Plan of Concentration

Emerson
COLLEGE

Faza Byandika Hikmatullah Wijaya

Computer Science \ Data Analysis

December of 2020

	Plan Sponsors:
Matt Ollis	Associate Professor of Mathematics
	at Emerson College
Adam Franklin-Lyons	Associate Professor of History
	at Emerson College
Jim Mahoney	Professor of Computer Science
	at Bennington College
	Outside Evaluator:
Tim Schroeder	Professor of Geology
	at Bennington College

Contents

”Teruntuk Ayah dan Bunda tersayang, terima kasih sudah menyekolahkan sampai sejauh ini. Selalu ada dikala senang dan sedih. Sekarang, sudah waktunya untuk Ayah sama Bunda dibahagiakan. Sudah 16 setengah tahun belajar di sekolah, baru bisa bayar pakai skripsi ini dulu. Sisanya ngutang dulu. Semoga ada kesempatan. Aaaammiiinn”

- Anak Tersayang.

Introduction

Plan Description

Component	%
Gyro: Data-driven Nonprofit Expansion Using Assessor Database	60
Personal Data Awareness: Using Data Analysis For The Good	20
Computer System Exam	10
Algorithm Exam	10

Using already available public data, this Plan concentrates on helping a nonprofit called All In Energy to plan for expansion. Analysis was performed using Python Language and Jupyter Notebook to better present the result of the analysis. Driven by concern on how our personal information is kept and used by the big companies such Google, Facebook, and Apple, there is a paper portion that talks more specifically about gray areas of that problem. Such a gray area happens when, for example, a nonprofit organization uses publicly available data for the benefit of each individual. The paper discusses not only why it is important to ask questions about it but also talks about one of the solutions that I developed through my coding project.

Faza Byandika Hikmatullah Wijaya

Public Personal Data Awareness: Using Data Analysis For The Good

1 Introduction

How could you misuse one's personal data for the benefit of their own self interest? As the landscape of the social media industry grows, most never imagine the effects that this growth may cause. I am a student, learning computer science and data analysis. In this paper I will focus on specific research cases, in which the usage of data raises ethical questions. I will examine how despite the good intent behind these instances of data analysis one's attempts to use the latest ethical guidelines fail. I will use my work in All In Energy as the model case for these discussions due its similarities in the usage of personal's data. So, this paper proves that at the end of the day, questioning oneself for every action taken is ethical, and is still the most reliable framework. While also of course taking all the necessary precautions.

2 Ethical Problem With The Usage of Data

Some data misuse is well known and obvious like Cambridge Analytica. In 2013, a researcher from Cambridge University, built an app to hold a survey for his research. It was named thisisyourdigitallife. This app set out to do one thing: collect answers regarding respondents' physiology. However, thisisyourdigitallife also collects data from facebook app in the same phone. It was Cambridge Analytica. The scandal was a big hit throughout the news regarding a lot of people's personal data. The impact was even bigger in the United States since the scandal may have had a direct impact on the 2016 presidential campaign of Donald Trump. However, it is easy to say that whatever Cambridge Analytica did was ethically wrong. They

could have done this and that. It is easy to identify the incident, but this is an extreme case. The scale of data and operations they were doing were big enough that it is easy for the general public to quickly judge if that was wrong. But, how about the gray area? An area where a simple right and wrong do not seem enough. That is why this paper will not talk about Cambridge Analytica and its scandalous practice. This paper wants to push the limit of current available frameworks in dealing with problems that do not have an exact black or white answer.

2.1 "But the data is already public."

We live in a world where people insist that there is only right and wrong. The reality, however, is that our decisions sit on a spectrum. It is not black and white. It is black, white, and shades of gray. It is easy to say if something is right or wrong if the case is on either end of the spectrum, like Cambridge Analytics for example. However, how about data that is already in public? Do you remember a time when you can call anyone in a city by just skimming through the phonebook? Imagine, skimming through all of the numbers in the phonebook in order to count how many John Smith there are. Or you were wondering how many phone numbers consist of 666 in them. You certainly were working with people's personal data. Although you aggregated them, anyone could still trace back each data point if they wanted to. So, what is the ethical consideration with such situations? Does data being public negate any ethical concern because it's already public? Turns out, it is not the case.

In the data analysis world, one can argue that dealing directly with human subjects is

relatively rare. This is true due to the nature of the digital world where data is abundant and datasets exist everywhere. Many of these datasets directly relate to human subjects such as tweets and youtube videos. To positively interpret this one may say that more data scientists can do a lot more research which leads to the advancement of humankind. However, this abundance raises a concern. That is how data scientists can easily think that subjects are usually not harmed by using their data. As stated in a book called Bit by Bit: Social Research in the Digital Age written by Matthew J. Salganik:

> "Data Scientist, on the other hand, has little systematic experience with research ethics because it is not commonly discussed in computer science and engineering." [Sal19]

The need for a systematic moral compass has never been higher especially in this age of big data. This book also proposes *The principles-based* [Sal19] approach where the moral compass can be calibrated for whatever institutions you are operating at.

There have been several attempts to kind of give a written guideline for the principles-based approach that the book was talking about. They are two notable efforts: The Belmont Report and The Menlo Report. Both reports are an emphasis of 4 principles: Respect for Persons, Beneficence, Justice, and Respect for Law and Public Interest. The reports are a great minimum standard for a user of any data that may relate to a real human subject. However, in a real world situation, most of the time, a minimum standard is not enough.

For example during data collection in a digital manner such as tweets or instagram photos, it is really easy to think that collecting such public data will not affect the person

in any detrimental way. At least that was what a group of researchers was thinking. Back in 2008, there was research being conducted called ‘‘Tastes, Ties, and Time’’ (T3). Despite the efforts to anonymize and make the data that was collected to be as unique as possible to minimize the possibility of tracing the source down, the release of the data provoked a protest. Article from 2010 by Michael Zimmer titled ‘‘But the data is already public’’: On The Ethics of Research in Facebook describes in detail how the research turned sideways. The controversy started when a group of people identified at which universities the research was taking place by reverse engineering the data.

> ”However, little ‘‘extreme effort’’ was needed to further ‘‘crack’’ the dataset; it was accomplished a day later, again without ever looking at the data itself (Zimmer 2008a). As Hargittai recognized, the unique majors listed in the codebook allowed for the ultimate identification of the source university. Only Harvard College offers the specific variety of the subjects’ majors that are listed in the code- book, such as Near Eastern Languages and Civilizations, Studies of Women, Gender and Sexuality, and Organismic and Evolutionary Biology.” [Zim10]

A couple of things that stood up from the article were the first reaction from the group of researchers,

> ”In his defense of releasing subjects’ Facebook profile data, Jason Kaufmann, the principle investigator of the T3 pro- ject, has stated that ‘‘our dataset contains almost no information that isn’t on Facebook’’ and that ‘‘We have not accessed any information not otherwise available on Facebook’’ (Kaufman 2008c).” [Zim10]

The reaction supports Matthew J. Salganik in the book Bit by Bit. When it comes to digital data, specifically social media like Facebook, there are a lot of complications that data scientists, in this case researchers, are not aware of. Furthermore, when we use social media, we tend to feel safer, more private. It feels like we are wearing a mask and people cannot harm us in a way real interaction does. However, that is not the case at all. When you post something on Facebook publicly, it means anyone from anywhere in the world can see your post anytime. That is the illusion we are used to. Due to such false security or assumptions, researchers need to take an extra step when dealing with social media data.

Later in the article, Zimmer emphasised that future researchers should be more understanding in terms of the context of social media data. Zimmer talked in detail why the researcher failed to prevent such an incident happening. One thing that struck me the most is the fact that IRB had approved the research. There is no blame specifically directed to the IRB per se. However, it acts as a reminder that even a common strict practice such as going through a review from IRB cannot fend off problems. The article concluded that it is unclear if the Harvard IRB fully understands the context and potential threat that might occur from the research. Furthermore, the research had violated a fundamental aspect of ethical research.

> "As noted above, the T3 researchers did not obtain any informed consent by the subjects within the dataset (nor were they asked to do so by their Institutional Review Board). Further, as described in detail, the researchers failed to respect the expectations likely held by the subjects regarding the relative accessibility

and purpose of their Facebook profile information." [Zim10]

This is the key that made the research become a disaster. Although Facebook data is open to the public, it does not mean it will not do any harm when we collect them and do research on them. Some people have expectations that whatever is on Facebook should stay only on Facebook. This is the very concern that I personally feared myself during the coding project.

2.2 Project Encore

Project Encore encountered similar problems. However, contrary to Zimmer with the T3 case, Arvind Narayanan and Bendert Zevenbergen did not even get reviewed by the IRB. When you visit a news website, the website wants to know who you are, so that they can give you the best recommendation of news you might like, for example. This mechanism is called cookies. Since there is nothing to stop the news website from sharing the cookies, ad agencies also collect them from the news website. So, the next time you visit a completely different website, but the new website has the same ad agency from the previous news website, they will know it is you. Even though you never visited the new website, the website already knows your preference which means they can serve contents that suit you better. This is called a third party cookies tracker. The problem is, when I mentioned above that the intent is to serve content that suits you most, I lied. As I mentioned above too, the people who collected these cookies are ad agencies. They want to give you the most suited ads. They want you to buy. Click. Sign up.

Narayanan and Zevenbergen's idea was to use similar techniques to the ad agencies but

to track censorship on websites by some entities. They did this by going to websites' owners and asking them to put the project's tracker in the websites. By doing it this way, it gives Narayanan and Zevenbergen the ability to track the original data from each website, and what is missing when the user accesses the website. From the concept, there are already potential problems that are larger than what Zimmer talked about. For instance, similar to how the ad agencies track users, the encore project was able to track individual users. This means every click, every link that a user would be monitored by the Encore project's code. That is why Narayanan and Zevenbergen felt the need to get the IRB's review so that they could be sure that the project is ethically and legally responsible. However, Princeton and Georgia Tech's IRB said otherwise. There was a huge debate back then if IP address could be an user identification.

> "The Office for Human Research Protections has not issued a formal statement on whether IP addresses are considered to be personally identifiable information for purposes of the HHS protection of human subjects regulations at 45 CFR Part 46. However, for purposes of the HIPAA Privacy Rule, the HHS Office for Civil Rights has opined that an IP address is considered to be a direct identifier of an individual. Other European data regulations consider IPs as identifiers, and as such fall under the realm of the EU Data Directives (1995, 2006). This presents a challenge for international research and should be considered carefully by researchers and boards." [NZ15]

The project still carried on. They still asked themselves on how to capture the data in

order to proceed with their research while keeping the risk as low as possible.

3 Coding Project

In the Fall 2019, I started my internship with All In Energy as a Data Analysis Fellow. All In Energy was just about 6 months old and my job was to prepare their database structure to be ready to scale up. This was the time of my first introduction to Salesforce, a Customer Relationship Manager software which All In Energy founders think will fit best with their needs. Handling data was not new for at the time but handling actual customers' personal data was. I remembered clearly the day when one of my coworkers told the team a story about a customer expressing how grateful and happy they were because they did not feel cold anymore during harsh winter. That time I was working on uploading All In Energy's old database from a spreadsheet into a more sophisticated system in Salesforce and I recognized the name of the customer that my coworker was talking about. It struck me that despite how indirect my work was to the customer, I still played a huge role in these efforts of making people's life better. By then I realized I was working on something bigger than myself. This was the start of me going into the data analysis game.

3.1 All In Energy

Our habitable planet is dying and if you are smart enough then you should agree with that. The founders of All In Energy are smart because they realize and understand the scale of the problem. Back in 2018, Gabe Saphiro and Rouwenna Altemose realized that in order to

save our habitable planet from dying we need to bring everyone into the fight. One way of doing so is to make sure everyone has access to energy efficient services. However, the reality showed otherwise. Paper from 2019 written by Xiaojing Xu and Chien-fei Chen concluded that there is a lack of education in terms of energy efficiency programs among Low Income Households.

> "Third, in terms of space heating and cooling practices, income level influences heater and A/C thermostat control strategies. While LIHs are most likely to set one temperature and leave it unchanged, the number of households that program their thermostats increases with income. This result continues to hold when analyzing programmable thermostat owners only. Additionally, income level and homeownership result in different thermostat set-points. LIHs set their thermostats higher during the winter and lower during summer, both when at home and away, than MIHs and HIHs." [XC19]

For example, renters in Low Income Households will likely set their thermostat lower in summer and higher in winter compared to Middle and High Income Household thus resulting in 40% of their income going to energy bills. Also, Low Income Households have lower adoption rates of energy efficiency programs, specifically when a huge upfront cost is involved. That makes sense, how could they invest in such a huge cost when almost half of their income goes toward energy bills in the first place? That means only people who come from privileged backgrounds are benefiting from the energy efficiency programs from the state. This proves how the energy industry is lacking diversity both among consumers and providers. How can

everyone join the fight to save our habitable planet if only a certain group can contribute and benefit from it? Yet, even worse, the groups who will suffer the most are the ones that cannot access these programs. Higher energy bills, colder winter due to bad insulation, hotter summer due to lack of air conditioner are examples of the absence of equity in the industry.

Those problems need to be addressed and the Massachusetts government made a move back in 2013. The program is called MassSave. The MassSave program was founded with a goal to increase energy efficiency in the state. No wonder that as of 2020, Massachusetts is ranked number one by American Council for an Energy-Efficient Economy for 9 consecutive years. For my personal opinion, Masssasschusetts should be some kind role model for the rest of the country because the United States has become the biggest energy consumer throughout history. This fact can be less impactful sometimes because we tend to not take the blame for something in the past, especially a collective sum like this one. It feels distant somehow. That feeling of distance still hinders Massachusetts, despite becoming the leader of energy efficiency. The reality is far from perfect.

> "Since 2008, the Green Justice Coalition and allied organizations have noted that some residents—including renters, low-to-moderate-income households, and people with limited English-proficiency—are either not reached at all by Mass Save programs, or are not adequately served when they do participate in the programs. For as long as these programs have been in place, GJC has challenged the utilities and policy-makers to quantify and address this disparity and is doing so again during the current three-year planning process." [EG18]

One part of how MassSave does their job is to incentivize people to have assessments of their house so that utility companies can upgrade their weatherization. However, there is a problem. People are not educated about this information. The dilemma is, some part of the incentive comes from people's utility bills. Yes, in Massachusetts, the state's energy efficiency program is one of many tax bullet points that people need to pay. So, people pay for something they do not even benefit from. No matter how good a product or service is, it is going to be useless if nobody benefits from it. After working in All In Energy for a year, I saw that part of the problem is bureaucracy. Although we do not have a lot in common with lightning, we, humans, tend to take the path with the least amount of resistance just like lightning. That is why the food delivery industry like Uber Eats is a huge industry despite most people who use the service do have a kitchen in their place. The process of contacting the MassSave representative or Home Performance Contractor company to get qualified in the MassSave incentive is tedious. It must be a burden to call them and beg for money practically through a mysterious and complicated qualifying process. Not to mention doing a regular call to any representative sucks in general and takes a lot of time. So, as you might imagine the service still has a lot of room to improve itself before it can serve everyone in need. This is where All In Energy fits in.

As I talked about previously, Gabe Saphiro and Rouwenna Altemose are smart. They believe that including everyone into the fight is the only way to keep the fight going and succeed. All people should be in. So, they founded All In Energy back in 2018. How they fit into the process of making MassSave a better program is by connecting landlords and

renters to Home Performance Contractor companies. This way, the process for getting into the MassSave program can be made much smoother. As a middle man, All In Energy helps the three major parties in the MassSave program. First, All In Energy helps by outsourcing marketing and customer discovery out from the Massachusetts government and HPCs so they can focus on providing incentives and doing the actual weatherization work. Second, All In Energy helps landlords and renters get through the process by figuring out which channel from MassSave they qualify for, scheduling a service date with an HPC, etc. This way MassSave will have higher conversion rates in their upgrading effort to a more efficient energy society. Then the HPCs will have more customers with less effort needed in their marketing sector. In return, All In Energy gets a commission for the service. The most important one for me is, landlords and renters will have an easy way to access the program. Government's programs tend to be scary so having someone that understands how to carry us in the process will definitely help make us sign up to the program. In simple words, All In Energy is trying to bridge the gap between this huge bureaucratic nightmare and regular landlords and renters who sometimes forget to pay their electric bills. This position gives All In Energy a lot of opportunities to contribute to the energy industry.

With that being said, this may raise a question, does that mean the government has failed to fulfill its duty? Well, it is complicated. We need to understand first the scale in which the government usually operates. Even in a really specific region such as Massachusetts, it needs to serve over 7 million people from different backgrounds, different types of houses, needs, etc. For parents, when was the last time you felt like you could handle 5 different kids at the

same time? For students, when the last time did you make a fair agreement from all of your 10 class project team members? Let alone handling 7 million people. That is true, a government will not only consist of one person, but still the scale at which it operates is unimaginable. So, by design, bureaucracy will always be slowest to catch up with changes in technology and trends. Also, it means whatever programs it tries to run need to have a general enough purpose that everyone can benefit from the initiative. This opens up opportunities for the private sector. The private sector's job is to smooth out and individualize these programs and processes so that it will give the targeted consumer the best experience and benefits.

There is so much more potential that can be done with people's personal data. Specially in this era of big data, where every bit of human movements can be turned into data and people will analyze them somehow then turn it into something useful. Taking for instance Tesla. They started selling what they called Tesla Powerwalls which in fact is a huge battery pack that can power your house. The great thing about it is that it can store electricity no matter what the source of the electricity it was connected to. The grid, solar panels, your DIY wind turbine, you name it. Even better, Tesla will track your electricity usage and use that data to determine when to charge the battery best and by how much. This is a huge deal because the reality is the price of electricity changes throughout the day. Which means you need to charge your battery pack during cheap hours and use the battery pack during the peak hours. You can save up a quite amount of dollars over the years by this method alone, not taking into account if you have solar panels installed. Not only this will benefit us as customers who need electricity but also the utility company and the US grid system as

a whole. If we can spread out the electricity usage from the grid, the utility companies will not need to use those old inefficient coal power plants to satisfy demand during peak hours which would be better for the environment overall. That example was what made me want to go deeper into the usage of people's personal data.

As mentioned in the beginning, how could we use personal data for the benefit of ours? We talked about how the usage of data itself sometimes raised a lot of ethical questions. On the other hand, the big companies clearly make a big buck each year profiting off people's data. So, how could we make a turn? Is this the only way? This is where my coding project is taking its place. After discussing it with All In Energy, the goal is to help them expand their operations to a new city where the under served communities are. This is a great purpose to advance my cause of finding a way to benefit from our own data. By using my analysis then applying it to an AI model, this will be an example that this is not the only way.

All In Energy began operating 2 years ago. That time gave All In Energy learning experiences with its industry and business model. Although the people who are running the day-to-day operation may change, data that All In Energy produced remains. In order to reach broader customers and scale up the impact that All In Energy has made through the years, expansion is the way to go. The teams and their board of directors could have picked whichever city they felt wanted to. However, human intuition is not enough to decide such a big maneuver for the organization. The expansion will not only affect those who do the marketing in the front line, but also those who plan a budget for the year on the top of organization. This is where data analysis comes in. By bringing an approximate picture of

what is happening in the real world, data analysis is a great tool to confirm or build your intuition and assumptions when making a decision. What this coding project does is exactly that. The coding project will provide All In Energy with analytical reports based on data that I can get my hands on. Also, by taking that analysis, the project will result in a machine learning model that will predict if a city is worth expanding into a series of parameters. Let me elaborate on how this plays out.

