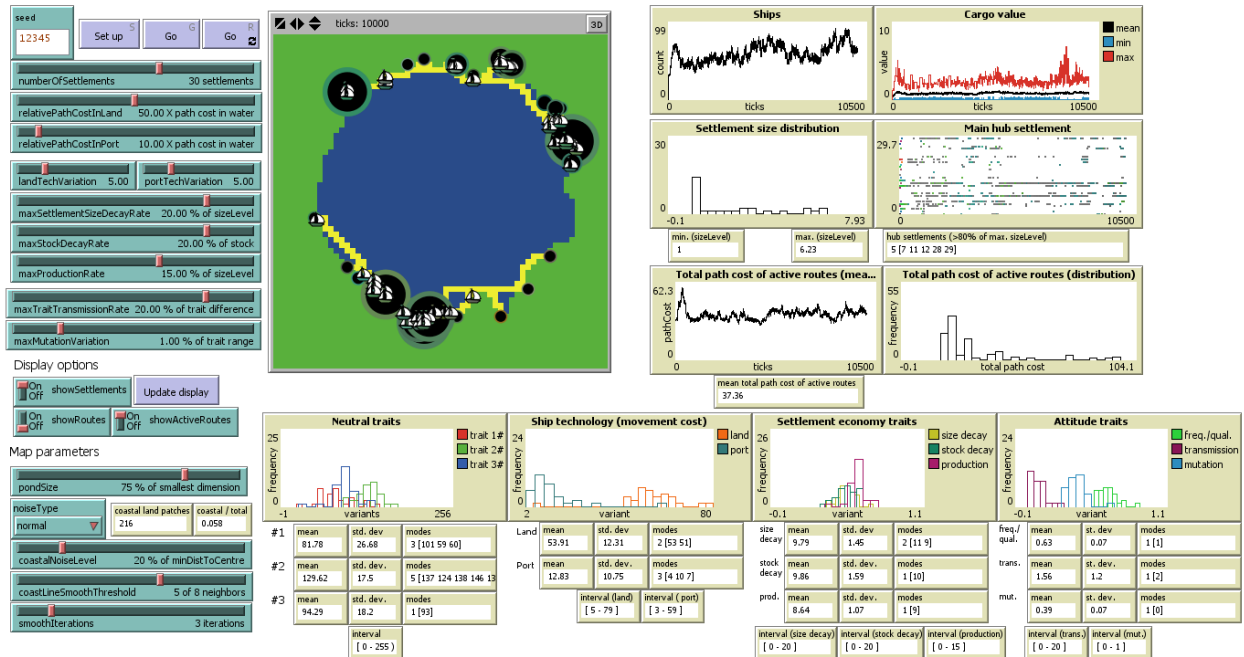


Figure 4.1: Interface of step 13

- Walk-through the implementation in NetLogo
- Simulations and experiment design
- Analysis and display of results (R example)

plot(1:10, 1:10)

