Open Source

A major source of inspiration for this work was the communities surrounding open source data and software. I also firmly believe that digital methods who depend on closed software are severely lacking in their capability. In the following section I will describe some of the benefits to using open source software in research, but I will also try to expand on these ideas to relate them to research itself. The vision is to do for research what open source does for software.

Since digital methods are computational in nature, the computer used in some analysis is often an important aspect of the process. There is the tendency to think that there might be some difference based on the operating system that gets used. I'm not entirely convinced that there remains much of a difference here, at least between Linux, Windows, or OSX. If there is any difference, it would be in how the hardware is utilized for computations, and in this way it might make a difference in terms of researcher work flow to use something like Linux which is highly customizable and controllable. There is also the benefit in Linux of being able to make some commands on the system level, but again this is probably something that relates more to workflow than it does to research. I'm also not entirely convinced that its realistic for researchers to be modifying their kernels, or auditing their OS source to gain some benefit from a Linux.

There is one instance where having an open source OS would make some difference, and that is purely ideological. Some Linux users feel very strongly that all computation should be something that a user has total access to in their systems. While on an enterprise level this is probably fairly impractical, on a research level this might engage some interesting ideas. In the section on API's, I mentioned there being a risk to users who do not question the effect that closing source might have on a research project. If it is important to researchers to understand and control every single step in their research, then using a closed source OS is simply not an option for them. For example, using an operating system like Windows in some digital research projects would involve the researcher to pass their computations through a layer which obscures exactly what is happening to their data. This is almost certainly a negligible risk, I really don't want to over sell this point, but it is a risk none the same. Practically speaking this is a non-issue, but this is why I say that it is so much more an ideological issue. This argument is also kind of silly to make, because at the hardware level there is almost nothing that is open. This means that literally, practically all computations go through a black box and researchers will never be able to claim total transparency. Again, not a siginificant enough problem to ring the alarm on digital methods, but something that researchers and consumers of research need to be aware of. The control that researchers give up isnt given to entropy or chaos, it is given to corporations. Much of the same issues brought up in the section on APIs can come back here, and again researchers need to ask themselves how comfortable they are giving control away to major corporate entities.

The more interesting side of the open-source debate is in the software side of things. Open source software, and the movements which surround it, are one of the more interesting things happening with computers in the contemporary world. There exists an ideology which argues that it should not be possible to 'own' software. Instead, software should be free to share and alter and share and fork and do whatever you want with because all it is is just some programming information. I am not going to argue for or against this ideology, but I am going to outline how open source software can benefit the research process. In this project, I write all of the scripts and code in R, an open source programming language and computing environment. I also intend to publicly share all of the code I have written. One of the things R is used for is it's data manipulation, analysis and visualization packages. The function of these packages are comparable to those which are found with SPSS or Stata. In the methods section I describe the suite of packages I use in R, and the IDE which I also used for this project called Rstudio. All of the packages and programs I used in this project are also open source, meaning I was able to open the source code and check for any mistakes or errors or other kinds of code that would not work with my project. I did not actually do this, but the point is that I was able to do it, and if I was able to so was anybody else, and since R is quite a commonly used language I am am satisfied that it does not contain something which might be malicious to my computer or my project. In fact, through Github and Arch Linux, I was able to compile a copy of Rstudio from source and install directly on my machine, and since the compiler itself is open source I can be fairly confident that the program I've built this project with is safe. More to the point, I would be able to do this on any computer, and because this software is open I would be able to do it without paying anything to anyone for it. The same things cannot be said for programs like Stata or SPSS. I don't want to go too far and suggest that all research should meet this level of scruinty for its source code, as I states before it is unreasonable and impractical to think that all research needs to be undertaken with strictly open source software. But it does say something about the repeatability and levels of access of this research. It wouldn't make sense to open a project like this up, if anybody wanting to contribute or audit it would not have access to the same materials. This also engages with the point that it costs nothing to use this software. If research is to be engaging on this level for others, it is nessecary that each aspect of the project be done in an environment which is available to all. Had I done the same things in SPSS or Stata for example, those who wish to contribute can expect to pay a fairly large licensing fee for access to the same software, not to mention that both are propriatary software and cannot be audited.