As mentioned previously, big data, collecting personal information, and machine learning have some side effects that may cause problems in not only our general society, but also our personal life as individuals. However, that does not mean these tools cannot be used in a better way. Just like another tool out there, it can be used for good or bad. A car can be used to drive a sick person from their house faster, or it can be used to kill people by driving through them. That is why my coding project will give an example that there is a better way to use them. Every year the state government will evaluate each building in the state to determine the building's value. The result is the assessor database. Massachusetts specifically provides a great access to this database, we can download them from the state's website. The database includes: building address, value, type of building, owner's name and address, etc. This raw data needs to be processed. The goal is to aggregate the raw data into more statistical data from each city so that we can analyze the data using statistical tools.

After collecting the data as mentioned in the previous paragraph, the next step is to analyze the data to give a sense of how it will answer the questions. This process is called Exploratory Data Analysis. This is the time where we explore the data only with curiosity.

Although machine learning is black box that we can throw things into and then it will do its magic to solve problems, the problem is, if we throw garbage into the box, it will also spit out garbage as the result. That is why one of the goals in the EDA process is to find patterns in the data using various statistical techniques. Fortunately, humans are really good at finding order out of chaos.

Another set of data that I gathered was demographic data of Massachusetts. The data comes from the census data which fortunately was new as this paper was written. The demographic data consists of the percentage of Spanish speakers and low income population for each city. The data varies a lot in terms of the confidence interval for each demographic category for each city. This means some cities have more accurate data due to larger sample size and some don't. Since a big confidence interval will not bring interesting data due to its inaccuracy, I filtered them down so that only data with confidence intervals less than 5% is left.

Due to privacy concerns with the assessor database specifically, I will not include my cleaning up process in the final bundle code. However, if you are interested in how I aggregated the data, feel free to contact me for further information. Although I did not personally collect the data, the government did, the question of whether IRB reviewed was needed or not arose. The data is actually public data, whoever you are, you can access it through the government website and do something with it. The problem is, the data consists of personal information such as names and mailing addresses. Since we can connect a person's personal information with an address, there are a lot of things that can go wrong from here. For

example, credit cards. If you happen to have someone's credit card, you can look up their name in the database and use the address as the billing address. It might fail once, but if you are a hacker who has access to thousands of credit card numbers, one of them will stick. And that is not what we want to happen.

I actually do not need the IRB review since I will not publish and use the individual data from each person rather, we only used their data in the aggregate. However, since I will provide my analysis to All In Energy, they may do something with the data and interact with specific people based on that data. That is why such ethical questions still arose, so I contacted the Emerson IRB to see what they think about this. After questioning about the details of what I am doing with data, here is their answer:

> "I ran this by another member of the IRB, and we're in agreement that cases such as this would not constitute human subjects research as there is no intervention or interaction with subjects, and the data is publicly accessible. The non-profit may choose to conduct research off of your gathered data, but that would be separate from what you would be doing." [Mat20]

As expected, the IRB concluded that there is no need to review the case and I could proceed. Stating that this case is similar to what the encore project went through is overestimating. The data is public and there is no intent to use the personalized data. So, in this situation it makes sense why IRB decided the case as they did.

Despite what the IRB said, there's still concern in the project as a whole. Whatever written above is a result of reading a couple of resources and I specifically am looking for

the answer to my ethical question about the usage of data. What about other people like me? Or another data scientist or coder with data as their Sunday football games? A lot of programmers and entry level data scientists are self taught. They can even work with real data on the internet or even collect data themselves from their bedroom. Not all of them will ask the same questions like I did. That is why my hope is that this paper will not only raise the awareness of advanced researchers who do deep research in particular areas, but also for people who just started realizing the power of data in our current digital world.

4 Conclusion

The three examples above show a couple of instances where usage of data could go in a spectrum where there are multiple angles to see it. The lack of concrete frameworks for researchers in today's era, where data can not only be obtained from physically examining people but also from remote places sitting down in front of a computer without ever meeting the subjects, creates a wider gray area where ethical questions are subject to personal questions. Asking yourself the ethical questions are a good start before further implementing whatever you plan to do. This practice puts the responsibility on you. Not the IRB. Not your university guy. Not your organization. As the Bit by Bit book proposed as The principles-based approach, the team from Project Encore, implemented the practice. This will ensure that even if the frameworks fail to meet the expectations of ethical standards, people who do the research will still operate on the ethical values: Respect for Persons, Beneficence, Justice, and Respect for Law and Public Interest. With that being said, as we operate in a world

where being perfect is almost impossible, but there is our obligation to do our best to be as close as possible to be perfect. So, taking all the precautions necessary to mitigate the possibility of unethical results or practices are crucial too so that what happened with the case that Zimmer brought will never again to the future generations.

Bibliography

[EG18] Khalida Smalls Eugenia Gibbons. Timely new report shows substantial disparities in energy savings among massachusetts communities, Apr 2018.

[Mat20] Eric Matthews, Nov 2020.

[NZ15] Arvind Narayanan and Bendert Zevenbergen. No encore for encore? ethical questions for web-based censorship measurement. *SSRN Electronic Journal*, 2015.

[Sal19] Matthew J. Salganik. *Bit by bit: social research in the digital age.* Princeton University Press, 2019.

[XC19] Xiaojing Xu and Chien-Fei Chen. Energy efficiency and energy justice for u.s. low-income households: An analysis of multifaceted challenges and potential. *Energy Policy*, 128:763–774, 2019.

[Zim10] Michael Zimmer. "but the data is already public": on the ethics of research in facebook. *Ethics and Information Technology*, 12(4):313–325, 2010.

Gyro: Data-driven Nonprofit Expansion Using Assessor Database

1 Introduction

Coming from the gyroscope effect, Gyro has sliders that adjust automatically to keep the weight in balance. Also, the name comes from my passion in gyro as a food in general. Gyro is a Jupyter Notebook that is specifically written to be used for All In Energy. This is a documentation of how I help All In Energy by providing data-driven analysis for them to expand their operations. In order to understand this section, it is recommended to open the live Jupyter Notebook. The Data analysis is available here. The final product (Gyro) is available here. The password is **linustorvalds**. Both Jupyter Notebooks are needed to be re run by clicking the fast-forward button next to a drop down title markdown. Please kindly wait until all cells are executed indicated by the website icon on the top turned from a hourglass icon into a book icon.

1.1 Tools

Python is a high level, general purpose programming language, which makes it easy to learn and use without sacrificing its versatility. I am choosing Python for this project, firstly, because I have been using it for at least four years now. Secondly, in the data science community, Python is the industry standard for doing analysis and machine learning. This means that there are a lot of open resources out there that I can access easily. That being said, Python is just a "screwdriver," if you will. It is versatile and can be used to build anything, but you need something else if you are working with a specific problem or project.

This is where Jupyter Notebook and QGIS come into play.

Jupyter Notebook is a browser based interface people can use to write code alongside markdown paragraphs of text. I often find this useful when I want to give an explanation about my analysis of certain graphs or numbers produced by the code. Jupyter Notebook also helps me organize the code in a way that people looking at it can understand what is going on.

QGIS is a software that makes maps. GIS itself means "Geographic Information System," and is a standard used to process and visualize data geographically on earth (at least until we colonize the Red Planet). QGIS is an open source interface for utilizing the GIS standard, meaning there is a huge community of users to ask if I have any problems with it. This tool allows me to visualize my data on a real map and analyze if there is anything going on. Converting the numbers into a tangible form also allows me to connect more with the people I am trying to help and make an impact on. The numbers are representative of real human beings (or at least their houses). Although I will not use QGIS in the final form, it helps me to shape the data into a format which is easy to work with.

Docker is an industry standard software program used to deploy applications in the cloud. The idea is to simulate and isolate the environment where the application runs best (developers should not have to worry about what the environment of the deployment server will look like). It also prepares applications to be scaled up easily by allowing you to clone the application and run it on a distributed system.

1.2 Data

Every year the state government evaluates each of its buildings to determine its value. The result is the assessor database. Massachusetts provides easy access to this database, so we can download them from the state's website. The database includes building addresses, values, types of buildings, owner's names and addresses, etc. However, this raw data needs to be processed. The goal is to aggregate the raw data into statistical data for each city so that we can analyze it using statistical tools.

Another set of data from Massachusetts that I gathered was demographic data. This comes from the Mass 2020 Census (fortunately recently conducted at the time of writing). The demographic data includes the percentage of spanish speakers as well as low income persons for each city. The data varies a lot in terms of the confidence interval for each demographic category for each city. This means some cities have more accurate data due to a larger sample size, and some don't. Because a large confidence interval will not make for interesting data due to its inaccuracy, I filtered it so that only the entries with confidence intervals of less than 5% were left.

Due to privacy concerns with the assessor database specifically, I will not include my data cleansing process in the final code bundle. However, if you are interested in how I aggregated the data, feel free to contact me for further information.

2 Exploratory Data Analysis

Data is abstract and chaotic. The rows and columns in the Massachusetts Assessor Database are undecipherable in their current form. The best way to understand this chaos is to visualize the data in a way that humans can understand, data visualization. However, before jumping right into visualizing a bunch of cool graphs, it is always a great idea to point out what it is we are looking for. In this particular case, All In Energy has a specific set of rules for which kind of houses are eligible for their programs. Most of these parameters or requirements are linked to what kind of services are suitable. These requirements will filter out a lot of potential households, which reduces the chance to make them sign up. Unfortunately, out of those potential households, not all of them will sign up. Based on All In Energy's previous experience in Boston and Cambridge, the reality is that they will not be able to serve all 100% of these houses, no matter how great they are. Only a certain percentage of all potential customers can be converted into sales. That is why we need criterion to maximize the chance.

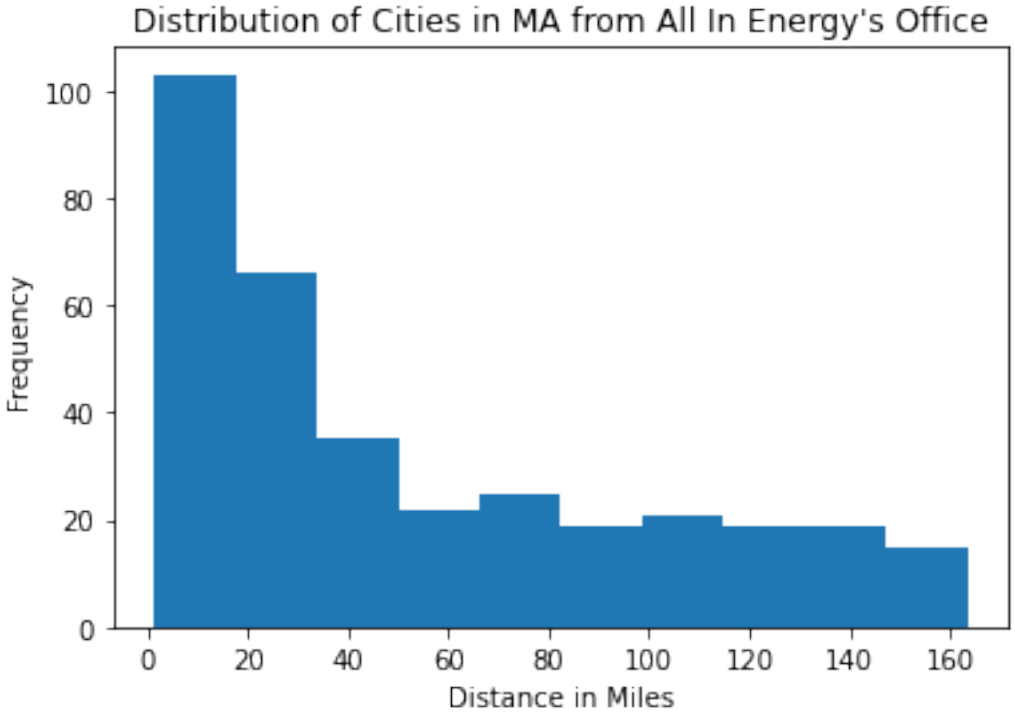
2.1 Criterion

2.1.1 Distance

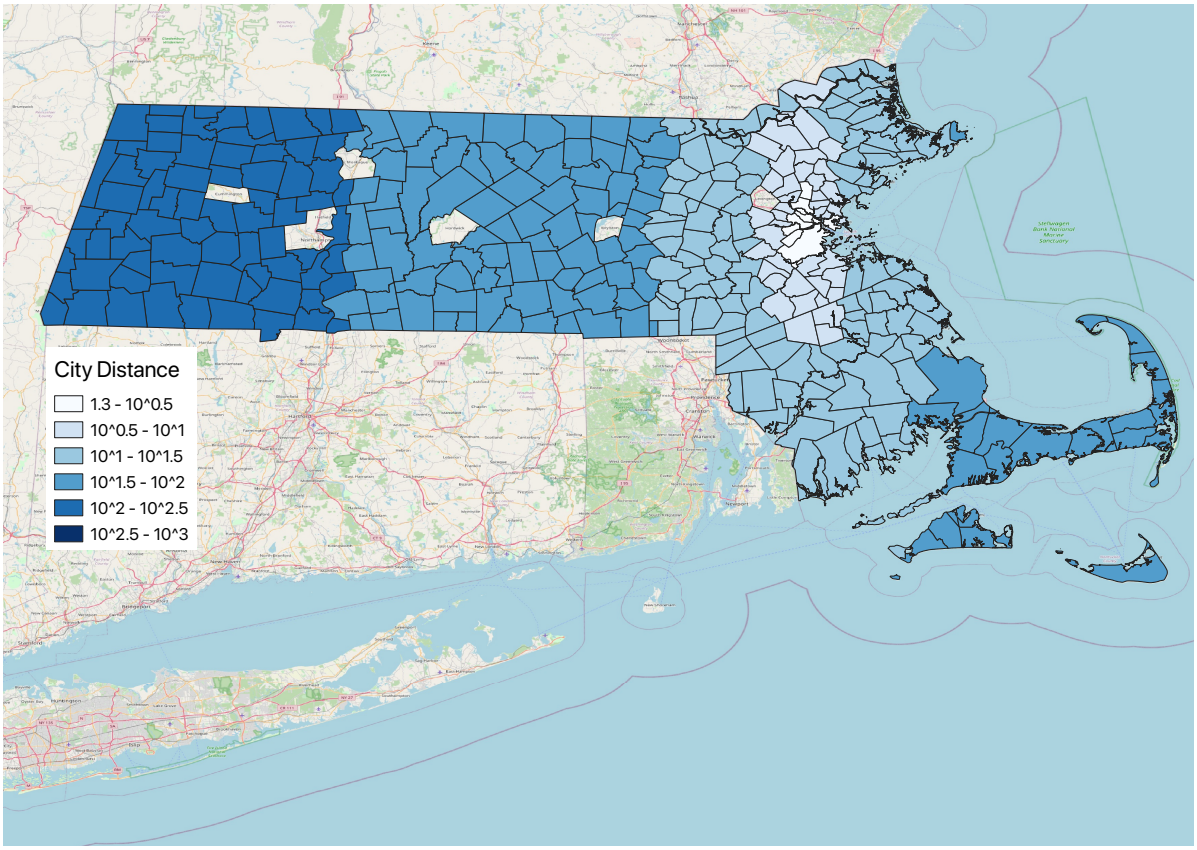
One criterion is the distance from the All In Energy office at 730 Commonwealth Ave, Brookline. Distance is not a huge factor in choosing a city because usually All In Energy will have a local team that will reach out to that specific city, and they are most likely to be located in the city itself. However, it is still a factor to consider because the further the

city is, the harder it is for the co founders, who are located in Boston, to be present and accommodate communication directly. Therefore, distance is a useful metric.

Unfortunately, the earth is not flat. It is a giant rock that has an uneven surface, at least on a human scale. The shortest distance between two points on a curved surface like earth is called geodesic. Fortunately, the dataset has coordinates for each building. Even better, there is a module in Python which will take care of the geodesic calculations. So I apply the operation to calculating the distance between each building and All In Energy in Brookline. Then I take the average of each city, and this is what it looks like:



An interesting result crops up where it happens that more than 200 cities are in the 60 mile range. This means the All In Energy team can reach a lot of cities in a less than 3 hour two way trip. Many options means a lot more to explore. Mapping each city's value into an actual map reveals another interesting result:



The gradations show us that there are many more cities that are close to the All In Energy office to the north and south than to the west.

The decision to take this approach instead of simply calculating the distance between the center of each city to All In Energy in Brookline, Boston is because we are assuming each building will be visited once before the return trip to All In Energy. This way, the distance will represent the average distance that an All In Energy staffer takes if they want to do a canvassing in that particular city.

Calculating the geodesic distance to each building gives a pretty interesting result. However, there are a lot of improvements that I can make in the future, such as using the Google Maps API so the value of the distance will be calculated based on the driving distance. This

is important because in the real world the closest distance between two points depends on which streets you are driving.

2.1.2 1-4 Family Buildings

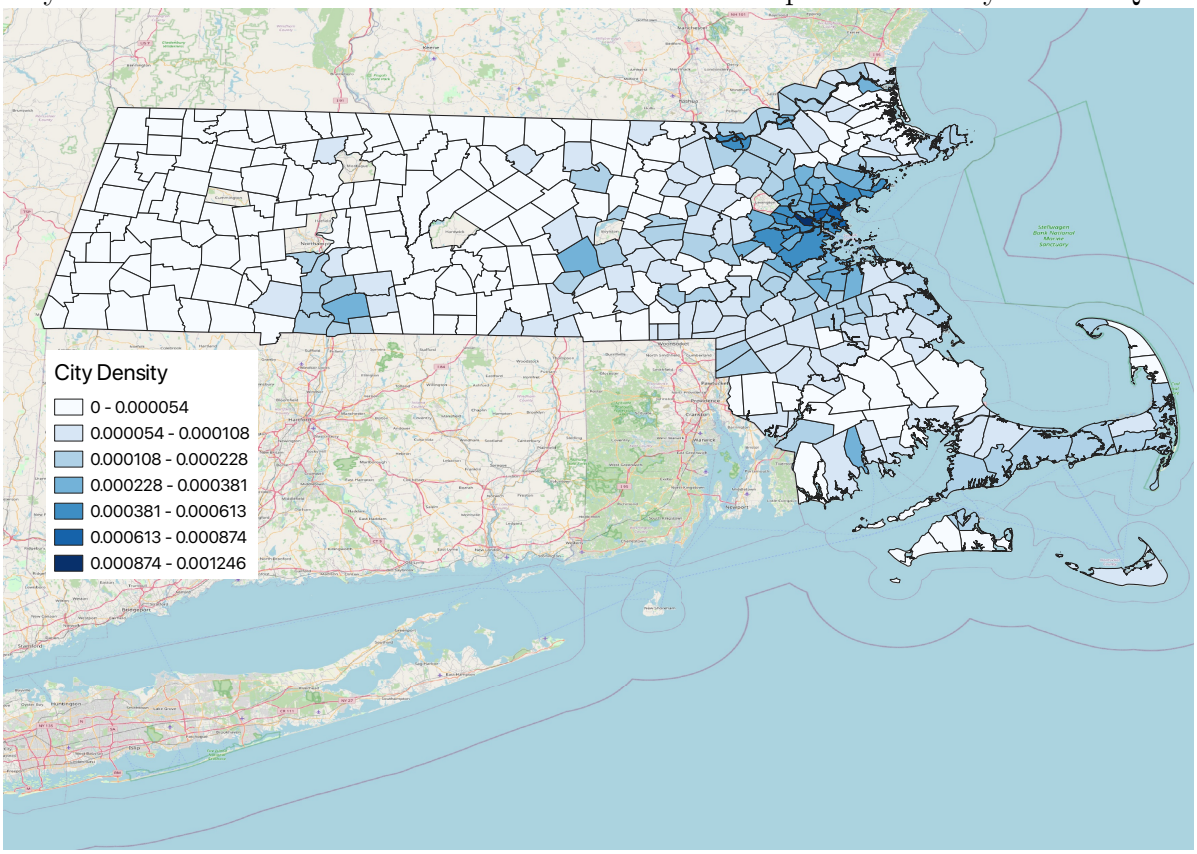
One of the requirements for the MassSave program is that the residence must be a 1-4 family building. This will rule out big apartments with many stories. The ratio of eligible to ineligible buildings will be one of the most crucial statistics. We do not want to go to a city where there are a lot of buildings when few of them are qualified.