- Next steps

4.3.4 Machine learning for image captioning and other classificatory tasks

blah

4.3.5 discussion

blah

4.3.6 exercises

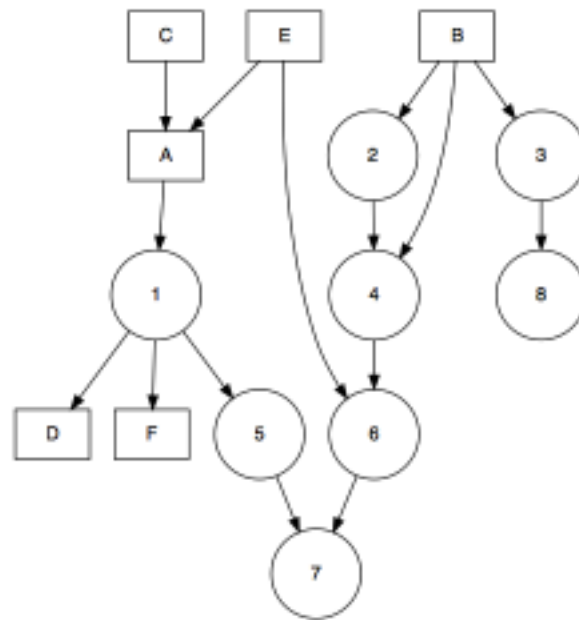


Figure 4.2: andros-first-diagram

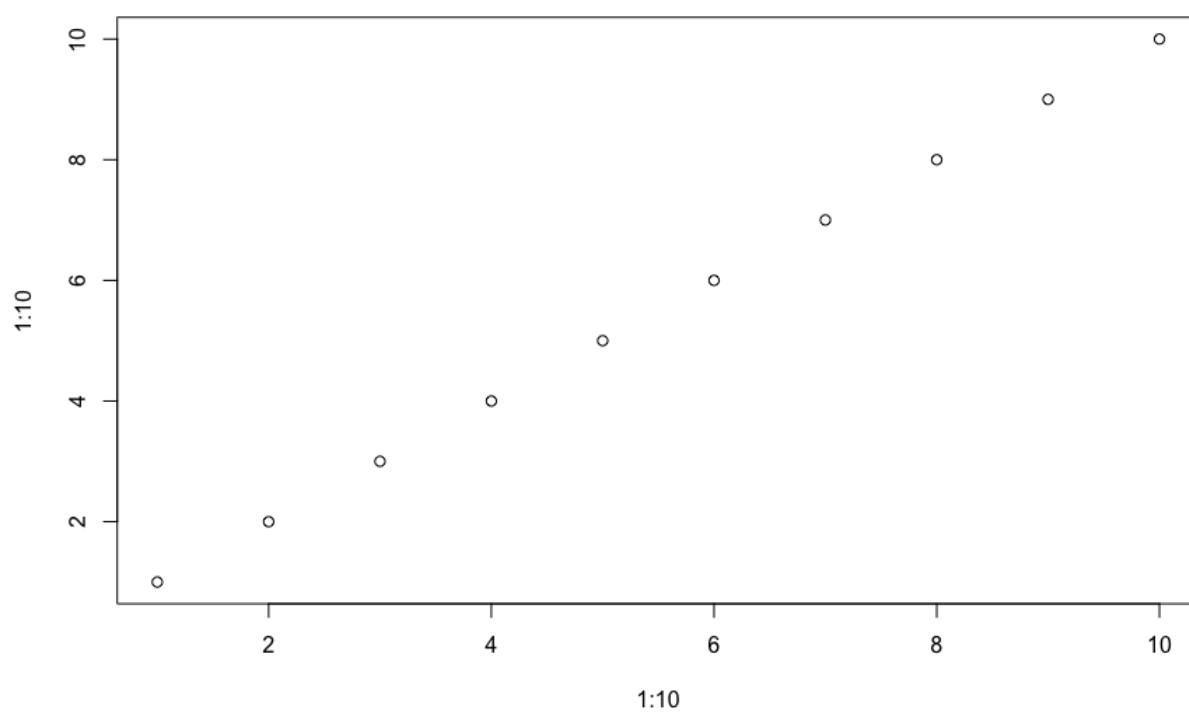


Figure 4.3: andros-second-diagram

Chapter 5

Digital Archaeology's Place in the World

blerg

5.1 Marketing Digital Archaeology

blag

5.1.1 discussion

5.1.2 exercises

5.2 Sustainability & Power in Digital Archaeology

the big ticket item.

5.2.1 discussion

5.2.2 exercises

Chapter 6

On the Horizons: Where Digital Archaeology Might Go Next

blargble

Chapter 7

Open Review

We are writing this textbook in the open, warts and all. We would be glad of all feedback, which you can make by using the Hypothes.is annotation bar on the right hand side of the screen. You will need an Hypothes.is account. Then, when you spot something that needs work or is in error or could be better (or even you just want to leave us an encouraging word) highlight the relevant text, hit the ‘annotate’ button, and leave your remark. Please also use the tagging function in Hypothesis to tag your comment with ‘ODATE’ so that we can collect the annotations together.

All annotations are public, and will be visible to other visitors on this site. If you have any questions about this, please get in touch with Shawn: shawn dot graham at carleton dot ca.

If you are comfortable with Git and Github, feel free to fork the source code, or leave issues or pull requests as appropriate.

7.1 Contribute

While the majority of this book is being written by Graham, Gupta, Carter and Compton, we know that we are leaving a great deal of material un-discussed. We would be delighted to consider additions to this present work, if you have particular expertise that you would like to share. As you can see, many sections in this work have yet to be written, and so we would be happy to consider contributions aimed there as well. Keep in mind that we are writing for an introductory audience, and that we are writing for a linux-based environment. Whether you are an academic, a professional archaeologist, a graduate student, or a friend of archaeology more generally, we’d be delighted to hear from you.