Another big reason that research benefits from using open source software is that it can breed innovation, in part due to the low barriers to entry. R-base for example, is a set of packages which are bundled together in R, these include basic statistical and analytical tools on a comparable level to what one may find in SPSS. The difference is that with R, third party developers can create a function and package it into a distributable format, upload it to a server and make it available for all R users to download and install on their own machines for use. This means that it is not only a modular program, but that there is no limit on what these modules can be. This is impossible on a program like SPSS, which has non-free modules and developers cannot write third party functions for the program. Of course, these third party modules are open-source, and can be audited at any time. The effect of open modules is two fold, one that a researcher can make sure the function does what it is supposed to, but I have also found that one of the best ways to understand a function is to review the source. The open-source nature of R also means that these packages can be expanded and updated at any time, and that the nature of what an R package can grow as far as developers can write, and then can be taken up at any time. 'ggplot2' is the star-child of R third party developer Hadley Wickam, a name I mention in the methods section, but is a great example of this. R already contained basic visualization tools in its R-base, but Wickam created a tool which not only helps him in his own research, but was able to make that available to all researchers using the same tools. While it is without a doubt that many of the same functionalities of R are available for other proprietary softwares, it is often that not one can do all and R might be akin to like a Swiss Army Knife. But that is not to say that R is the only programming environment/language which can boast a large user base and strong development. Python is another great example of many of the same features of open source data software. There are just as many wonderful packages for Python as there are for R, and in fact because both are open source, it's possible to translate many of the packages of one language into another. Ggplot2 for example has both a package for R and for Python, and so much of the same syntax can be used that visualizations written in R that use ggplot2 can be done in Python just as easily.

OpenData is another area of the open-source movement which I find interesting, and has many research implications I would like to discuss. Many times, data is the product of some private enterprise. If say a mining company takes Sonar readings of a certain tract of land using equipment they own, then the data that they collect is by all rights their own to do with whatever they please. If they chose however, they could release that data to the public and it would be open. If they submitted it to a much larger database which includes data about tracts of land from the entire region, and this database was accessible to all it would be said that they have contributed to an open database. This type of open-data movement is present in most forms of science, which benefits enormously from sharing resources like this, and sociology is no exception.

One question that Gurdstein brings up however is if researchers are able to make effective use of this data. A part of this question mirrors the concerns I brought up in the digital methodology section, while the data might be available to all, it might not be accessible to all. Much like digital methods, accessing datasets such as these might not be possible for those without sophisticated knowledge of data management and computer science. While it is often touted as a hallmark of the internet and digital culture which empowers all, Gurdstein argues that it might actually only empower those with the infastructure and skills ot make use of the data. I personally sway towards this argument too, because as noted before, many of the bottlenecks to access are not just at a knowledge level but also at a resource level.

The GEDLT is a great example I will use to illustrate this. From their website, “GDELT monitors print, broadcast, and web news media in over 100 languages from across every country in the world to keep continually updated on breaking developments anywhere on the planet”. The dataset that makes up the GDELT event database is quoted at over 100GB, and even the website states that users will require 'deep technical knowledge' and 'extensive experience' to make use of this data. GDELT.org fortunately makes a visualization and analysis service available, but the code for this is obscured and therefore a little dangerous. The data itself has been argued to be not of great quality as well, often getting representation of events incorrect through overrepresentation and double-counting. As I mentioned before, if I wanted to I wouldn't be able to process the data in the GDELT at all, while I have the memory I simply don't have the hardware nessecary to make many of the computations. I also don't have the required 'deep technical skills' that make this kind of analysis possible. Effectively, the barriers to entry for analysis on this data is so high strictly because it is so big means that it is a different animal that regular open data. This is a really important distinction to make, that while data might be open it doesn't mean that is is able to be used effectively. One must question if this is a supply side problem, or if it is a demand side problem. Whose responsibility is to to make data usable? GDELT does provide a reduced data set, which reduces all data down to a single day metric, but this is effectively a different data set than the actual GDELT. Is this problem a technological one? Can hardware capability, and more sophisticated software can bridge the gap between these massive data sets and casual researchers?

One of the most common questions about open source software, and one that will no doubt be thrown at open source research, is why would anybody contribute? If the whole project relies on contribution from people who seemingly are unrelated to the project, why would it ever survive. The answer to this question for software is likely the same for research, and that it is because people are much more connected to these projects than they seem. There is plenty of research which seeks to answer this very question, all seem to come up with siliar answer ($). One of the reasons is that it is a great way to learn. Software coding is a very difficult skill to develop, and one that requires a lot of hands on experience. Sometimes, those who are just starting out benefit most from being able to see the full code of working software to learn how it is done, much like how just watching somebody cook a meal teaches somebody a lot about cooking. Once a beginner starts to make contributions to the project, they learn from mistakes they make and how to effectively implement their ideas into software. If they are good enough, they start to build a reputation. This reputation is another reason that researchers have found people make contributions to open source projects. Often these contributors use their contributions to display their skills to potential employers, or simply to build up credit in the softwre development industtry in general.