Knowing the number and category of buildings, and the total area of each city, we can approximate how dense a city is. This is useful because as part of marketing strategy, All In Energy has used the traditional approach of knocking on people's doors and asking them to sign up to the program in person. That is, before the Covid-19 Pandemic began. At the time of writing, there is not a plan to continue in person canvassing. If All In Energy resumes this practice, knowing which city is the most dense will help the team. Reducing the need to move in a wide radius to get the same amount of people. Currently, the assumption is that the denser a city is, the more buildings are clumped together, however I understand that this might not always be the case. Before taking this approach, I tried to use some algorithms to find how many big clumps were in a certain area, however it was not consistent. The algorithm I used took as an input the number of clumps to search for, which defeated the purpose. So for our purposes using the density of each city will give us the approximation of that information.

The number of potential customers in a city is also an important statistic that we can get

from knowing which city has the most buildings. This is important because of how limited All In Energy's budget is to do the expansion. In business, there is something called customer acquisition cost. When bringing on a new customer, a company calculates the amount of money that is needed to do so. If a city does not have enough potential customers, the expansion may result in loss. Although All In Energy is a nonprofit organization, the cost to operate their business is not zero.

My hypothesis is that the cities that surround Boston should be denser than those in the countryside like cities in central Massachusetts. Here is a map of the density data in QGIS.



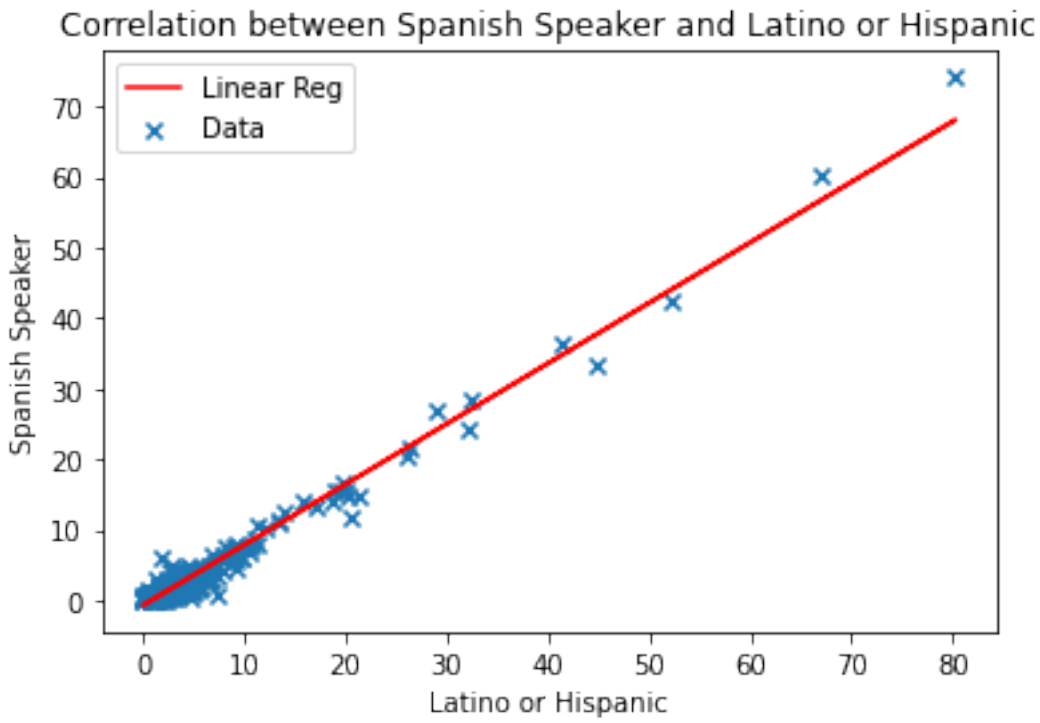
Although the color gradation is not as obvious as that on the distance map, the cities around Boston have more qualified buildings than further cities. One of the possible reasons for this

is that the commuter rail reaches all the way to Worcester, which happens to have many more qualified buildings than most of the cities around Boston.

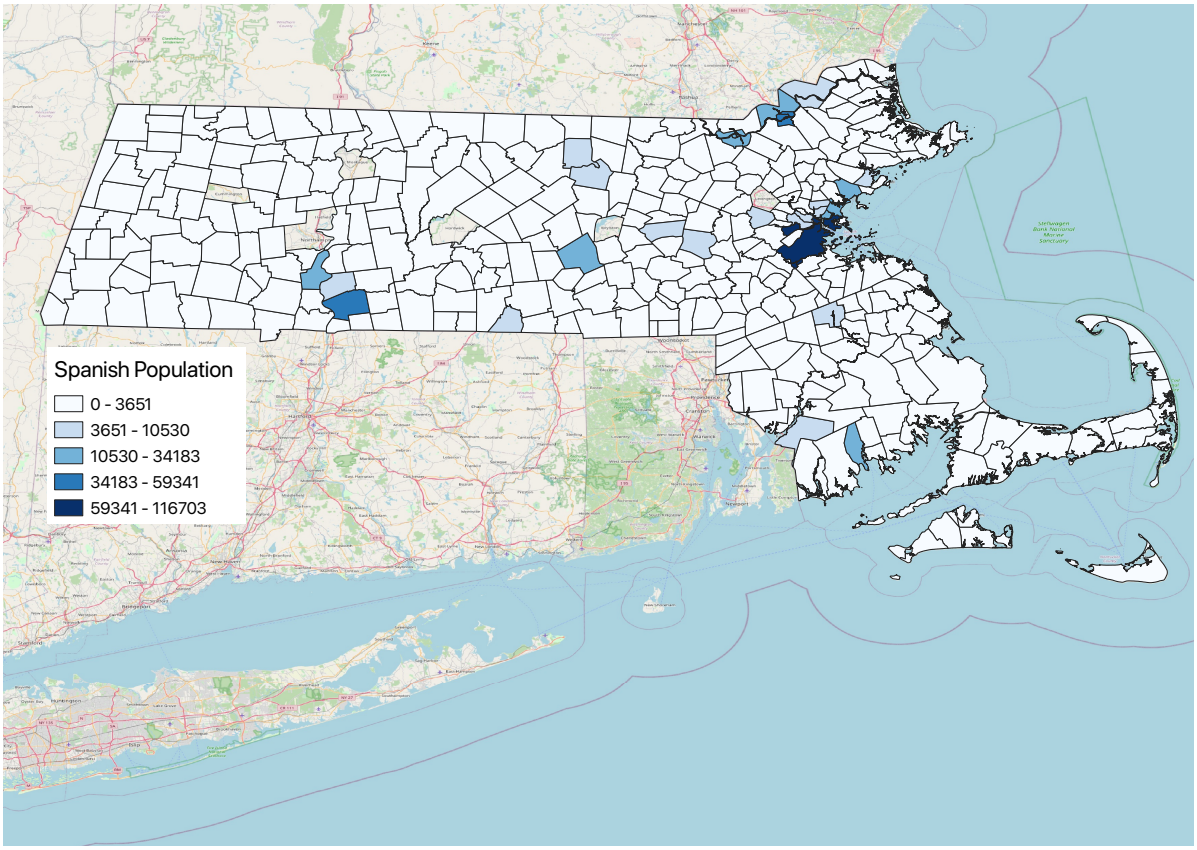
2.1.3 Spanish Speakers

One of All In Energy's goals is to bring the energy efficiency industry to under served communities. This is why the third criterion is the percentage of the population that speaks a second language. The current All In Energy operation focuses on spanish speakers specifically. This was reflected by the huge success of the Lawrence and Methuen operation on the August-October operation.

In order to observe this criterion we look at two statistics that the data provides: the percentage of Spanish speakers and the percentage of Hispanic population. By using linear regression, we can determine if there is a strong correlation between the two. If that is the case, it means that I can choose either one and it will represent roughly the same data. Below is the graph showing the relationship between the two with the line of the best fit.



From the graph we can see that there is some kind of relationship between the two. In order to confirm the relationship, we need to know the correlation coefficient. After running the code, the correlation coefficient shows 0.98, meaning they are almost an exact match. So, I proceed with the percentage of Spanish speakers from each city. This leads me to this map that I generated in QGIS:

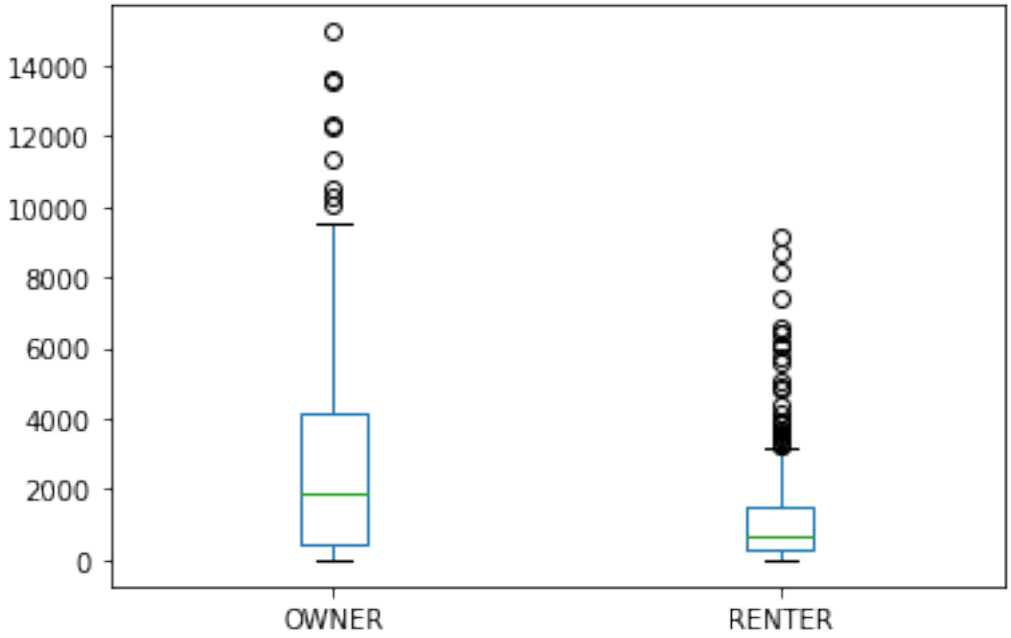


This map represents rough estimates of the populations of Spanish speakers in each city. I calculated them by finding the ratio of Spanish speakers to the total population for each city. Seeing the map, it shows that Massachusetts has a fairly homogeneous Spanish speaker population. There are some exceptions to this such as Worcester, Methuen, Lawrence. This means that All In Energy does not have a lot of options if they only want to focus on a large population of Spanish speakers.

2.1.4 Renters

Low income renters will tend to be less energy efficient than owners. That is why I think the criterion of renters and owners is important to put in the model. Take a look at the

distribution of renters and owners across Massachusetts:



The graph shows that across the state there are more owners than renters in any given city. This is shown by the difference in the mean marked by the yellow line. Therefore the metric of renters and owners would not have much of an effect on the eligibility of a city.

2.2 Analysis Conclusion

In order to determine which city to approach, we need to rank them. I chose to give each city a score based on their summed ranks in each of the criteria. For example, if Boston ranks first in the distance criterion, it will receive 10 points. Then if it ranks third in the Spanish speaker criteria, it will receive an additional 7 points, which would total 17 points. If we calculate the top ten with this method the table looks like this:

City	Score
WORCESTER	28
SPRINGFIELD	27
LOWELL	24
SOMERVILLE	20
BARNSTABLE	16
CHELSEA	13
MEDFORD	11

The table shows the top seven cities ranked by their total score. Boston, Lawrence, Methuen, and Cambridge are not included in the calculation because All In Energy is already operating on those cities.

3 Gyro

Although ranking cities by score works, there is one problem that may occur. In a rapidly expanding organization priorities can change quite frequently. What if finding Spanish speakers becomes the priority? What if distance becomes more important because All In Energy wants to go back to canvassing? With the current ranking system there is no flexibility in which of the criteria is most impactful. That is why I created Gyro. The goal is to provide All In Energy a tool to find the best city to expand to, with the capability of prioritizing certain criteria.

3.1 How It Works

The magic is in the weights. Instead of allotting each criterion a fixed score no matter what it is, I added a weight to each of them. If All In Energy wants to focus on Spanish speakers they can move a slider and make the Spanish speaking criterion have the most weight. This approach enables a lot of flexibility for All In Energy to explore. In terms of data, there is no change. It uses the exact same data from previous analysis, the only difference being the presentation and the output that it produces.

3.2 Deployment

After analyzing the data and developing Gyro, I wanted Gyro to be available for All In Energy whenever they need it, with or without me on board. That is why I want to deploy it to the cloud. There are many services like Binder and Google Collab that provide a way to deploy a Jupyter Notebook in the cloud, however I had no luck making them work. One of the reasons is that I have less control on how the Jupyter Notebook is built. My Jupyter Notebook has a lot of extensions which are not available on the services mentioned above. I have tried in many ways to install the extensions using free software but none have worked. I am pretty sure there are paid services which I would be able to have more control over, but in this particular case that option would be overkill. Instead, I deployed it on my own server back home in Indonesia.

I have this old PC from around 2008. The last time I went back home, I turned the PC into a server. By putting a linux server OS on it, then configured my router, now I

have remote access to it from around the world. I also bought a domain with my name, Jimmyganteng.com. With that set up, I deployed Gyro using Docker containers. The idea is to virtually replicate the environment that I used to develop the application. However, not to be confused with a virtualization on the OS level, Docker only runs a container which has its own kernel. Now, Gyro is alive.

3.3 Future Improvement

There are some problems with sticking to Jupyter Notebook as the final environment to run the code. As a feature, Gyro is able to map the score distribution in color by using Jupyter Notebook's plugin called Ipyleaflet. The only problem is that the plugin cannot render the data on the go. Every time you want to update a criterion weighting you need to rerun the cell that generates the map in order to see the map update, and takes a while to refresh. This is due to the fact that its underlying code is not designed to do so. The code was designed to render a map after the statistics are generated, not while they are still under calculation. A possible future improvement would be to move to a more sophisticated data visualization environment where calculations could be shown right away.

It seems Gyro is enough to solve the problem I addressed in the beginning. There are, however, many ways to improve this method. Right now, the analysis is only strictly relevant in Massachusetts. In the future, if All In Energy wants to expand to outside Massachusetts, it would require a whole new type of analysis. This is where machine learning algorithms could come in handy. By using the current data and analysis, we can build a model that will

work for a city in any state. Thinking a little bit bigger, we could scale the method to other organizations that would benefit from the algorithm. Of course, rather than pure corporate profit, the central focus would be benefiting under served communities.

Examinations

1 Computer System Examination

1.1 Preamble

Do as much as you have time for of the CMU Computer Systems shell lab. Explain what's going on in a way that shows your understanding of the material, including enough context to make sense of what you did, and with a bibliography of the resources you used. This is open take-home exam: books or web sources are OK as long as you cite them explicitly and they aren't a drop-in solution to the problem. The point is to convey what you know of the underlying ideas. Of course, working code is always nice too. ;)

1.2 Solution

1.2.1 Introduction

As a computer science student, understanding the underlying principles of how computers work is necessary. Although the knowledge might not be applicable to day-to-day problems, understanding the principles will help one approach problems in a computer science way. That is why one component of my Plan of Concentration is an exam in Computer System Class. This exam is supervised by Jim Mahoney from Bennington College as he was my computer science professor during my study back in Marlboro College period. This paper acts as a write-up to the exam which involves doing one of the labs from Computer System Class in Carnegie Mellon University. The goal of doing the lab is to facilitate me to talk about

one of the topics in the class. For this Plan of Concentration specifically, Jim Mahoney and I agreed to choose the Shell Lab as the topic for this exam.

1.2.2 Shell Lab

The Shell Lab from the class actually talks about how signalling and process works in linux environments. The goal of this lab is to create a c program that acts as a mini shell in linux terminal. The program should be able to execute a program and by creating a new process, managing foreground and background processes, and handling signals that is caused by both the processes and user inputs such as CTRL + C and CTRL + Z.

Fortunately, the lab provides a skeleton to bootstrap the process. A TAR file is available to download from the CMU website which consists of many files that will facilitate the development and testing of the C program. One of the files in the TAR file is tsh.c which has all subroutines needed to start developing the shell. After reading the hand out of how to use the files, the development started with parsing the user input. Fortunately, the skeleton file already provides a parsing function which populates an array that later will hold the command and its arguments entered by the user. The parsing function returns 1 if the user enters & at the end of the line signaling that the command should run on the background.

After parsing the user input, the logic begins. Since the topic of this lab is about signalling and processes, I jump straight away to figuring out how to manage processes. There are a couple subroutines and global variables provided where their job is to book keep processes. My job is to build a logic to decide when and where these subroutines should be run based on the user inputs. For example, when the user wants to run built-in commands such as

jobs, bg, fg, and quit, I do not have to create a new process. However, when it is not a built-in command, I need to create a new process then also register them to one of the global variables called jobs. This flow is achieved mainly by calling fork() function. Based on a book called Computer System: A Programmer's Perspective, when the fork() function is executed, the main process is called the parent process, and the new process is called the child process. The difference between them are stated as follows:

> "The newly created child process is almost, but not quite, identical to the parent. The child gets an identical (but separate) copy of the parent's user-level virtual address space, including the text, data, and bss segments, heap, and user stack. The child also gets identical copies of any of the parent's open file descriptors, which means the child can read and write any files that were open in the parent when it called fork. The most significant difference between the parent and the newly created child is that they have different PIDs." [BO11]

It is important to understand how the forking happens because confusingly, the same code will be run even in the child process. So, although the code tries to run whatever program the user wants to run, the main code will also run before the execution of the new program as the forking happens. One of the ways to make sure there is no infinite recursion happens, is to use the return value of fork() which is a PID. If the code runs a child process, the PID will always be 0.

The other main focus is signaling. The Computer System books defines signal as follows:

> "A signal is a small message that notifies a process that an event of some type

has occurred in the system." [BO11]

When a process terminates, or a user types a certain combination such as CTRL + Z or CTRL + C, a parent process will receive a signal. The parent process may catch that signal and do some processing. Since I am building a mini shell, I do not want to quit the shell program when the user wants to terminate a child. That is why I need to do processing by catching the signal. There are already empty subroutines written in the skeleton file that need to be filled with logic and bookkeeping purposes.

1.2.3 Testing

The lab expects me to complete certain tests that come with the TAR file. Although there is no completed or failed message, I need to mimic the output of the reference shell file called tshref. By executing make test01, the included will compile my tsh.c, then test the compiled tsh file. Unfortunately, due to time constraints, I could only complete test01 until test05. Below is the output of all the tests:

```
$ make test01
$ #
$ # trace01.txt - Properly terminate on EOF.
$ #
$ make test03
$ #
$ # trace03.txt - Run a foreground job.
$ #
$ tsh> quit
$ make test05
$ #
$ # trace05.txt - Process jobs builtin command.
$ #
$ tsh> ./myspin 2 &
$ tsh> ./myspin 3 &
$ tsh> jobs
$ [1] (14656) Foreground /bin/echo -e tsh> ./myspin 2 \046
$ [2] (14657) Running ./myspin 2 &
$ [3] (14658) Foreground /bin/echo -e tsh> ./myspin 3 \046
$ [4] (14659) Running ./myspin 3 &
$ [5] (14660) Foreground /bin/echo tsh> jobs
```

1.2.4 Docker

In the current era of cloud servers, when you want to deploy your application in the cloud, people use a program called Docker. The idea is to replicate the environment that the developer uses to develop the application virtually. However, not to confuse with virtualization in OS level, Docker only runs a container which has its own kernel. The shell lab reminds me of this paradigm of Docker containers because essentially Docker created a process which runs a whole new kernel on top of the host kernel. You can even have a container inside a container. This is interesting to me because the complexity of such a paradigm is out of my current experience and expertise. However, the solution Docker has to offer is simple and elegant enough in my perspective that it can accelerate the world's computers to a more distributed system.