Please write to Shawn at shawn dot graham at carleton dot ca to discuss your idea and how it might fit into the overall arc of the present work by **January 31st 2018**. The primary authors will discuss whether or not to invite a full draft. A full draft will need to be submitted by **March 15th 2018**. We will then offer feedback. The piece will go up on this draft site by the end of the month, whereupon it will enjoy the same open review as the other parts. Accepted contributors will be listed as full authors, eg ‘Graham, Gupta, Carter, Compton, YOUR NAME, 2018 The Open Digital Archaeology Textbook Environment’.

References

- Boettiger, Carl. “Welcome to My Lab Notebook - Reloaded.” Website. *Lab Notebook*. <http://www.carlboettiger.info/09/28/Welcome-to-my-lab-notebook.html>.
- Caraher, William. 2012. “Archaeological Glitch Art.” *The Archaeology of the Mediterranean World*. <https://mediterraneanworld.wordpress.com/2012/11/21/archaeological-glitch-art/>.
- Deetz, James. 1965. *The Dynamics of Stylistic Change in Arikara Ceramics*. Urbana: University of Illinois Press.
- Ethan Watrall. 2017. “Archaeology, the Digital Humanities, and the ‘Big Tent.’” In *Debates in the Digital Humanities*, 2016th ed. Accessed February 23. <http://dhdebates.gc.cuny.edu/debates/text/79>.
- Evans, Thomas L., Patrick T. Daly, and MyiLibrary, eds. 2006. *Digital Archaeology: Bridging Method and Theory*. London ; New York: Routledge. <http://proxy.library.carleton.ca/login?url=http://www.mylibrary.com?id=29182>.
- Goldstone, Andrew. 2018. “Teaching Quantitative Methods: What Makes It Hard 9in Literary Studies).” In *Debates in the Digital Humanities*.
- Graham, Shawn. 2017. “Cacophony: Bad Algorithmic Music to Muse To.” <https://electricarchaeology.ca/2017/02/03/cacophony-bad-algorithmic-music-to-muse-to/>.
- Graham, Shawn, Scott Weingart, and Ian Milligan. 2012. “Getting Started with Topic Modeling and MALLET.” *Programming Historian*, September. <http://programminghistorian.org/lessons/topic-modeling-and-mallet>.
- Kansa, Eric C., Sarah Witcher Kansa, and Ethan Watrall. 2011. *Archaeology 2.0: New Approaches to Communication and Collaboration*. Cotsen Digital Archaeology. <http://escholarship.org/uc/item/1r6137tb>.
- Liu, Alan. 2004. *The Laws of Cool: Knowledge Work and the Culture of Information*. 1 edition. Chicago: University of Chicago Press.
- Lucas, Gavin. 2012. *Understanding the Archaeological Record*. Cambridge University Press.
- Mickel, Allison. 2016. “Tracing Teams, Texts, and Topics: Applying Social Network Analysis to Understand Archaeological Knowledge Production at Çatalhöyük.” *Journal of Archaeological Method and Theory* 23 (4): 1095–1126. doi:10.1007/s10816-015-9261-z.
- Montello, Daniel R., Sara Irina Fabrikant, Marco Ruocco, and Richard S. Middleton. 2003. “Testing the First Law of Cognitive Geography on Point-Display Spatializations.” In *International Conference on Spatial Information Theory*, 316–31. Springer. http://link.springer.com/chapter/10.1007/978-3-540-39923-0_21.
- Mullen, Lincoln. 2017. “A Confirmation of Andrew Goldstone on ‘Teaching Quantitative Methods.’” *The Backward Glance*. <http://lincolnmullen.com/blog/a-confirmation-of-andrew-goldstone-on-teaching-quantitative-methods/>.
- Ramsay, Stephen. 2011. *Reading Machines: Toward an Algorithmic Criticism*. 1st Edition edition. Urbana: University of Illinois Press.
- Samuels, Lisa, and Jerome J. McGann. 1999. “Deformance and Interpretation.” *New Literary History* 30 (1):

25–56. doi:10.1353/nlh.1999.0010.

Shawn Graham. 2014. “A Digital Archaeology of Digital Archaeology: Work in Progress.” <https://electricarchaeology.ca/2014/11/06/a-digital-archaeology-of-digital-archaeology-work-in-progress/>.

tjowens. 2012. “Discovery and Justification Are Different: Notes on Science-Ing the Humanities.” *Trevor Owens*. <http://www.trevorowens.org/2012/11/discovery-and-justification-are-different-notes-on-sciencing-the-humanities/>.