A really important reason that people contributions to open source projects is because it is a piece of software that they use often enough that working on it benefits them directly, and the project indirectly. This is a case that I described earlier when arguing for the benefits of R and third party developers who create packages for their own sake, and simply commit changes altruistically. This is without a doubt probably one of the more salient reasons that people contribute, at least from the perspective of many active projects with a large user base. These contributions are not just made by solo coders either, large companies also make massive contributions. This is less obvious with programs like R, but with Linux this is one of the more common ways in which things get developed. Red Hat Enterprise Linux (RHEL hereafter) is a great example of this. RHEL is a distribution of Linux which specifically targets industry instead of personal uses, and is used quite widely. The RHEL website claims that many corporations, universities, and non-profits have relied on RHEL for any number of computational uses. These institutions use RHEL for their day to day operations, and often times make contributions to the distribution for their own benefit, which is also contributing to the Linux Kernel itself, a shared resource among all Linux distributions. These types of contributions don't just help these institutions who use RHEL, but all who use the Linux Kernal. What this means for the larger world is that developers using Linux benefit from those who contribute, and these developers are very wide spread. Google's Android operating system is based on the Linux kernel for example, which powers a great number of people's smart phones who all benefit from controbutions to open source.

Another reason people do these types of contributions is just for fun. Sometime the project themselves are not made out of some serious nature or use, and sometimes the people who many contributions are doing it strictly as a hobby. This is the same thing that powers much of digital culture, that being a part of a community and a shared project is often a rewarding experience for those in it beyond any type of reward that they might get from being a part of some larger project. It isn't really the case that this is what is going on for all of the open source projects that exist in the world, but it is certainly a factor for some. Some coders approach it as just a hobby, and an excerise in their skills.

One question that often comes up from those wishing to harness to power of open source software on their own project is how to facilitate this. There is no secret sauce to facilitating contribution that is unique to open source software, it is just the same as what one may find in any other form of project. It might be helpful to think of open source software development as a kind of conversation between developers, and asking how to facilitate contributions is like asking how to faciliate conversation. I personally think that the biggest factor that can lead to contributions is the level of interest that a project generates. The Linux kernal for example is used by so many users that there is an undeniable interest in the Linux kernal, and so there is a great number of contributions to it on all fronts. Some of these come from building up the weakest points in the Kernal, some of them are contrbitions which allow spport for newer hardware, some are just to fix mistakes and bugs in previous code. It is this constant use of Linux however which generates so much interest, and so the faciliation of contribution to open source comes from this.

One way that projects can contribute facilitation is by becoming a part of a network of contributors, on a platform which makes it easy for them to make contributions. GitHub is a perfect example of how platforms can facilitate contributions to projects. Github is a collaboration platform, created to make the use of Git easier. Git being the source code management tool created by Linus Tarvolds, creator of Linux. By using Github to manage their code, developers are sending an invitaation to other developers to take a look at their code and to use it as they see fit, or sometimes contribute to it. If they want, these code files can be copied to their own computers filesystem, and when alterations are made these changes can be comitted to the main branch of code. Many IDE's, including R studio, have Git extensions, allowing for easy sharing and collaboration on code projects. It is not nessecary for an open source project to be on github, a really big part of facilitating contribution is lowering barreirs to contribution, and being in an easily discoverable place.

One issue that comes up with this is the Free Rider problem. This project itself is a fine example of this. I have benefitted from the contributions of others to open source projects, while making no contributions of my own. This isn't just a problem of software, it is a problem in a lot of digital products and culture. Another good example of this besides Linux is Wikipedia, where many go as their first source for knowledge and yet not most do not make contributions. Some might be quick to say that Wikipedia could be so much better if everybody contributed something, but this isn't necessary. Running Wikipedia is not free, it costs money to run the website and serve content, and so Wikipedia makes up for this as a non-profit and asks for donations during some campaigns during the year. This is the area where the Free-Rider problem really comes into account for me. As Wikipedia grows and grows, how can they sustain themselves without the financial contributions of the people who use it. Free rider problems don't just exist online, they are everywhere in the world. I feel that the best solution is the same as the reason for faciliating contributions, that if there is enough interest the project will likely be sustained on contributors. I believe that because Wikipedia is used by so many, it is this interest which will likely sustain it.

Finally, a question that comes to mind when thinking about what all of this might mean for research. Who actually owns and controls the research? This is a really important question for research, which may have findings that could change the world, but it is also important for software which can impact millions of users. It is important that some level of control be kept over the projects, but luckily being open-source doesn't mean that projects are totally anarchic. With Wikipedia, absolutely anybody can edit any article, but that doesn't mean that these changes will be always accepted by administrators. Often times, these edits are simply reversed and things go back to how it was before a malicious or errenous contributon was made. For research, one has to question if anybody wants to truly 'own' research. I know that some researchers treat the projects like their children, but at the cost of the expansion or colloboration on that project I'm not totally convinced that it is worth it to retain total control simply for the sake of it. The potential that comes from transparency, and opening of digital methods to all who are interested not only helps communicate the findings and methods of research, but also helps to enable those who are interested to contribute or fix mistakes. This is so in line with traditions of the scientific method, where peer-review, repeatability, and transparency are so critical. I'm not arguing that all research pursuits consider open source methods, of course this is unthinkable in some circumstances, only that for those who practice digital methods this may be one thing to think about.