1.2.5 Conclusion

Although I did not finish the lab completely as the hand out of the lab intended due to time constraint, the process of writing the code gives me a deeper understanding of how process and signal works in a linux environment. Part of my Plan of Concentration is deploying my code into some kind of a server to allow people to try my code. Signal and process knowledge from this chapter of Computer System Class supplements a good portion of the process.

2 Algorithm Examination

2.1 Preamble

This is open take-home exam: books or web sources are OK as long as the problem doesn't set other constraints, you cite them explicitly, and that they aren't a drop-in solution to the problem. However, the more your answer is a summary of someone else's article, the less we will be impressed. Don't ask other people for help. Don't just give a numerical result, give an explanation. Your job is to convince us you understand this stuff. So in terms of a grading rubric, what you should think about is

* technical merit - correctness & thoroughness of response

* clarity of expression (including docs & tests for code)

* demonstration of understanding (vs summarizing other's work)

As always with my exams, if you think there's a mistake in one of the questions or it doesn't make sense, you can (a) ask for clarification, and/or (b) make and state an explicit interpretation and do the problem that way. (Again: the point is to show your mastery, not to get the "right answer" per se.)

Good luck.

2.2 Questions

2.2.1 Question 1.

Implement, numerically test, and explain two different sorting algorithms with different O() behaviors on randomly chosen lists of numbers with various sizes n. Use two different programming languages and coding styles. Show graphically that the expecected performance is consistent with your numerical experiments.

Solution:

Every time a computer does an operation, it needs to pay in some kind of form. The form can be steps or room to put data temporarily. The price of this operation can be expressed in a mathematical way and it is called complexity. Jim asked me to implement 2 different sorting algorithms with 2 different O() written in 2 different programming languages and 2 different styles of coding. The way I approached this by first, choosing 2 different algorithms. When talking about complexity, there are 2 main things to consider. It is a choice between speed and space. So, I chose one that is fast but takes a lot of space, and also another one which is slower but takes almost no space. They are selection sort and counting sort.

Selection Sort

Since the problem requires me to write the algorithms in two different programming lan-

guages, I started this relatively simple sorting algorithm with JavaScript. The principle behind selection sort is that to find the smallest number each time you check all numbers starting from the first number in the sequence. In theory, selection sort should have an $O(1)$ in space complexity. That means selection sort can perform an in-place operation to the number sequence without needing another space in memory. However, in my implementation instead of doing an in-place modification to the original array, I made a copy of them which makes the implementation have $O(n)$ space complexity. The algorithm comes with a caveat. As mentioned before, when doing a sort, there are two things to consider, space or speed. Due to the nature of the algorithm, selection sort has $O(n^2)$. This can also be inspect on the code itself, below is a snippet of my selection sort implementation in selection_sort.js file:

```
14  for (let i = 0; i < arrLength; i++) {
15    let currentLowest = i;
16    steps++;
17    for (let j = i; j < arrLength; j++) {
18      steps++;
19      if (result[currentLowest] > result[j]) {
20        currentLowest = j
21      }
22    }
23    const swap = result[i];
24    result[i] = result[currentLowest]
25    result[currentLowest] = swap
26  }
```

There is a for loop inside a for loop which makes the algorithm need to do a lot more of steps as the sequence gets longer.

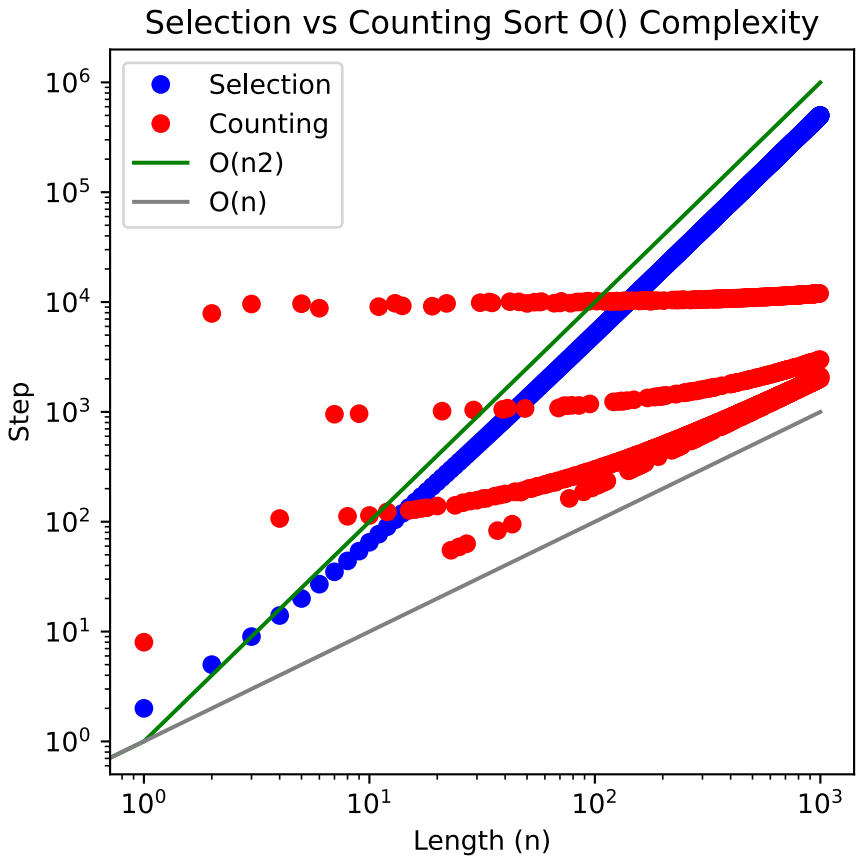
Counting Sort

Different from selection sort, counting sort does not rely on comparing a number with another

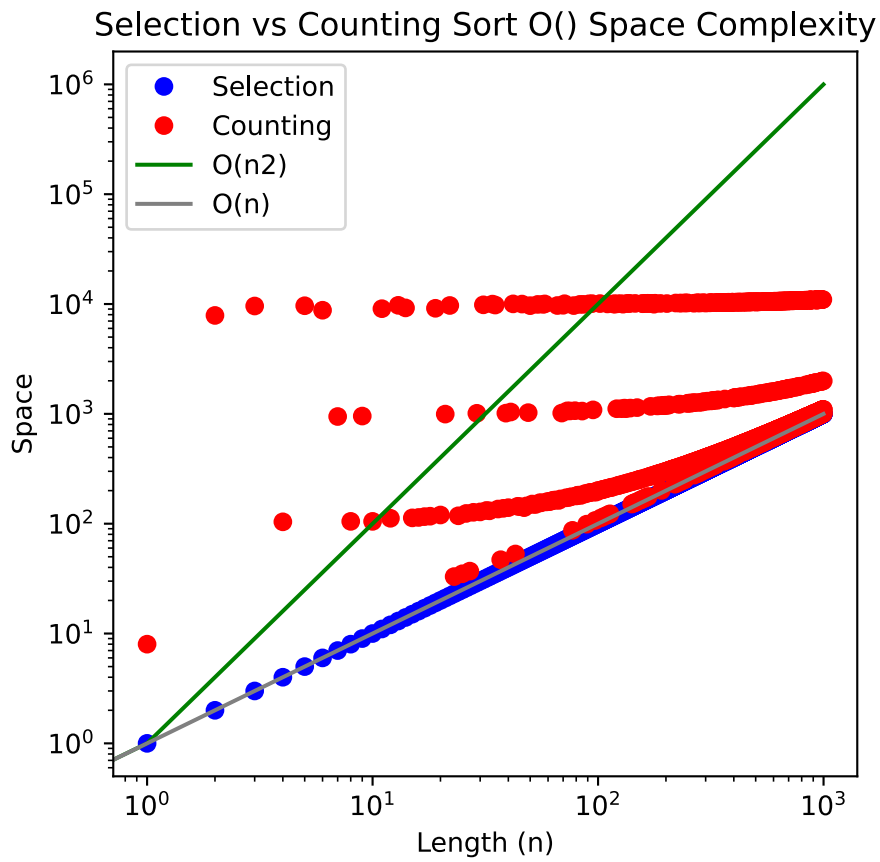
element in order to sort the sequence of numbers. Instead, the algorithm uses an indexing method. This indexing method also comes with a price. The space complexity for counting sort is $O(n \cdot k)$ where k is the range of numbers in a sequence. For example, if a number sequence is 1,10,2,3,4,5, the algorithm will need 60 units of space to store all numbers between 0 and 10. However, the speed of this algorithm is really fast relative to the selection sort. In theory it should have $O(n \cdot k)$ too in terms of speed. I implemented this algorithm using class inheritance method in Python by extending the built-in List variable. This will allow me to perform any built-in List method while also adding my own sorting algorithm.

Testing

In order to test how both implementations work, I set up a test. First, I generated a JSON file called input.json which contains a list of lists of sequences of numbers with different length and random range. I chose JSON format because it is easy to parse for both language Python and JavaScript. Then, I load the numbers and run the algorithm to sort all the lists of numbers. Here are the results:



The graph shows both selection and counting performance. The counting sort is more superior than the selection sort since counting's complexity is O(n k), where the k is the range of numbers in the list. Specifically when sorting a really long sequence of numbers, like 100000 numbers, the counting sort will have less complexity than selection sort. However it comes with caveats.



This graph shows the caveats. Although counting sort takes less steps to sort the list, it takes more space than selection sort. Specifically, when the range of the numbers in the sequence is large.

2.2.2 Question 2.

In a language of your choice, code and explain both a depth first and breadth first search algorithm of a tree of possible moves for a small "sliding block" puzzle. (Using a stack and queue would be a good approach.) A sample search might be to get from:

start finish

2 1 3 1 2 3

5 4 6 4 5 6

7 8 . 7 8 .

where the "." is the empty square and the two possible first moves slide either the 6 or the 8 to bottom right corner.

Solution:

Graph is a term used to describe a relationship. Specifically in computer science, graphs are used as a representation of connections between variables or data.

> "More precisely, a graph $G = (V, E)$ consists of a set of vertices V together with a set E of vertex pairs or edges. Graphs are important because they can be used to represent essentially any relationship. For example, graphs can model a network of roads, with cities as vertices and roads between cities as edges, as shown in Figure 5.1. Electronic circuits can also be modeled as graphs, with junctions as vertices and components as edges." [Ski08]

As mentioned in the book The Algorithm Design Manual by Steven S. Skiena, understanding how graphs work, will add another simple yet powerful tool to solve the problem. One of the problems is to find a solution where there are many possible combinations. For Sliding block problem as the question stated, I need to find a solution to transform any particular state into another state using a tree. A tree is used because there are possibilities that a move will result in a loop that never ends. So, the tree structure is needed.

Board Class

In order to solve the problem I need to create a structure where the search can perform. Therefore, I created the Board class. The Board class functions as the backbone of the game. I used class here so that I can make a lot of board instances which store their own data. This way, no matter what kind of algorithm that will be applied, the game can go on seamlessly.

State Class

The State Class acts as a tree node in the graph that I am trying to search. Each node will store all moves that happen in any particular branch. This way, I can list all the moves that a specific branch has and visualize the moves.

Searches Class

After going through how the game works, I need to do a search down the trees. There are two common ways to search a tree, breadth or depth first. Breadth first search is done by visiting all children of a node first. This means in my case of moving a block my search algorithm needs to visit all possible moves at any state of the game. On the contrary, depth-first search is done by picking a move after another while looking at the other possible move until

the desired state or the end of a branch is reached before looking at other possible moves.

Searches class holds all the search algorithms needs. I implemented memoization to make the search more efficient because the search can be stuck in a loop pretty easily. In order to implement depth and depth search algorithms, I am using queue and stack data structure. Queue is a list where everything comes in first will come out first. On the other hand, stack is a list where everything comes last will come out first.

After doing the search, I found a couple of problems, the breadth search method takes so much memory that my laptop could not handle, so I tweaked it a little bit by limiting the search loop to 100 thousand iterations. However, despite the limitation, breadth search first approach does not complete the search. This problem occurs because in my case of the game, the solution is located deep in one of the tree branches. This is proven by how the depth search algorithm can give the result in 16 moves, compared to breadth first search with 93 thousand moves.

2.2.3 Question 3.

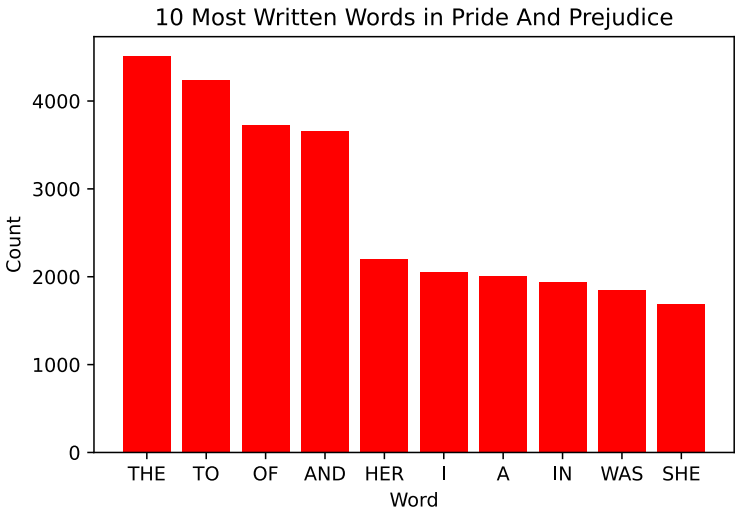
Explain what a ”hash table” is, and what it’s O() behavior looks like. Implement one, use it to make a histogram of a word counts in a large text file, and discuss its performance. The specifics (collision algorithm, hash function, programming language) are all up to you.

Solution:

Hash table is a data structure that uses a hash function to determine index for every element. This is an advantage in terms of retrieving information because instead of doing a linear

search on each element. The search complexity of a hash table is $O(1)$. However, in the implementation, we do not have an infinite amount of index. That means although we have a unique hash value for everything, we do not necessarily have the index to translate the hash into an index in array. This is where collision algorithms are needed. The way I implemented the collision algorithm is by expanding the array of the data. If the array is 20% full, an expanding subroutine will allocate 10x more space in the array and re-index every entry.

In order to test the implementation, I downloaded a book called Pride and Prejudice by Jane Austen from Project Gutenberg. First, I tested how fast the algorithm is compared to the built-in Python dictionary. By setting up a timer, the result is as expected. Due to my implementation relies on expansion in order to avoid collision, it is rather slow. It took 2.5 seconds to index the words compared to the built-in dictionary, 0.05 seconds. This is due to the fact that every time an expansion is executed, every entry needs to be re-indexed which takes quite a while. Also, there is a bug that I could not fix where during the expansion some of the keys are missing. This is a bad implementation if you are looking for a reliable hash table. However, below is the histogram of the words in Pride And Prejudice.



2.2.4 Question 4.

Explain in your own words what exactly is meant by ”P” and ”NP” as complexity classes in computer science, why this is such an important question, and what is and isn't known about them. Give explicit examples of problems that are in each of these classes, and explain why the known algorithms have behaviors consistent with your P and NP descriptions. You may use external sources if you need to - and if so be clear which ones you used, of course - but the point here is to convey to us your understanding, not to just summarize a wikipedia article.

Solution:

At the time as this paper is written, there is a big question in the computer science field about the limit of classical computer's ability to crunch numbers. The question is so famous because it is part of a famous competition where the winner will receive $1 million, Millennium

Prize Problems. As the prize shows, the question is not an ordinary question. The question whether P equals NP is a question of fundamental mechanics of human understanding about algorithms.

> "The primary question in P vs NP is whether verification is really an easier task than initial discovery. Suppose that while taking an exam you "happen" to notice the answer of the student next to you. Are you now better off? You wouldn't dare to turn it in without checking, since an able student such as yourself could answer the question correctly if you took enough time to solve it. The issue is whether you can really verify the answer faster than you could find it from scratch." [Ski08]

As stated in the book The Algorithm Design Manual by Steven S. Skiena above, P and NP is about proofing we can solve problems in polynomial time if we can check the result in polynomial time. For example, the sliding block problem from question Graph Search. If we want to check whether the puzzle is solved correctly, it is simply a sorting problem. So, the worst case, simple algorithm such as selection sort will solve the problem in $O(n^2)$. However, if we change the grid into bigger numbers, it gets harder to find a path to solve them. A brute force method would require a $O(n!)$ to find all possible states in a given grid. So, the sliding problem from Graph Search is a NP problem. Whereas, sorting problem is a P problem because solving it can be as fast as verifying it. Just like mentioned above, sorting will have $O(n^2)$ to solve it, and $O(n^2)$ too in verifying it.

The part that is interesting to me is what the impact this question has to everyday

life. Since the question is talking about the measure of algorithms solving problems, it can impact how we perceive a problem. If NP is equal to P, it means there is always going to be a fast enough algorithm to solve every problem. Whether predicting the weather, finding the fastest routes, or winning a chess game, there will be an algorithm that solves the problem fast enough without having to wait until the heat death of the universe. Algorithms also apply to not only computer programs, but also human's way to solve problems. That means there would not be hard enough problems that humans can not solve fast enough. What is the point of life then? We are humans, we solve problems, we are curious to observe our surroundings. We need problems. We need to do things that we never know what the outcome will be. So, even though it will be so cool to prove P is equal to NP and get $1 million, I personally still wish it will be proved to be false.

Bibliography

[BO11] Randal E. Bryant and David Richard. OHallaron. *Computer systems: a programmers perspective*. Pearson, 2011.

[Ski08] Steven S. Skiena. *The Algorithm Design Manual*. Springer London, 2008.

Appendix: Gyro: Data-driven Nonprofit Expansion Using Assessor Database

1 Gyro

This is the deliverable Jupyter Notebook that later will be used by All In Energy to expand its operation. Live Jupyter Notebook is available here. The password is **linustorvalds**. Jupyter Notebooks is needed to be re run by clicking the fast-forward button next to a drop down title markdown. Please kindly wait until all cells are executed indicated by the website icon on the top turned from a hourglass icon into a book icon.

Gyro

November 26, 2020

1 GYRO

Finding the best city based on cities' data using weighted criteria.

1.1 A. Data

After doing analysis on cities' data in analysis notebook, I want to address its core problem with the assumption that the criteria is equally important. I am going to use the exact data that I aggregated from cleaning notebook.

[1]:
```
%pip install geopandas
%pip install ipyleaflet
%pip install -U matplotlib
%pip install jupyter_contrib_nbextensions
!jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
Requirement already satisfied: geopandas in /opt/conda/lib/python3.8/site-
packages (0.8.1)
Requirement already satisfied: pandas>=0.23.0 in /opt/conda/lib/python3.8/site-
packages (from geopandas) (1.1.1)
Requirement already satisfied: pyproj>=2.2.0 in /opt/conda/lib/python3.8/site-
packages (from geopandas) (3.0.0.post1)
Requirement already satisfied: shapely in /opt/conda/lib/python3.8/site-packages
(from geopandas) (1.7.1)
Requirement already satisfied: fiona in /opt/conda/lib/python3.8/site-packages
(from geopandas) (1.8.18)
Requirement already satisfied: numpy>=1.15.4 in /opt/conda/lib/python3.8/site-
packages (from pandas>=0.23.0->geopandas) (1.19.1)
Requirement already satisfied: python-dateutil>=2.7.3 in
/opt/conda/lib/python3.8/site-packages (from pandas>=0.23.0->geopandas) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in /opt/conda/lib/python3.8/site-
packages (from pandas>=0.23.0->geopandas) (2020.1)
Requirement already satisfied: certifi in /opt/conda/lib/python3.8/site-packages
(from pyproj>=2.2.0->geopandas) (2020.6.20)
Requirement already satisfied: munch in /opt/conda/lib/python3.8/site-packages
(from fiona->geopandas) (2.5.0)
Requirement already satisfied: click-plugins>=1.0 in
/opt/conda/lib/python3.8/site-packages (from fiona->geopandas) (1.1.1)
Requirement already satisfied: click<8,>=4.0 in /opt/conda/lib/python3.8/site-
```

```
packages (from fiona->geopandas) (7.1.2)
Requirement already satisfied: cligj>=0.5 in /opt/conda/lib/python3.8/site-
packages (from fiona->geopandas) (0.7.1)
Requirement already satisfied: attrs>=17 in /opt/conda/lib/python3.8/site-
packages (from fiona->geopandas) (20.1.0)
Requirement already satisfied: six>=1.7 in /opt/conda/lib/python3.8/site-
packages (from fiona->geopandas) (1.15.0)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: ipyleaflet in /opt/conda/lib/python3.8/site-
packages (0.13.3)
Requirement already satisfied: branca<0.5,>=0.3.1 in
/opt/conda/lib/python3.8/site-packages (from ipyleaflet) (0.4.1)
Requirement already satisfied: ipywidgets<8,>=7.5.0 in
/opt/conda/lib/python3.8/site-packages (from ipyleaflet) (7.5.1)
Requirement already satisfied: traittypes<3,>=0.2.1 in
/opt/conda/lib/python3.8/site-packages (from ipyleaflet) (0.2.1)
Requirement already satisfied: jinja2 in /opt/conda/lib/python3.8/site-packages
(from branca<0.5,>=0.3.1->ipyleaflet) (2.11.2)
Requirement already satisfied: ipykernel>=4.5.1 in
/opt/conda/lib/python3.8/site-packages (from ipywidgets<8,>=7.5.0->ipyleaflet)
(5.3.4)
Requirement already satisfied: ipython>=4.0.0; python_version >= "3.3" in
/opt/conda/lib/python3.8/site-packages (from ipywidgets<8,>=7.5.0->ipyleaflet)
(7.17.0)
Requirement already satisfied: nbformat>=4.2.0 in /opt/conda/lib/python3.8/site-
packages (from ipywidgets<8,>=7.5.0->ipyleaflet) (5.0.7)
Requirement already satisfied: traitlets>=4.3.1 in
/opt/conda/lib/python3.8/site-packages (from ipywidgets<8,>=7.5.0->ipyleaflet)
(4.3.3)
Requirement already satisfied: widgetsnbextension~=3.5.0 in
/opt/conda/lib/python3.8/site-packages (from ipywidgets<8,>=7.5.0->ipyleaflet)
(3.5.1)
Requirement already satisfied: MarkupSafe>=0.23 in
/opt/conda/lib/python3.8/site-packages (from
jinja2->branca<0.5,>=0.3.1->ipyleaflet) (1.1.1)
Requirement already satisfied: jupyter-client in /opt/conda/lib/python3.8/site-
packages (from ipykernel>=4.5.1->ipywidgets<8,>=7.5.0->ipyleaflet) (6.1.6)
Requirement already satisfied: tornado>=4.2 in /opt/conda/lib/python3.8/site-
packages (from ipykernel>=4.5.1->ipywidgets<8,>=7.5.0->ipyleaflet) (6.0.4)
Requirement already satisfied: backcall in /opt/conda/lib/python3.8/site-
packages (from ipython>=4.0.0; python_version >=
"3.3"->ipywidgets<8,>=7.5.0->ipyleaflet) (0.2.0)
Requirement already satisfied: decorator in /opt/conda/lib/python3.8/site-
packages (from ipython>=4.0.0; python_version >=
"3.3"->ipywidgets<8,>=7.5.0->ipyleaflet) (4.4.2)
Requirement already satisfied: pygments in /opt/conda/lib/python3.8/site-
packages (from ipython>=4.0.0; python_version >=
"3.3"->ipywidgets<8,>=7.5.0->ipyleaflet) (2.6.1)
```

```
Requirement already satisfied: pexpect; sys_platform != "win32" in
/opt/conda/lib/python3.8/site-packages (from ipython>=4.0.0; python_version >=
"3.3"->ipywidgets<8,>=7.5.0->ipyleaflet) (4.8.0)
Requirement already satisfied: pickleshare in /opt/conda/lib/python3.8/site-
packages (from ipython>=4.0.0; python_version >=
"3.3"->ipywidgets<8,>=7.5.0->ipyleaflet) (0.7.5)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in
/opt/conda/lib/python3.8/site-packages (from ipython>=4.0.0; python_version >=
"3.3"->ipywidgets<8,>=7.5.0->ipyleaflet) (3.0.6)
Requirement already satisfied: jedi>=0.10 in /opt/conda/lib/python3.8/site-
packages (from ipython>=4.0.0; python_version >=
"3.3"->ipywidgets<8,>=7.5.0->ipyleaflet) (0.17.2)
Requirement already satisfied: setuptools>=18.5 in
/opt/conda/lib/python3.8/site-packages (from ipython>=4.0.0; python_version >=
"3.3"->ipywidgets<8,>=7.5.0->ipyleaflet) (49.6.0.post20200814)
Requirement already satisfied: jupyter-core in /opt/conda/lib/python3.8/site-
packages (from nbformat>=4.2.0->ipywidgets<8,>=7.5.0->ipyleaflet) (4.6.3)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in
/opt/conda/lib/python3.8/site-packages (from
nbformat>=4.2.0->ipywidgets<8,>=7.5.0->ipyleaflet) (3.2.0)
Requirement already satisfied: ipython-genutils in
/opt/conda/lib/python3.8/site-packages (from
nbformat>=4.2.0->ipywidgets<8,>=7.5.0->ipyleaflet) (0.2.0)
Requirement already satisfied: six in /opt/conda/lib/python3.8/site-packages
(from traitlets>=4.3.1->ipywidgets<8,>=7.5.0->ipyleaflet) (1.15.0)
Requirement already satisfied: notebook>=4.4.1 in /opt/conda/lib/python3.8/site-
packages (from widgetsnbextension~=3.5.0->ipywidgets<8,>=7.5.0->ipyleaflet)
(6.1.3)
Requirement already satisfied: pyzmq>=13 in /opt/conda/lib/python3.8/site-
packages (from jupyter-
client->ipykernel>=4.5.1->ipywidgets<8,>=7.5.0->ipyleaflet) (19.0.2)
Requirement already satisfied: python-dateutil>=2.1 in
/opt/conda/lib/python3.8/site-packages (from jupyter-
client->ipykernel>=4.5.1->ipywidgets<8,>=7.5.0->ipyleaflet) (2.8.1)
Requirement already satisfied: ptyprocess>=0.5 in /opt/conda/lib/python3.8/site-
packages (from pexpect; sys_platform != "win32"->ipython>=4.0.0; python_version
>= "3.3"->ipywidgets<8,>=7.5.0->ipyleaflet) (0.6.0)
Requirement already satisfied: wcwidth in /opt/conda/lib/python3.8/site-packages
(from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->ipython>=4.0.0;
python_version >= "3.3"->ipywidgets<8,>=7.5.0->ipyleaflet) (0.2.5)
Requirement already satisfied: parso<0.8.0,>=0.7.0 in
/opt/conda/lib/python3.8/site-packages (from jedi>=0.10->ipython>=4.0.0;
python_version >= "3.3"->ipywidgets<8,>=7.5.0->ipyleaflet) (0.7.1)
Requirement already satisfied: attrs>=17.4.0 in /opt/conda/lib/python3.8/site-
packages (from
jsonschema!=2.5.0,>=2.4->nbformat>=4.2.0->ipywidgets<8,>=7.5.0->ipyleaflet)
(20.1.0)
Requirement already satisfied: pyrsistent>=0.14.0 in
```

```
/opt/conda/lib/python3.8/site-packages (from
jsonschema!=2.5.0,>=2.4->nbformat>=4.2.0->ipywidgets<8,>=7.5.0->ipyleaflet)
(0.16.0)
Requirement already satisfied: terminado>=0.8.3 in
/opt/conda/lib/python3.8/site-packages (from
notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets<8,>=7.5.0->ipyleaflet)
(0.8.3)
Requirement already satisfied: argon2-cffi in /opt/conda/lib/python3.8/site-
packages (from
notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets<8,>=7.5.0->ipyleaflet)
(20.1.0)
Requirement already satisfied: prometheus-client in
/opt/conda/lib/python3.8/site-packages (from
notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets<8,>=7.5.0->ipyleaflet)
(0.8.0)
Requirement already satisfied: nbconvert in /opt/conda/lib/python3.8/site-
packages (from
notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets<8,>=7.5.0->ipyleaflet)
(5.6.1)
Requirement already satisfied: Send2Trash in /opt/conda/lib/python3.8/site-
packages (from
notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets<8,>=7.5.0->ipyleaflet)
(1.5.0)
Requirement already satisfied: cffi>=1.0.0 in /opt/conda/lib/python3.8/site-
packages (from argon2-cffi->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidge
ts<8,>=7.5.0->ipyleaflet) (1.14.1)
Requirement already satisfied: defusedxml in /opt/conda/lib/python3.8/site-
packages (from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets
<8,>=7.5.0->ipyleaflet) (0.6.0)
Requirement already satisfied: pandocfilters>=1.4.1 in
/opt/conda/lib/python3.8/site-packages (from nbconvert->notebook>=4.4.1->widgets
nbextension~=3.5.0->ipywidgets<8,>=7.5.0->ipyleaflet) (1.4.2)
Requirement already satisfied: mistune<2,>=0.8.1 in
/opt/conda/lib/python3.8/site-packages (from nbconvert->notebook>=4.4.1->widgets
nbextension~=3.5.0->ipywidgets<8,>=7.5.0->ipyleaflet) (0.8.4)
Requirement already satisfied: entrypoints>=0.2.2 in
/opt/conda/lib/python3.8/site-packages (from nbconvert->notebook>=4.4.1->widgets
nbextension~=3.5.0->ipywidgets<8,>=7.5.0->ipyleaflet) (0.3)
Requirement already satisfied: bleach in /opt/conda/lib/python3.8/site-packages
(from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets<8,>=7.5.
0->ipyleaflet) (3.1.5)
Requirement already satisfied: testpath in /opt/conda/lib/python3.8/site-
packages (from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets
<8,>=7.5.0->ipyleaflet) (0.4.4)
Requirement already satisfied: pycparser in /opt/conda/lib/python3.8/site-
packages (from cffi>=1.0.0->argon2-cffi->notebook>=4.4.1->widgetsnbextension~=3.
5.0->ipywidgets<8,>=7.5.0->ipyleaflet) (2.20)
Requirement already satisfied: webencodings in /opt/conda/lib/python3.8/site-
```

```
packages (from bleach->nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ip
ywidgets<8,>=7.5.0->ipyleaflet) (0.5.1)
Requirement already satisfied: packaging in /opt/conda/lib/python3.8/site-
packages (from bleach->nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ip
ywidgets<8,>=7.5.0->ipyleaflet) (20.4)
Requirement already satisfied: pyparsing>=2.0.2 in
/opt/conda/lib/python3.8/site-packages (from packaging->bleach->nbconvert->noteb
ook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets<8,>=7.5.0->ipyleaflet) (2.4.7)
Note: you may need to restart the kernel to use updated packages.
Requirement already up-to-date: matplotlib in /opt/conda/lib/python3.8/site-
packages (3.3.3)
Requirement already satisfied, skipping upgrade: numpy>=1.15 in
/opt/conda/lib/python3.8/site-packages (from matplotlib) (1.19.1)
Requirement already satisfied, skipping upgrade: pillow>=6.2.0 in
/opt/conda/lib/python3.8/site-packages (from matplotlib) (7.2.0)
Requirement already satisfied, skipping upgrade: kiwisolver>=1.0.1 in
/opt/conda/lib/python3.8/site-packages (from matplotlib) (1.2.0)
Requirement already satisfied, skipping upgrade: cycler>=0.10 in
/opt/conda/lib/python3.8/site-packages (from matplotlib) (0.10.0)
Requirement already satisfied, skipping upgrade:
pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in /opt/conda/lib/python3.8/site-
packages (from matplotlib) (2.4.7)
Requirement already satisfied, skipping upgrade: python-dateutil>=2.1 in
/opt/conda/lib/python3.8/site-packages (from matplotlib) (2.8.1)
Requirement already satisfied, skipping upgrade: six in
/opt/conda/lib/python3.8/site-packages (from cycler>=0.10->matplotlib) (1.15.0)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: jupyter_contrib_nbextensions in
/opt/conda/lib/python3.8/site-packages (0.5.1)
Requirement already satisfied: jupyter-highlight-selected-word>=0.1.1 in
/opt/conda/lib/python3.8/site-packages (from jupyter_contrib_nbextensions)
(0.2.0)
Requirement already satisfied: pyyaml in /opt/conda/lib/python3.8/site-packages
(from jupyter_contrib_nbextensions) (5.3.1)
Requirement already satisfied: tornado in /opt/conda/lib/python3.8/site-packages
(from jupyter_contrib_nbextensions) (6.0.4)
Requirement already satisfied: nbconvert>=4.2 in /opt/conda/lib/python3.8/site-
packages (from jupyter_contrib_nbextensions) (5.6.1)
Requirement already satisfied: ipython-genutils in
/opt/conda/lib/python3.8/site-packages (from jupyter_contrib_nbextensions)
(0.2.0)
Requirement already satisfied: notebook>=4.0 in /opt/conda/lib/python3.8/site-
packages (from jupyter_contrib_nbextensions) (6.1.3)
Requirement already satisfied: jupyter-contrib-core>=0.3.3 in
/opt/conda/lib/python3.8/site-packages (from jupyter_contrib_nbextensions)
(0.3.3)
Requirement already satisfied: lxml in /opt/conda/lib/python3.8/site-packages
(from jupyter_contrib_nbextensions) (4.6.1)
```

```
Requirement already satisfied: jupyter-core in /opt/conda/lib/python3.8/site-
packages (from jupyter_contrib_nbextensions) (4.6.3)
Requirement already satisfied: jupyter-latex-envs>=1.3.8 in
/opt/conda/lib/python3.8/site-packages (from jupyter_contrib_nbextensions)
(1.4.6)
Requirement already satisfied: jupyter-nbextensions-configurator>=0.4.0 in
/opt/conda/lib/python3.8/site-packages (from jupyter_contrib_nbextensions)
(0.4.1)
Requirement already satisfied: traitlets>=4.1 in /opt/conda/lib/python3.8/site-
packages (from jupyter_contrib_nbextensions) (4.3.3)
Requirement already satisfied: nbformat>=4.4 in /opt/conda/lib/python3.8/site-
packages (from nbconvert>=4.2->jupyter_contrib_nbextensions) (5.0.7)
Requirement already satisfied: entrypoints>=0.2.2 in
/opt/conda/lib/python3.8/site-packages (from
nbconvert>=4.2->jupyter_contrib_nbextensions) (0.3)
Requirement already satisfied: testpath in /opt/conda/lib/python3.8/site-
packages (from nbconvert>=4.2->jupyter_contrib_nbextensions) (0.4.4)
Requirement already satisfied: pygments in /opt/conda/lib/python3.8/site-
packages (from nbconvert>=4.2->jupyter_contrib_nbextensions) (2.6.1)
Requirement already satisfied: bleach in /opt/conda/lib/python3.8/site-packages
(from nbconvert>=4.2->jupyter_contrib_nbextensions) (3.1.5)
Requirement already satisfied: defusedxml in /opt/conda/lib/python3.8/site-
packages (from nbconvert>=4.2->jupyter_contrib_nbextensions) (0.6.0)
Requirement already satisfied: jinja2>=2.4 in /opt/conda/lib/python3.8/site-
packages (from nbconvert>=4.2->jupyter_contrib_nbextensions) (2.11.2)
Requirement already satisfied: mistune<2,>=0.8.1 in
/opt/conda/lib/python3.8/site-packages (from
nbconvert>=4.2->jupyter_contrib_nbextensions) (0.8.4)
Requirement already satisfied: pandocfilters>=1.4.1 in
/opt/conda/lib/python3.8/site-packages (from
nbconvert>=4.2->jupyter_contrib_nbextensions) (1.4.2)
Requirement already satisfied: Send2Trash in /opt/conda/lib/python3.8/site-
packages (from notebook>=4.0->jupyter_contrib_nbextensions) (1.5.0)
Requirement already satisfied: argon2-cffi in /opt/conda/lib/python3.8/site-
packages (from notebook>=4.0->jupyter_contrib_nbextensions) (20.1.0)
Requirement already satisfied: terminado>=0.8.3 in
/opt/conda/lib/python3.8/site-packages (from
notebook>=4.0->jupyter_contrib_nbextensions) (0.8.3)
Requirement already satisfied: ipykernel in /opt/conda/lib/python3.8/site-
packages (from notebook>=4.0->jupyter_contrib_nbextensions) (5.3.4)
Requirement already satisfied: pyzmq>=17 in /opt/conda/lib/python3.8/site-
packages (from notebook>=4.0->jupyter_contrib_nbextensions) (19.0.2)
Requirement already satisfied: jupyter-client>=5.3.4 in
/opt/conda/lib/python3.8/site-packages (from
notebook>=4.0->jupyter_contrib_nbextensions) (6.1.6)
Requirement already satisfied: prometheus-client in
/opt/conda/lib/python3.8/site-packages (from
notebook>=4.0->jupyter_contrib_nbextensions) (0.8.0)
```

```
Requirement already satisfied: setuptools in /opt/conda/lib/python3.8/site-
packages (from jupyter-contrib-core>=0.3.3->jupyter_contrib_nbextensions)
(49.6.0.post20200814)
Requirement already satisfied: ipython in /opt/conda/lib/python3.8/site-packages
(from jupyter-latex-envs>=1.3.8->jupyter_contrib_nbextensions) (7.17.0)
Requirement already satisfied: six in /opt/conda/lib/python3.8/site-packages
(from traitlets>=4.1->jupyter_contrib_nbextensions) (1.15.0)
Requirement already satisfied: decorator in /opt/conda/lib/python3.8/site-
packages (from traitlets>=4.1->jupyter_contrib_nbextensions) (4.4.2)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in
/opt/conda/lib/python3.8/site-packages (from
nbformat>=4.4->nbconvert>=4.2->jupyter_contrib_nbextensions) (3.2.0)
Requirement already satisfied: packaging in /opt/conda/lib/python3.8/site-
packages (from bleach->nbconvert>=4.2->jupyter_contrib_nbextensions) (20.4)
Requirement already satisfied: webencodings in /opt/conda/lib/python3.8/site-
packages (from bleach->nbconvert>=4.2->jupyter_contrib_nbextensions) (0.5.1)
Requirement already satisfied: MarkupSafe>=0.23 in
/opt/conda/lib/python3.8/site-packages (from
jinja2>=2.4->nbconvert>=4.2->jupyter_contrib_nbextensions) (1.1.1)
Requirement already satisfied: cffi>=1.0.0 in /opt/conda/lib/python3.8/site-
packages (from argon2-cffi->notebook>=4.0->jupyter_contrib_nbextensions)
(1.14.1)
Requirement already satisfied: python-dateutil>=2.1 in
/opt/conda/lib/python3.8/site-packages (from jupyter-
client>=5.3.4->notebook>=4.0->jupyter_contrib_nbextensions) (2.8.1)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in
/opt/conda/lib/python3.8/site-packages (from ipython->jupyter-latex-
envs>=1.3.8->jupyter_contrib_nbextensions) (3.0.6)
Requirement already satisfied: jedi>=0.10 in /opt/conda/lib/python3.8/site-
packages (from ipython->jupyter-latex-envs>=1.3.8->jupyter_contrib_nbextensions)
(0.17.2)
Requirement already satisfied: pickleshare in /opt/conda/lib/python3.8/site-
packages (from ipython->jupyter-latex-envs>=1.3.8->jupyter_contrib_nbextensions)
(0.7.5)
Requirement already satisfied: backcall in /opt/conda/lib/python3.8/site-
packages (from ipython->jupyter-latex-envs>=1.3.8->jupyter_contrib_nbextensions)
(0.2.0)
Requirement already satisfied: pexpect; sys_platform != "win32" in
/opt/conda/lib/python3.8/site-packages (from ipython->jupyter-latex-
envs>=1.3.8->jupyter_contrib_nbextensions) (4.8.0)
Requirement already satisfied: attrs>=17.4.0 in /opt/conda/lib/python3.8/site-
packages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert>=4.2->jupyter_c
ontrib_nbextensions) (20.1.0)
Requirement already satisfied: pyrsistent>=0.14.0 in
/opt/conda/lib/python3.8/site-packages (from jsonschema!=2.5.0,>=2.4->nbformat>=
4.4->nbconvert>=4.2->jupyter_contrib_nbextensions) (0.16.0)
Requirement already satisfied: pyparsing>=2.0.2 in
/opt/conda/lib/python3.8/site-packages (from
```

```
packaging->bleach->nbconvert>=4.2->jupyter_contrib_nbextensions) (2.4.7)
Requirement already satisfied: pycparser in /opt/conda/lib/python3.8/site-
packages (from
cffi>=1.0.0->argon2-cffi->notebook>=4.0->jupyter_contrib_nbextensions) (2.20)
Requirement already satisfied: wcwidth in /opt/conda/lib/python3.8/site-packages
(from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->ipython->jupyter-latex-
envs>=1.3.8->jupyter_contrib_nbextensions) (0.2.5)
Requirement already satisfied: parso<0.8.0,>=0.7.0 in
/opt/conda/lib/python3.8/site-packages (from jedi>=0.10->ipython->jupyter-latex-
envs>=1.3.8->jupyter_contrib_nbextensions) (0.7.1)
Requirement already satisfied: ptyprocess>=0.5 in /opt/conda/lib/python3.8/site-
packages (from pexpect; sys_platform != "win32"->ipython->jupyter-latex-
envs>=1.3.8->jupyter_contrib_nbextensions) (0.6.0)
Note: you may need to restart the kernel to use updated packages.
Enabling notebook extension jupyter-leaflet/extension…
      - Validating: OK
```

[2]:
```
import pandas as pd
data = pd.read_csv('MA Cities Data Oct 10.csv')
data.sample(5)
```

[2]:

	CITY	1-Family	2-Family	3-Family	4-Family	5+ Family	OWNER	\
100	FREETOWN	1893.0	1048.0	NaN	NaN	3.0	2201.0	
107	GRAFTON	1988.0	3162.0	41.0	NaN	30.0	4355.0	
134	HOPKINTON	1925.0	2491.0	NaN	13.0	NaN	3037.0	
190	NANTUCKET	4614.0	2867.0	NaN	17.0	3.0	1013.0	
306	WATERTOWN	1044.0	1310.0	364.0	95.0	56.0	2137.0	

	RENTER	SHAPE_AREA	distance	…	Gas	<20K	20K-39K	40K-59K	\
100	743.0	9.185947e+07	14.995952	…	True	8.03	12.22	9.84	
107	865.0	6.035575e+07	40.633570	…	True	7.70	11.88	12.85	
134	1392.0	7.213707e+07	29.725743	…	True	4.67	3.52	8.90	
190	6415.0	1.266795e+08	75.085279	…	False	7.63	10.80	10.83	
306	731.0	1.068145e+07	5.473547	…	True	9.60	8.16	9.85	

	60K-74K	75K-99K	>100K	Spanish Speaker	Black	Latino or Hispanic
100	11.13	14.20	NaN	0.43	0.49	2.31
107	5.94	10.23	51.39	4.78	4.46	6.69
134	5.26	7.68	NaN	1.43	1.73	2.63
190	9.65	7.60	NaN	4.17	5.13	4.55
306	9.21	13.74	49.44	6.04	1.86	9.68

[5 rows x 24 columns]

First, I need to transform the data into the same form as I drew conclusion in analysis notebook. I am going to redo all the operation in each criteria.

1.1.1 1. The closest city from Boston

[3]:
```
distance = data[['CITY', 'distance']].copy()
most_distance = distance.sort_values('distance').rename(columns={'distance':
 ↪'Distance'})
```

1.1.2 2. The City with The Most of Family Buildings

[4]:
```
buildings = data[['CITY', '1-Family', '2-Family', '3-Family', '4-Family', '5+␣
 ↪Family']].copy()
buildings['1-4 Family'] = buildings.drop(columns=['5+ Family', 'CITY']).
 ↪sum(axis=1)
buildings['Density'] = buildings['1-4 Family'] / data['SHAPE_AREA']
most_buildings = buildings.sort_values('1-4 Family', ascending=False)[['CITY',␣
 ↪'1-4 Family']]
most_dense = buildings.sort_values('Density', ascending=False)[['CITY',␣
 ↪'Density']]
```

1.1.3 3. The City with The Most People Speaking Second Language

[5]:
```
spanish = data[['CITY', 'Spanish Speaker', 'Latino or Hispanic']].copy()
population = pd.read_csv('MA cities population.csv', dtype={'Population':
 ↪'int32'})
spanish = spanish.merge(population, on='CITY', how='left')
spanish = spanish.fillna(0.0)
spanish['Spanish Speaker Population'] = spanish['Spanish Speaker'] / 100 *␣
 ↪spanish['Population']
most_spanish = spanish.sort_values('Spanish Speaker Population',␣
 ↪ascending=False)[['CITY', 'Spanish Speaker Population']]
```

1.1.4 4. The City with The Most Renter

[6]:
```
owner_renter = data[['CITY', 'OWNER', 'RENTER']].fillna(0).astype({'OWNER':
 ↪'int32', 'RENTER':'int32'}).copy()
most_renters = owner_renter.sort_values('RENTER', ascending=False)[['CITY',␣
 ↪'RENTER']].rename(columns={'RENTER':'Renter'})
```

[7]:
```
mosts = [most_buildings, most_dense, most_spanish, most_renters]
input_data = most_distance.copy()
for criteria in mosts:
    input_data = input_data.merge(criteria, how='left', on='CITY')
input_data = input_data.dropna()
input_data.sample(10)
```

[7]:

	CITY	Distance	1-4 Family	Density	\
132	TYNGSBOROUGH	22.830409	3822.0	0.000082	

```
0       SOMERVILLE    1.282582     13335.0  0.001246
120   NEW BEDFORD   19.715794     19236.0  0.000366
68    BRIDGEWATER   12.519884      6079.0  0.000083
63        DIGHTON   11.989034      1165.0  0.000020
334         LENOX  149.848976      1417.0  0.000025
169         BOLTON   34.552039       999.0  0.000019
245          ATHOL   77.839426      3694.0  0.000043
113       CONCORD   18.714195      4915.0  0.000073
189    OAK BLUFFS   42.267162      2969.0  0.000155

     Spanish Speaker Population  Renter
132                     194.1685    1057
0                      6533.2080    5759
120                   14666.8294    6156
68                      864.4747    1027
63                       42.2251     157
334                     136.9488     700
169                      24.4170     160
245                     371.9044     847
113                     913.7394     897
189                      33.1357    2871
```

1.2 B. Scoring

In order to quantify how 'good' a city is, I need to come up with a way to score them. Fortunately, all of the values are already in numerical format. One of ways to do score them is by normalizing the values them sum them all up. This will result in a score number with a specific range no matter how different the numbers are. Also, since some criteria like distance for example, I want the score gets higher the less distance a city has, I need to make the values negative before I normalized them.

[8]:
```
normalized_data = input_data.copy().drop(columns=['CITY'])
normalized_data['Distance'] = normalized_data['Distance'] * -1
```

[9]:
```
normalized_data = (normalized_data-normalized_data.min())/(normalized_data.
 ↪max()-normalized_data.min())
normalized_data['Score'] = normalized_data.sum(axis=1)
normalized_data['Score'] = (normalized_data['Score']-normalized_data['Score'].
 ↪min())/(normalized_data['Score'].max()-normalized_data['Score'].min())
normalized_data['CITY'] = input_data['CITY']
normalized_data.sort_values('Score', ascending=False)
```

[9]:
```
     Distance  1-4 Family   Density  Spanish Speaker Population    Renter  \
4    0.990176    1.000000  0.377651                    1.000000  1.000000
0    1.000000    0.218380  1.000000                    0.055981  0.084876
201  0.708336    0.532419  0.261582                    0.237062  0.319106
89   0.909412    0.363967  0.473434                    0.132679  0.147466
```

```
57   0.941165    0.254384  0.416441                    0.292909  0.057543
..        …           …         …                         …         …
337  0.047756    0.004133  0.004232                    0.000047  0.001756
340  0.036371    0.005200  0.005393                    0.000335  0.003852
342  0.014034    0.004035  0.004167                    0.000534  0.003040
341  0.017224    0.002034  0.003682                    0.000041  0.001077
343  0.000000    0.001837  0.001627                    0.000000  0.001594

        Score              CITY
4    1.000000            BOSTON
0    0.539607        SOMERVILLE
201  0.470675         WORCESTER
89   0.463444            LOWELL
57   0.448656              LYNN
..        …                 …
337  0.012118          RICHMOND
340  0.010565  WEST STOCKBRIDGE
342  0.004757          EGREMONT
341  0.004355            ALFORD
343  0.000000  MOUNT WASHINGTON

[344 rows x 7 columns]
```

1.3 C. Weighting

I got the score however the problem still remains like the analysis notebook, all the criteria is still fixed to be equally important. Here is where weight coefficient comes into the play. First, I will set the weight to be equally distributed to prove if we still have the same result as above.

[10]:
```
weights = {
    'Distance': 0.2,
    '1-4 Family': 0.2,
    'Density': 0.2,
    'Spanish Speaker Population': 0.2,
    'Renter': 0.2
}
```

[11]:
```
copy_data = normalized_data.copy()
for col, weight in weights.items():
    copy_data[col] = copy_data[col] * weight
normalized_data['Score'] = copy_data[list(weights.keys())].sum(axis=1)
normalized_data['Score'] = (normalized_data['Score']-normalized_data['Score'].
 ↪min())/(normalized_data['Score'].max()-normalized_data['Score'].min())
normalized_data.sort_values('Score', ascending=False)
```

[11]:
```
     Distance  1-4 Family   Density  Spanish Speaker Population    Renter  \
4    0.990176    1.000000  0.377651                    1.000000  1.000000
```

```
0    1.000000    0.218380  1.000000                    0.055981  0.084876
201  0.708336    0.532419  0.261582                    0.237062  0.319106
89   0.909412    0.363967  0.473434                    0.132679  0.147466
57   0.941165    0.254384  0.416441                    0.292909  0.057543
..        …           …       …                         …         …
337  0.047756    0.004133  0.004232                    0.000047  0.001756
340  0.036371    0.005200  0.005393                    0.000335  0.003852
342  0.014034    0.004035  0.004167                    0.000534  0.003040
341  0.017224    0.002034  0.003682                    0.000041  0.001077
343  0.000000    0.001837  0.001627                    0.000000  0.001594

        Score              CITY
4    1.000000            BOSTON
0    0.539607        SOMERVILLE
201  0.470675         WORCESTER
89   0.463444            LOWELL
57   0.448656              LYNN
..        …                 …
337  0.012118          RICHMOND
340  0.010565  WEST STOCKBRIDGE
342  0.004757          EGREMONT
341  0.004355            ALFORD
343  0.000000  MOUNT WASHINGTON

[344 rows x 7 columns]
```

The score remains the same despite the weights on each criteria as expected.

[12]:
```
weights = {
    'Distance': 0.1,
    '1-4 Family': 0.4,
    'Density': 0.2,
    'Spanish Speaker Population': 0.2,
    'Renter': 0.1
}
copy_data = normalized_data.copy()
for col, weight in weights.items():
    copy_data[col] = copy_data[col] * weight
normalized_data['Score'] = copy_data[list(weights.keys())].sum(axis=1)
normalized_data['Score'] = (normalized_data['Score']-normalized_data['Score'].
 ↪min())/(normalized_data['Score'].max()-normalized_data['Score'].min())
normalized_data.sort_values('Score', ascending=False)
```

[12]:
```
     Distance  1-4 Family   Density  Spanish Speaker Population    Renter  \
4    0.990176    1.000000  0.377651                    1.000000  1.000000
201  0.708336    0.532419  0.261582                    0.237062  0.319106
270  0.393835    0.515935  0.294624                    0.438693  0.168689
```

```
0    1.000000    0.218380  1.000000                    0.055981  0.084876
89   0.909412    0.363967  0.473434                    0.132679  0.147466
..        …           …         …                           …         …
337  0.047756    0.004133  0.004232                    0.000047  0.001756
340  0.036371    0.005200  0.005393                    0.000335  0.003852
342  0.014034    0.004035  0.004167                    0.000534  0.003040
341  0.017224    0.002034  0.003682                    0.000041  0.001077
343  0.000000    0.001837  0.001627                    0.000000  0.001594

        Score              CITY
4    1.000000            BOSTON
201  0.474302         WORCESTER
270  0.467259       SPRINGFIELD
0    0.464678        SOMERVILLE
89   0.425130            LOWELL
..        …                 …
337  0.007146          RICHMOND
340  0.006902  WEST STOCKBRIDGE
342  0.003483          EGREMONT
341  0.002483            ALFORD
343  0.000000  MOUNT WASHINGTON

[344 rows x 7 columns]
```

After I change the weight where 1-4 family criteria is more important than the distance and renter, the new table shows slightly different results

1.4 D. Visualization

On analysis notebook I used ipyleaflet module to make an interactive maps. However, in order to display an interactive choropleth maps with live calculation, the module will not be sufficient. It was not built for this purpose, so I need to use a simpler one, Matplotlib.

[13]:
```
from ipywidgets.widgets import interact, FloatSlider
from IPython.display import display
import numpy as np
```

[14]:
```
# Setting Default Weights
Distance = FloatSlider(min=0.0, max=1.0, step=0.05, value=0.2,␣
 ↪description='Distance')
Family = FloatSlider(min=0.0, max=1.0, step=0.05, value=0.2, description='1-4␣
 ↪Family')
Density = FloatSlider(min=0.0, max=1.0, step=0.05, value=0.2,␣
 ↪description='Density')
Spanish = FloatSlider(min=0.0, max=1.0, step=0.05, value=0.2,␣
 ↪description='Spanish Speaker Population')
```

```
Renter = FloatSlider(min=0.0, max=1.0, step=0.05, value=0.2,␣
↪description='Renter')
weights = {
    'Distance': Distance,
    '1-4 Family': Family,
    'Density': Density,
    'Spanish Speaker Population': Spanish,
    'Renter': Renter
}
visited_cities = ['CAMBRIDGE', 'BOSTON', 'METHUEN', 'LAWRENCE']
viz_data = normalized_data[~normalized_data['CITY'].isin(visited_cities)].
↪copy()[['CITY', 'Score']]
```

[15]:
```
def update_weight(source):
    '''
        Update criteria's weight.
    '''
    if source.old == source.new:
        return
    criteria = source.owner.description
    val = source.new
    weights[criteria].value = val
    difference = sum([slider.value for slider in weights.values()])
    offset = 1 - difference
    sliders = [slider for c, slider in weights.items() if c != criteria]
    sliders.sort(key=lambda slider: slider.value, reverse=offset < 0)
    sliders[0].value += offset
```

[16]:
```
def handle_change(Distance, Family, Density, Spanish, Renter):
    '''
        Handle changes from a slider and update the map.
    '''
    copy_data = normalized_data[~normalized_data['CITY'].isin(visited_cities)].
↪copy()
    for col, weight in weights.items():
        copy_data[col] = copy_data[col] * weight.value
    viz_data['Score'] = copy_data[list(weights.keys())].sum(axis=1)
    viz_data['Score'] = (viz_data['Score']-viz_data['Score'].min())/
↪(viz_data['Score'].max()-viz_data['Score'].min())
    viz_data['Score'] = viz_data['Score'] * 100
    return viz_data.sort_values('Score', ascending=False).head(10)
```

[17]:
```
_sliders = [slider.observe(update_weight, names='value') for slider in weights.
↪values()]
interact(handle_change, Distance=Distance, Family=Family, Density=Density,␣
↪Spanish=Spanish, Renter=Renter)
```

```
interactive(children=(FloatSlider(value=0.2, description='Distance', max=1.0, step=0.05), Floa
```

[17]:
```
<function __main__.handle_change(Distance, Family, Density, Spanish, Renter)>
```

[18]:
```
from ipyleaflet import (Map, GeoData, basemaps, WidgetControl, GeoJSON,
 LayersControl, Choropleth, SearchControl, Marker)
from ipywidgets import Text, HTML
from branca.colormap import linear
import geopandas as gpd
import json
```

[19]:
```
cities_shp = gpd.read_file('maps/map cities.shp')
cities_shp = cities_shp.rename(columns={'TOWN':'CITY'})
cities_shp = cities_shp[['CITY', 'geometry']].merge(viz_data, on='CITY',␣
 ↪how='left')
cities_shp.to_file("maps/_map_buffer.geojson", driver='GeoJSON')
all_in_energy = (42.3497392, -71.1067746)
zoom = 9
m = Map(center=all_in_energy, zoom=zoom)
geojson_data = json.load(open("maps/_map_buffer.geojson",'r'))
maps_data = dict(zip(viz_data['CITY'], viz_data['Score']))
for feature in geojson_data['features']:
    properties = feature['properties']
    if not properties['CITY'] in maps_data:
        maps_data[properties['CITY']] = 0
    feature.update(id=properties['CITY'])
distance_layer = Choropleth(
    geo_data=geojson_data,
    choro_data=maps_data,
    colormap=linear.YlOrRd_04,
    border_color='black',
    style={'fillOpacity': 1})
marker = Marker(location=all_in_energy, draggable=False)
html = HTML('''Hover Over Cities''')
html.layout.margin = '0px 20px 20px 20px'
control = WidgetControl(widget=html, position='topright')
m.add_control(control)
def update_html(feature, **kwargs):
     html.value = '''
     <h3><b>{}</b></h3>
     <h4>Score: {}</h4>
      '''.format(feature['properties']['CITY'],
        feature['properties']['Score'] if feature['properties']['Score'] else␣
 ↪'No data or Visited')
distance_layer.on_hover(update_html)
m.add_layer(marker)
```

```
m.add_layer(distance_layer)
m
```

```
Map(center=[42.3497392, -71.1067746], controls=(ZoomControl(options=['position', 'zoom_in_text
```

[ ]:

2 Data Analysis

This is my analysis of Massachusetts Assesor Database to help All In Energy take a data-driven decision. Live Jupyter Notebook is available here. The password is **linustorvalds**. Both Jupyter Notebooks are needed to be re run by clicking the fast-forward button next to a drop down title markdown. Please kindly wait until all cells are executed indicated by the website icon on the top turned from a hourglass icon into a book icon.

analysis

November 26, 2020

1 Finding The Next Best City.

I am going to help people in All In Energy take a data-driven approach in deciding where to expand next. I am using Mass Assessor Database as the basis to survey the area and discover which cities have potential. Then, I am going to use All In Energy data to do a comparison between the potential cities and determine which would be a good choice to repeat the success achived in Boston, Cambridge, Methuen, and Lawrence.

2 The Goals

"The best" is a relative term used to compare. Defining what "the best" means here is crucial to this process as it will shape our decisions about what we are looking for. The metrics for determining the best city should be: * Have the least amount of distance from All In Energy in Boston. * Have the most amount of 1-4 Family buildings. * Have the most population that speaks other language than English. * Have the most renters.

2.1 A. Jupyter Notebook

The raw data from the Mass Government website was pretty huge. It was around 2GB originally, and, realizing my personal laptop could not do suffiecient work with such big data, I decided to register for Google Colab. It is a free Jupyter Notebook that is hosted by Google, and it utilizes the substantial amount of computing power that Google provides for this purpose.

2.2 B. Data

Although Assessor Database provided a lot of useful information, I had to strip down the raw data to make it the way I needed it to be. The MassSave Program is focused on 1-4 family buildings and 5-49 residence houses. So, I used that category to filter the data and took only fields that I thought were going to be useful. This is what comes from that cleaning process:

[1]:
```
%pip install geopandas
%pip install ipyleaflet
%pip install -U matplotlib
%pip install jupyter_contrib_nbextensions
!jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
Requirement already satisfied: geopandas in /opt/conda/lib/python3.8/site-
packages (0.8.1)
Requirement already satisfied: pyproj>=2.2.0 in /opt/conda/lib/python3.8/site-
```

```
packages (from geopandas) (3.0.0.post1)
Requirement already satisfied: shapely in /opt/conda/lib/python3.8/site-packages
(from geopandas) (1.7.1)
Requirement already satisfied: pandas>=0.23.0 in /opt/conda/lib/python3.8/site-
packages (from geopandas) (1.1.1)
Requirement already satisfied: fiona in /opt/conda/lib/python3.8/site-packages
(from geopandas) (1.8.18)
Requirement already satisfied: certifi in /opt/conda/lib/python3.8/site-packages
(from pyproj>=2.2.0->geopandas) (2020.6.20)
Requirement already satisfied: python-dateutil>=2.7.3 in
/opt/conda/lib/python3.8/site-packages (from pandas>=0.23.0->geopandas) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in /opt/conda/lib/python3.8/site-
packages (from pandas>=0.23.0->geopandas) (2020.1)
Requirement already satisfied: numpy>=1.15.4 in /opt/conda/lib/python3.8/site-
packages (from pandas>=0.23.0->geopandas) (1.19.1)
Requirement already satisfied: click<8,>=4.0 in /opt/conda/lib/python3.8/site-
packages (from fiona->geopandas) (7.1.2)
Requirement already satisfied: six>=1.7 in /opt/conda/lib/python3.8/site-
packages (from fiona->geopandas) (1.15.0)
Requirement already satisfied: cligj>=0.5 in /opt/conda/lib/python3.8/site-
packages (from fiona->geopandas) (0.7.1)
Requirement already satisfied: munch in /opt/conda/lib/python3.8/site-packages
(from fiona->geopandas) (2.5.0)
Requirement already satisfied: click-plugins>=1.0 in
/opt/conda/lib/python3.8/site-packages (from fiona->geopandas) (1.1.1)
Requirement already satisfied: attrs>=17 in /opt/conda/lib/python3.8/site-
packages (from fiona->geopandas) (20.1.0)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: ipyleaflet in /opt/conda/lib/python3.8/site-
packages (0.13.3)
Requirement already satisfied: traittypes<3,>=0.2.1 in
/opt/conda/lib/python3.8/site-packages (from ipyleaflet) (0.2.1)
Requirement already satisfied: ipywidgets<8,>=7.5.0 in
/opt/conda/lib/python3.8/site-packages (from ipyleaflet) (7.5.1)
Requirement already satisfied: branca<0.5,>=0.3.1 in
/opt/conda/lib/python3.8/site-packages (from ipyleaflet) (0.4.1)
Requirement already satisfied: traitlets>=4.2.2 in
/opt/conda/lib/python3.8/site-packages (from traittypes<3,>=0.2.1->ipyleaflet)
(4.3.3)
Requirement already satisfied: ipykernel>=4.5.1 in
/opt/conda/lib/python3.8/site-packages (from ipywidgets<8,>=7.5.0->ipyleaflet)
(5.3.4)
Requirement already satisfied: ipython>=4.0.0; python_version >= "3.3" in
/opt/conda/lib/python3.8/site-packages (from ipywidgets<8,>=7.5.0->ipyleaflet)
(7.17.0)
Requirement already satisfied: widgetsnbextension~=3.5.0 in
/opt/conda/lib/python3.8/site-packages (from ipywidgets<8,>=7.5.0->ipyleaflet)
(3.5.1)
```

```
Requirement already satisfied: nbformat>=4.2.0 in /opt/conda/lib/python3.8/site-
packages (from ipywidgets<8,>=7.5.0->ipyleaflet) (5.0.7)
Requirement already satisfied: jinja2 in /opt/conda/lib/python3.8/site-packages
(from branca<0.5,>=0.3.1->ipyleaflet) (2.11.2)
Requirement already satisfied: six in /opt/conda/lib/python3.8/site-packages
(from traitlets>=4.2.2->traittypes<3,>=0.2.1->ipyleaflet) (1.15.0)
Requirement already satisfied: ipython-genutils in
/opt/conda/lib/python3.8/site-packages (from
traitlets>=4.2.2->traittypes<3,>=0.2.1->ipyleaflet) (0.2.0)
Requirement already satisfied: decorator in /opt/conda/lib/python3.8/site-
packages (from traitlets>=4.2.2->traittypes<3,>=0.2.1->ipyleaflet) (4.4.2)
Requirement already satisfied: tornado>=4.2 in /opt/conda/lib/python3.8/site-
packages (from ipykernel>=4.5.1->ipywidgets<8,>=7.5.0->ipyleaflet) (6.0.4)
Requirement already satisfied: jupyter-client in /opt/conda/lib/python3.8/site-
packages (from ipykernel>=4.5.1->ipywidgets<8,>=7.5.0->ipyleaflet) (6.1.6)
Requirement already satisfied: pickleshare in /opt/conda/lib/python3.8/site-
packages (from ipython>=4.0.0; python_version >=
"3.3"->ipywidgets<8,>=7.5.0->ipyleaflet) (0.7.5)
Requirement already satisfied: jedi>=0.10 in /opt/conda/lib/python3.8/site-
packages (from ipython>=4.0.0; python_version >=
"3.3"->ipywidgets<8,>=7.5.0->ipyleaflet) (0.17.2)
Requirement already satisfied: pygments in /opt/conda/lib/python3.8/site-
packages (from ipython>=4.0.0; python_version >=
"3.3"->ipywidgets<8,>=7.5.0->ipyleaflet) (2.6.1)
Requirement already satisfied: backcall in /opt/conda/lib/python3.8/site-
packages (from ipython>=4.0.0; python_version >=
"3.3"->ipywidgets<8,>=7.5.0->ipyleaflet) (0.2.0)
Requirement already satisfied: pexpect; sys_platform != "win32" in
/opt/conda/lib/python3.8/site-packages (from ipython>=4.0.0; python_version >=
"3.3"->ipywidgets<8,>=7.5.0->ipyleaflet) (4.8.0)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in
/opt/conda/lib/python3.8/site-packages (from ipython>=4.0.0; python_version >=
"3.3"->ipywidgets<8,>=7.5.0->ipyleaflet) (3.0.6)
Requirement already satisfied: setuptools>=18.5 in
/opt/conda/lib/python3.8/site-packages (from ipython>=4.0.0; python_version >=
"3.3"->ipywidgets<8,>=7.5.0->ipyleaflet) (49.6.0.post20200814)
Requirement already satisfied: notebook>=4.4.1 in /opt/conda/lib/python3.8/site-
packages (from widgetsnbextension~=3.5.0->ipywidgets<8,>=7.5.0->ipyleaflet)
(6.1.3)
Requirement already satisfied: jupyter-core in /opt/conda/lib/python3.8/site-
packages (from nbformat>=4.2.0->ipywidgets<8,>=7.5.0->ipyleaflet) (4.6.3)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in
/opt/conda/lib/python3.8/site-packages (from
nbformat>=4.2.0->ipywidgets<8,>=7.5.0->ipyleaflet) (3.2.0)
Requirement already satisfied: MarkupSafe>=0.23 in
/opt/conda/lib/python3.8/site-packages (from
jinja2->branca<0.5,>=0.3.1->ipyleaflet) (1.1.1)
Requirement already satisfied: pyzmq>=13 in /opt/conda/lib/python3.8/site-
```

```
packages (from jupyter-
client->ipykernel>=4.5.1->ipywidgets<8,>=7.5.0->ipyleaflet) (19.0.2)
Requirement already satisfied: python-dateutil>=2.1 in
/opt/conda/lib/python3.8/site-packages (from jupyter-
client->ipykernel>=4.5.1->ipywidgets<8,>=7.5.0->ipyleaflet) (2.8.1)
Requirement already satisfied: parso<0.8.0,>=0.7.0 in
/opt/conda/lib/python3.8/site-packages (from jedi>=0.10->ipython>=4.0.0;
python_version >= "3.3"->ipywidgets<8,>=7.5.0->ipyleaflet) (0.7.1)
Requirement already satisfied: ptyprocess>=0.5 in /opt/conda/lib/python3.8/site-
packages (from pexpect; sys_platform != "win32"->ipython>=4.0.0; python_version
>= "3.3"->ipywidgets<8,>=7.5.0->ipyleaflet) (0.6.0)
Requirement already satisfied: wcwidth in /opt/conda/lib/python3.8/site-packages
(from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->ipython>=4.0.0;
python_version >= "3.3"->ipywidgets<8,>=7.5.0->ipyleaflet) (0.2.5)
Requirement already satisfied: nbconvert in /opt/conda/lib/python3.8/site-
packages (from
notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets<8,>=7.5.0->ipyleaflet)
(5.6.1)
Requirement already satisfied: prometheus-client in
/opt/conda/lib/python3.8/site-packages (from
notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets<8,>=7.5.0->ipyleaflet)
(0.8.0)
Requirement already satisfied: terminado>=0.8.3 in
/opt/conda/lib/python3.8/site-packages (from
notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets<8,>=7.5.0->ipyleaflet)
(0.8.3)
Requirement already satisfied: Send2Trash in /opt/conda/lib/python3.8/site-
packages (from
notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets<8,>=7.5.0->ipyleaflet)
(1.5.0)
Requirement already satisfied: argon2-cffi in /opt/conda/lib/python3.8/site-
packages (from
notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets<8,>=7.5.0->ipyleaflet)
(20.1.0)
Requirement already satisfied: attrs>=17.4.0 in /opt/conda/lib/python3.8/site-
packages (from
jsonschema!=2.5.0,>=2.4->nbformat>=4.2.0->ipywidgets<8,>=7.5.0->ipyleaflet)
(20.1.0)
Requirement already satisfied: pyrsistent>=0.14.0 in
/opt/conda/lib/python3.8/site-packages (from
jsonschema!=2.5.0,>=2.4->nbformat>=4.2.0->ipywidgets<8,>=7.5.0->ipyleaflet)
(0.16.0)
Requirement already satisfied: bleach in /opt/conda/lib/python3.8/site-packages
(from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets<8,>=7.5.
0->ipyleaflet) (3.1.5)
Requirement already satisfied: testpath in /opt/conda/lib/python3.8/site-
packages (from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets
<8,>=7.5.0->ipyleaflet) (0.4.4)
```

```
Requirement already satisfied: mistune<2,>=0.8.1 in
/opt/conda/lib/python3.8/site-packages (from nbconvert->notebook>=4.4.1->widgets
nbextension~=3.5.0->ipywidgets<8,>=7.5.0->ipyleaflet) (0.8.4)
Requirement already satisfied: entrypoints>=0.2.2 in
/opt/conda/lib/python3.8/site-packages (from nbconvert->notebook>=4.4.1->widgets
nbextension~=3.5.0->ipywidgets<8,>=7.5.0->ipyleaflet) (0.3)
Requirement already satisfied: defusedxml in /opt/conda/lib/python3.8/site-
packages (from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets
<8,>=7.5.0->ipyleaflet) (0.6.0)
Requirement already satisfied: pandocfilters>=1.4.1 in
/opt/conda/lib/python3.8/site-packages (from nbconvert->notebook>=4.4.1->widgets
nbextension~=3.5.0->ipywidgets<8,>=7.5.0->ipyleaflet) (1.4.2)
Requirement already satisfied: cffi>=1.0.0 in /opt/conda/lib/python3.8/site-
packages (from argon2-cffi->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidge
ts<8,>=7.5.0->ipyleaflet) (1.14.1)
Requirement already satisfied: packaging in /opt/conda/lib/python3.8/site-
packages (from bleach->nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ip
ywidgets<8,>=7.5.0->ipyleaflet) (20.4)
Requirement already satisfied: webencodings in /opt/conda/lib/python3.8/site-
packages (from bleach->nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ip
ywidgets<8,>=7.5.0->ipyleaflet) (0.5.1)
Requirement already satisfied: pycparser in /opt/conda/lib/python3.8/site-
packages (from cffi>=1.0.0->argon2-cffi->notebook>=4.4.1->widgetsnbextension~=3.
5.0->ipywidgets<8,>=7.5.0->ipyleaflet) (2.20)
Requirement already satisfied: pyparsing>=2.0.2 in
/opt/conda/lib/python3.8/site-packages (from packaging->bleach->nbconvert->noteb
ook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets<8,>=7.5.0->ipyleaflet) (2.4.7)
Note: you may need to restart the kernel to use updated packages.
Requirement already up-to-date: matplotlib in /opt/conda/lib/python3.8/site-
packages (3.3.3)
Requirement already satisfied, skipping upgrade: numpy>=1.15 in
/opt/conda/lib/python3.8/site-packages (from matplotlib) (1.19.1)
Requirement already satisfied, skipping upgrade: kiwisolver>=1.0.1 in
/opt/conda/lib/python3.8/site-packages (from matplotlib) (1.2.0)
Requirement already satisfied, skipping upgrade: pillow>=6.2.0 in
/opt/conda/lib/python3.8/site-packages (from matplotlib) (7.2.0)
Requirement already satisfied, skipping upgrade: cycler>=0.10 in
/opt/conda/lib/python3.8/site-packages (from matplotlib) (0.10.0)
Requirement already satisfied, skipping upgrade: python-dateutil>=2.1 in
/opt/conda/lib/python3.8/site-packages (from matplotlib) (2.8.1)
Requirement already satisfied, skipping upgrade:
pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in /opt/conda/lib/python3.8/site-
packages (from matplotlib) (2.4.7)
Requirement already satisfied, skipping upgrade: six in
/opt/conda/lib/python3.8/site-packages (from cycler>=0.10->matplotlib) (1.15.0)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: jupyter_contrib_nbextensions in
/opt/conda/lib/python3.8/site-packages (0.5.1)
```

```
Requirement already satisfied: jupyter-core in /opt/conda/lib/python3.8/site-
packages (from jupyter_contrib_nbextensions) (4.6.3)
Requirement already satisfied: jupyter-nbextensions-configurator>=0.4.0 in
/opt/conda/lib/python3.8/site-packages (from jupyter_contrib_nbextensions)
(0.4.1)
Requirement already satisfied: notebook>=4.0 in /opt/conda/lib/python3.8/site-
packages (from jupyter_contrib_nbextensions) (6.1.3)
Requirement already satisfied: tornado in /opt/conda/lib/python3.8/site-packages
(from jupyter_contrib_nbextensions) (6.0.4)
Requirement already satisfied: jupyter-latex-envs>=1.3.8 in
/opt/conda/lib/python3.8/site-packages (from jupyter_contrib_nbextensions)
(1.4.6)
Requirement already satisfied: traitlets>=4.1 in /opt/conda/lib/python3.8/site-
packages (from jupyter_contrib_nbextensions) (4.3.3)
Requirement already satisfied: ipython-genutils in
/opt/conda/lib/python3.8/site-packages (from jupyter_contrib_nbextensions)
(0.2.0)
Requirement already satisfied: nbconvert>=4.2 in /opt/conda/lib/python3.8/site-
packages (from jupyter_contrib_nbextensions) (5.6.1)
Requirement already satisfied: lxml in /opt/conda/lib/python3.8/site-packages
(from jupyter_contrib_nbextensions) (4.6.1)
Requirement already satisfied: pyyaml in /opt/conda/lib/python3.8/site-packages
(from jupyter_contrib_nbextensions) (5.3.1)
Requirement already satisfied: jupyter-contrib-core>=0.3.3 in
/opt/conda/lib/python3.8/site-packages (from jupyter_contrib_nbextensions)
(0.3.3)
Requirement already satisfied: jupyter-highlight-selected-word>=0.1.1 in
/opt/conda/lib/python3.8/site-packages (from jupyter_contrib_nbextensions)
(0.2.0)
Requirement already satisfied: terminado>=0.8.3 in
/opt/conda/lib/python3.8/site-packages (from
notebook>=4.0->jupyter_contrib_nbextensions) (0.8.3)
Requirement already satisfied: Send2Trash in /opt/conda/lib/python3.8/site-
packages (from notebook>=4.0->jupyter_contrib_nbextensions) (1.5.0)
Requirement already satisfied: nbformat in /opt/conda/lib/python3.8/site-
packages (from notebook>=4.0->jupyter_contrib_nbextensions) (5.0.7)
Requirement already satisfied: pyzmq>=17 in /opt/conda/lib/python3.8/site-
packages (from notebook>=4.0->jupyter_contrib_nbextensions) (19.0.2)
Requirement already satisfied: argon2-cffi in /opt/conda/lib/python3.8/site-
packages (from notebook>=4.0->jupyter_contrib_nbextensions) (20.1.0)
Requirement already satisfied: jupyter-client>=5.3.4 in
/opt/conda/lib/python3.8/site-packages (from
notebook>=4.0->jupyter_contrib_nbextensions) (6.1.6)
Requirement already satisfied: jinja2 in /opt/conda/lib/python3.8/site-packages
(from notebook>=4.0->jupyter_contrib_nbextensions) (2.11.2)
Requirement already satisfied: prometheus-client in
/opt/conda/lib/python3.8/site-packages (from
notebook>=4.0->jupyter_contrib_nbextensions) (0.8.0)
```

```
Requirement already satisfied: ipykernel in /opt/conda/lib/python3.8/site-
packages (from notebook>=4.0->jupyter_contrib_nbextensions) (5.3.4)
Requirement already satisfied: ipython in /opt/conda/lib/python3.8/site-packages
(from jupyter-latex-envs>=1.3.8->jupyter_contrib_nbextensions) (7.17.0)
Requirement already satisfied: six in /opt/conda/lib/python3.8/site-packages
(from traitlets>=4.1->jupyter_contrib_nbextensions) (1.15.0)
Requirement already satisfied: decorator in /opt/conda/lib/python3.8/site-
packages (from traitlets>=4.1->jupyter_contrib_nbextensions) (4.4.2)
Requirement already satisfied: pygments in /opt/conda/lib/python3.8/site-
packages (from nbconvert>=4.2->jupyter_contrib_nbextensions) (2.6.1)
Requirement already satisfied: bleach in /opt/conda/lib/python3.8/site-packages
(from nbconvert>=4.2->jupyter_contrib_nbextensions) (3.1.5)
Requirement already satisfied: defusedxml in /opt/conda/lib/python3.8/site-
packages (from nbconvert>=4.2->jupyter_contrib_nbextensions) (0.6.0)
Requirement already satisfied: entrypoints>=0.2.2 in
/opt/conda/lib/python3.8/site-packages (from
nbconvert>=4.2->jupyter_contrib_nbextensions) (0.3)
Requirement already satisfied: testpath in /opt/conda/lib/python3.8/site-
packages (from nbconvert>=4.2->jupyter_contrib_nbextensions) (0.4.4)
Requirement already satisfied: mistune<2,>=0.8.1 in
/opt/conda/lib/python3.8/site-packages (from
nbconvert>=4.2->jupyter_contrib_nbextensions) (0.8.4)
Requirement already satisfied: pandocfilters>=1.4.1 in
/opt/conda/lib/python3.8/site-packages (from
nbconvert>=4.2->jupyter_contrib_nbextensions) (1.4.2)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.8/site-
packages (from jupyter-contrib-core>=0.3.3->jupyter_contrib_nbextensions)
(49.6.0.post20200814)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in
/opt/conda/lib/python3.8/site-packages (from
nbformat->notebook>=4.0->jupyter_contrib_nbextensions) (3.2.0)
Requirement already satisfied: cffi>=1.0.0 in /opt/conda/lib/python3.8/site-
packages (from argon2-cffi->notebook>=4.0->jupyter_contrib_nbextensions)
(1.14.1)
Requirement already satisfied: python-dateutil>=2.1 in
/opt/conda/lib/python3.8/site-packages (from jupyter-
client>=5.3.4->notebook>=4.0->jupyter_contrib_nbextensions) (2.8.1)
Requirement already satisfied: MarkupSafe>=0.23 in
/opt/conda/lib/python3.8/site-packages (from
jinja2->notebook>=4.0->jupyter_contrib_nbextensions) (1.1.1)
Requirement already satisfied: pexpect; sys_platform != "win32" in
/opt/conda/lib/python3.8/site-packages (from ipython->jupyter-latex-
envs>=1.3.8->jupyter_contrib_nbextensions) (4.8.0)
Requirement already satisfied: pickleshare in /opt/conda/lib/python3.8/site-
packages (from ipython->jupyter-latex-envs>=1.3.8->jupyter_contrib_nbextensions)
(0.7.5)
Requirement already satisfied: backcall in /opt/conda/lib/python3.8/site-
packages (from ipython->jupyter-latex-envs>=1.3.8->jupyter_contrib_nbextensions)
```

```
(0.2.0)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in
/opt/conda/lib/python3.8/site-packages (from ipython->jupyter-latex-
envs>=1.3.8->jupyter_contrib_nbextensions) (3.0.6)
Requirement already satisfied: jedi>=0.10 in /opt/conda/lib/python3.8/site-
packages (from ipython->jupyter-latex-envs>=1.3.8->jupyter_contrib_nbextensions)
(0.17.2)
Requirement already satisfied: webencodings in /opt/conda/lib/python3.8/site-
packages (from bleach->nbconvert>=4.2->jupyter_contrib_nbextensions) (0.5.1)
Requirement already satisfied: packaging in /opt/conda/lib/python3.8/site-
packages (from bleach->nbconvert>=4.2->jupyter_contrib_nbextensions) (20.4)
Requirement already satisfied: attrs>=17.4.0 in /opt/conda/lib/python3.8/site-
packages (from
jsonschema!=2.5.0,>=2.4->nbformat->notebook>=4.0->jupyter_contrib_nbextensions)
(20.1.0)
Requirement already satisfied: pyrsistent>=0.14.0 in
/opt/conda/lib/python3.8/site-packages (from
jsonschema!=2.5.0,>=2.4->nbformat->notebook>=4.0->jupyter_contrib_nbextensions)
(0.16.0)
Requirement already satisfied: pycparser in /opt/conda/lib/python3.8/site-
packages (from
cffi>=1.0.0->argon2-cffi->notebook>=4.0->jupyter_contrib_nbextensions) (2.20)
Requirement already satisfied: ptyprocess>=0.5 in /opt/conda/lib/python3.8/site-
packages (from pexpect; sys_platform != "win32"->ipython->jupyter-latex-
envs>=1.3.8->jupyter_contrib_nbextensions) (0.6.0)
Requirement already satisfied: wcwidth in /opt/conda/lib/python3.8/site-packages
(from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->ipython->jupyter-latex-
envs>=1.3.8->jupyter_contrib_nbextensions) (0.2.5)
Requirement already satisfied: parso<0.8.0,>=0.7.0 in
/opt/conda/lib/python3.8/site-packages (from jedi>=0.10->ipython->jupyter-latex-
envs>=1.3.8->jupyter_contrib_nbextensions) (0.7.1)
Requirement already satisfied: pyparsing>=2.0.2 in
/opt/conda/lib/python3.8/site-packages (from
packaging->bleach->nbconvert>=4.2->jupyter_contrib_nbextensions) (2.4.7)
Note: you may need to restart the kernel to use updated packages.
Enabling notebook extension jupyter-leaflet/extension…
      - Validating: OK
```

[2]:
```
import pandas as pd
data = pd.read_csv('MA Cities Data Oct 10.csv')
```

[3]:
```
data.sample(5)
```

[3]:

	CITY	1-Family	2-Family	3-Family	4-Family	5+ Family	OWNER	\
300	WALTHAM	7712.0	5012.0	238.0	NaN	704.0	3155.0	
156	LUNENBURG	1363.0	NaN	NaN	NaN	NaN	789.0	
68	DALTON	1601.0	401.0	25.0	18.0	5.0	1384.0	

```
310  WELLFLEET    2208.0     415.0       NaN       NaN        8.0   365.0
10   ARLINGTON    3880.0    2704.0       NaN      72.0       11.0  5182.0

      RENTER    SHAPE_AREA    distance  …    Gas   <20K 20K-39K 40K-59K  \
300  10511.0  3.564154e+07    8.977461  …   True  11.54   10.27   13.34
156    574.0  7.186314e+07   42.732611  …   True  11.11    9.37   10.68
68     666.0  5.664039e+07  142.791287  …   True   9.86     NaN     NaN
310   2266.0  5.439851e+07   76.279187  …  False   9.97     NaN   14.39
10    1485.0  1.420394e+07    4.661131  …   True   8.46   10.12    9.27

    60K-74K  75K-99K  >100K  Spanish Speaker  Black  Latino or Hispanic
300    9.46    12.36  43.02            11.39   7.49               13.49
156    5.73    11.30    NaN             3.59   2.08                5.90
68      NaN    12.55    NaN             3.91   0.20                3.95
310     NaN      NaN    NaN             0.29   1.84                0.34
10     6.58    11.82  53.75             3.15   2.63                5.06

[5 rows x 24 columns]
```

I chose Pandas for processing such big data. Pandas has been really great in helping me do a lot of things using very little code. Also, it makes it easy to massage a really large amount data with in a fraction of the time it would take otherwise. For example, cities All In Energy is currently operating on I want to exclude from the main data, so I filter them out.

[4]:
```
data.describe()
```

[4]:
```
           1-Family      2-Family     3-Family     4-Family     5+ Family  \
count    342.000000    304.000000    86.000000   107.000000    178.000000
mean    2668.570175   1798.256579   356.779070   117.317757    384.308989
std     3204.195302   2295.561841   757.935748   339.111369   2639.422005
min       21.000000      6.000000     2.000000     1.000000      1.000000
25%      818.250000    507.250000    18.250000     6.000000      4.250000
50%     1827.500000   1220.000000    80.500000    21.000000     17.000000
75%     3505.250000   2411.750000   280.000000    82.000000     56.000000
max    32736.000000  25047.000000  4775.000000  2928.000000  33505.000000

              OWNER        RENTER    SHAPE_AREA    distance  Electric MWh  \
count    339.000000    344.000000  3.440000e+02  344.000000  2.980000e+02
mean    2947.477876   1644.369186  5.949554e+07   53.565072  5.544814e+04
std     3300.623378   4209.717501  3.394042e+07   46.316631  9.470242e+04
min        1.000000      8.000000  3.249309e+06    1.282582  4.650000e+02
25%      510.000000    311.750000  3.711579e+07   15.049065  1.241850e+04
50%     2054.000000    705.500000  5.447793e+07   34.633778  3.482050e+04
75%     4356.500000   1544.250000  7.357752e+07   90.733849  6.744325e+04
max    26708.000000  67766.000000  2.661817e+08  163.520727  1.331961e+06

              Threm        <20K     20K-39K     40K-59K     60K-74K  \
```

count	2.400000e+02	300.000000	267.000000	265.000000	303.000000
mean	5.370727e+06	9.526000	12.264232	11.494679	8.432541
std	9.401798e+06	4.885163	4.427211	3.523575	2.482513
min	1.832600e+04	2.240000	1.440000	3.700000	2.190000
25%	1.231115e+06	6.305000	9.240000	8.640000	6.770000
50%	3.194283e+06	8.500000	11.650000	11.570000	8.460000
75%	6.575716e+06	11.250000	15.135000	14.260000	10.030000
max	1.171037e+08	30.700000	24.540000	20.100000	16.500000

	75K-99K	>100K	Spanish Speaker	Black	Latino or Hispanic
count	286.000000	143.000000	344.000000	335.000000	335.000000
mean	12.127657	40.155175	3.625930	2.488239	5.109642
std	2.992728	14.916932	7.064476	4.204283	8.222208
min	0.780000	13.650000	0.000000	0.000000	0.000000
25%	10.222500	27.560000	0.847500	0.480000	1.600000
50%	12.535000	40.090000	1.745000	1.420000	3.060000
75%	14.165000	52.115000	3.560000	2.730000	5.085000
max	19.460000	71.890000	74.150000	40.810000	80.260000

2.3 C. Exploratory Data Analysis

In order to answer the question, "which city is the best to expand to next?" first we need to get a grasp on what kind of data we have. I will go through and try to find cities that fit best with each criterion.

2.3.1 1. The closest city from Boston

Although the distance from the All In Energy office in Brookline is not necessarily the most important factor, it is useful to know if there is any city surrounding Boston that fits the other criteria.

[5]:
```
distance = data[['CITY', 'distance']].copy()
distance.sample(10)
```

[5]:

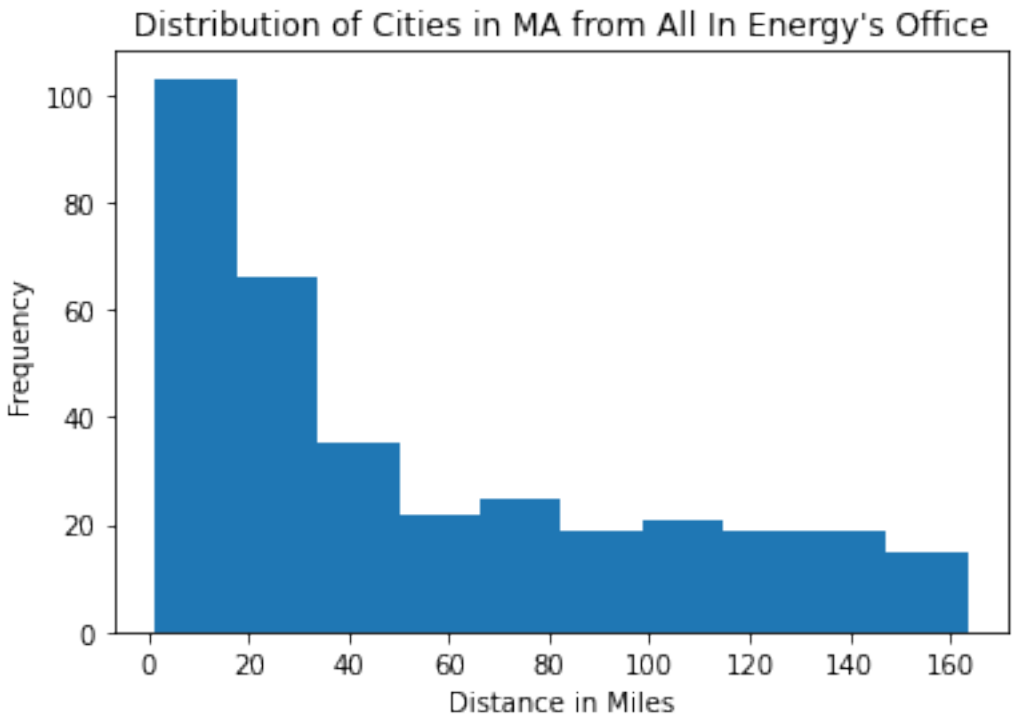
	CITY	distance
303	WARREN	75.871428
203	NORTH ANDOVER	7.621628
253	SANDWICH	46.385820
8	ANDOVER	7.921973
85	EASTHAMPTON	108.575303
173	MENDON	31.144759
83	EAST LONGMEADOW	97.078821
130	HOLLAND	73.223220
335	WINCHENDON	65.602505
179	MILFORD	29.364941

First, we can plot the distance data into a histogram in order to get a sense of how spread out our data is. I am using Matplotlib library for plotting.

[6]:
```
import matplotlib.pyplot as plt
```

[7]:
```
plt.hist(distance['distance'], bins=10)
plt.title("Distribution of Cities in MA from All In Energy's Office")
plt.xlabel('Distance in Miles')
plt.ylabel('Frequency')
```

[7]:
```
Text(0, 0.5, 'Frequency')
```



The histogram shows us that fortunately a lot of cities are in range of 60 miles or less. That is a good news because it means we have a lot of options in the range of 1 hour of driving. Next, we can map the distance values into a Choropleth map so we can better visualize them. I am using ipyleaflet for interactive map visualization.

[8]:
```
from ipyleaflet import (Map, GeoData, basemaps, WidgetControl, GeoJSON,
 LayersControl, Icon, Marker,basemap_to_tiles, Choropleth,
 MarkerCluster, Heatmap, SearchControl,
 FullScreenControl, projections, SplitMapControl)
from ipywidgets import Text, HTML
from branca.colormap import linear
import geopandas as gpd
import json
import numpy as np
import matplotlib.pyplot as plt
```

[9]:
```
def create_bins(lower_bound, width, quantity):
    """ create_bins returns an equal-width (distance) partitioning.
        It returns an ascending list of tuples, representing the intervals.
        A tuple bins[i], i.e. (bins[i][0], bins[i][1])  with i > 0
        and i < quantity, satisfies the following conditions:
            (1) bins[i][0] + width == bins[i][1]
            (2) bins[i-1][0] + width == bins[i][0] and
                bins[i-1][1] + width == bins[i][1]
        source: https://www.python-course.eu/pandas_python_binning.php
    """

    bins = []
    for low in np.arange(lower_bound,
                         lower_bound + quantity*width + 1.0, width):
        bins.append((low, low+width))
    return bins

def values_to_bins(values, lower=0, width=10, bins=5, normalize=False):
    '''
        Put values into bins so we can control the classes in Choropleth Maps.
    '''
    if normalize:
        values = np.array(values)
        values = (values - np.min(values)) / (np.max(values) - np.min(values))
        values = values * 10
    bins = create_bins(lower, width, bins)
    bins = pd.IntervalIndex.from_tuples(bins)
    categorical_object = pd.cut(values, bins)
    results = []
    for value in categorical_object:
        if not pd.isna(value):
            results.append(value.left)
        else:
            results.append(-1.0)
    results = np.array(results)
    return (results - np.min(results)) / (np.max(results) - np.min(results))
```

[10]:
```
distance['bins'] = values_to_bins(distance['distance'], width=20, bins=10)
```

[11]:
```
cities_shp = gpd.read_file('maps/map cities.shp')
cities_shp = cities_shp.rename(columns={'TOWN':'CITY'})
cities_shp = cities_shp[['CITY', 'geometry']].merge(distance[['CITY',␣
↪'distance']], on='CITY', how='left')
cities_shp.to_file("maps/distance cities.geojson", driver='GeoJSON')
```

[12]:
```
all_in_energy = (42.3497392, -71.1067746)
zoom = 9
```

```
m = Map(center=all_in_energy, zoom=zoom)
geojson_data = json.load(open("maps/distance cities.geojson",'r'))
maps_data = dict(zip(distance['CITY'], distance['bins']))
for feature in geojson_data['features']:
    properties = feature['properties']
    if not properties['CITY'] in maps_data:
        maps_data[properties['CITY']] = 0
    feature.update(id=properties['CITY'])
distance_layer = Choropleth(
    geo_data=geojson_data,
    choro_data=maps_data,
    colormap=linear.YlOrRd_04,
    border_color='black',
    style={'fillOpacity': 1})
marker = Marker(location=all_in_energy, draggable=False)
html = HTML('''Hover Over Cities''')
html.layout.margin = '0px 20px 20px 20px'
control = WidgetControl(widget=html, position='topright')
m.add_control(control)
def update_html(feature, **kwargs):
     html.value = '''
     <h3><b>{}</b></h3>
     <h4>Distance: {:.4} miles</h4>
      '''.format(feature['properties']['CITY'],
        feature['properties']['distance'] if feature['properties']['distance']␣
 ↪else -1.0)
distance_layer.on_hover(update_html)
m.add_layer(marker)
m.add_layer(distance_layer)
```

2.4 Map of Each City's Distance from Boston

[13]:
```
m
```

```
Map(center=[42.3497392, -71.1067746], controls=(ZoomControl(options=['position', 'zoom_in_text
```

Interestingly, there are a lot more of cities within a 20 mile drive to the north and south Boston than to the west. This means our next city will be more likely to be north or south of Boston. Next, we can take the 10 closest cities to Boston as our candidates.

[14]:
```
visited = ['CAMBRIDGE', 'BOSTON', 'METHUEN', 'LAWRENCE']
most_distance = distance[~distance['CITY'].isin(visited)].
 ↪sort_values('distance').head(10)
most_distance
```

[14]:

	CITY	distance	bins
266	SOMERVILLE	1.282582	0.0
45	BROOKLINE	1.593503	0.0
170	MEDFORD	1.840088	0.0
276	STONEHAM	3.149234	0.0
183	MILTON	3.326722	0.0
336	WINCHESTER	3.579450	0.0
159	MALDEN	3.892915	0.0
172	MELROSE	4.002186	0.0
238	READING	4.039968	0.0
91	EVERETT	4.102009	0.0

2.4.1 2. The City with The Most Family Buildings

The number of 1-4 family buildings is one of the most important criteria because it is one of the requirements that the MassSave program demanded. So, knowing which city has the most buildings with this kind of style will be crucial.

[15]:
```
buildings = data[['CITY', '1-Family', '2-Family', '3-Family', '4-Family', '5+␣
↪Family']].copy()
buildings.sample(10)
```

[15]:

	CITY	1-Family	2-Family	3-Family	4-Family	5+ Family
141	LAKEVILLE	2398.0	1213.0	NaN	NaN	8.0
103	GILL	406.0	69.0	NaN	2.0	NaN
226	PETERSHAM	344.0	131.0	NaN	2.0	NaN
35	BOSTON	32736.0	25047.0	277.0	2928.0	33505.0
261	SHERBORN	556.0	497.0	NaN	NaN	NaN
157	LYNN	7297.0	6614.0	1619.0	NaN	700.0
121	HARVARD	1077.0	751.0	NaN	NaN	2.0
265	SOMERSET	1676.0	NaN	NaN	NaN	NaN
124	HAWLEY	21.0	NaN	NaN	NaN	NaN
19	AYER	1097.0	NaN	NaN	NaN	NaN

We only want to see the 1-4 Family buildings. So, we need to combine the columns into one then plot them.

[16]:
```
buildings['1-4 Family'] = buildings.drop(columns=['5+ Family', 'CITY']).
↪sum(axis=1)
buildings
```

[16]:

	CITY	1-Family	2-Family	3-Family	4-Family	5+ Family	\
0	ABINGTON	2726.0	1795.0	83.0	NaN	35.0	
1	ACTON	2650.0	3019.0	NaN	NaN	34.0	
2	ACUSHNET	2650.0	615.0	NaN	NaN	NaN	
3	ADAMS	1002.0	NaN	NaN	NaN	NaN	
4	AGAWAM	7054.0	1638.0	NaN	25.0	NaN	
..	…	…	…	…	…	…	

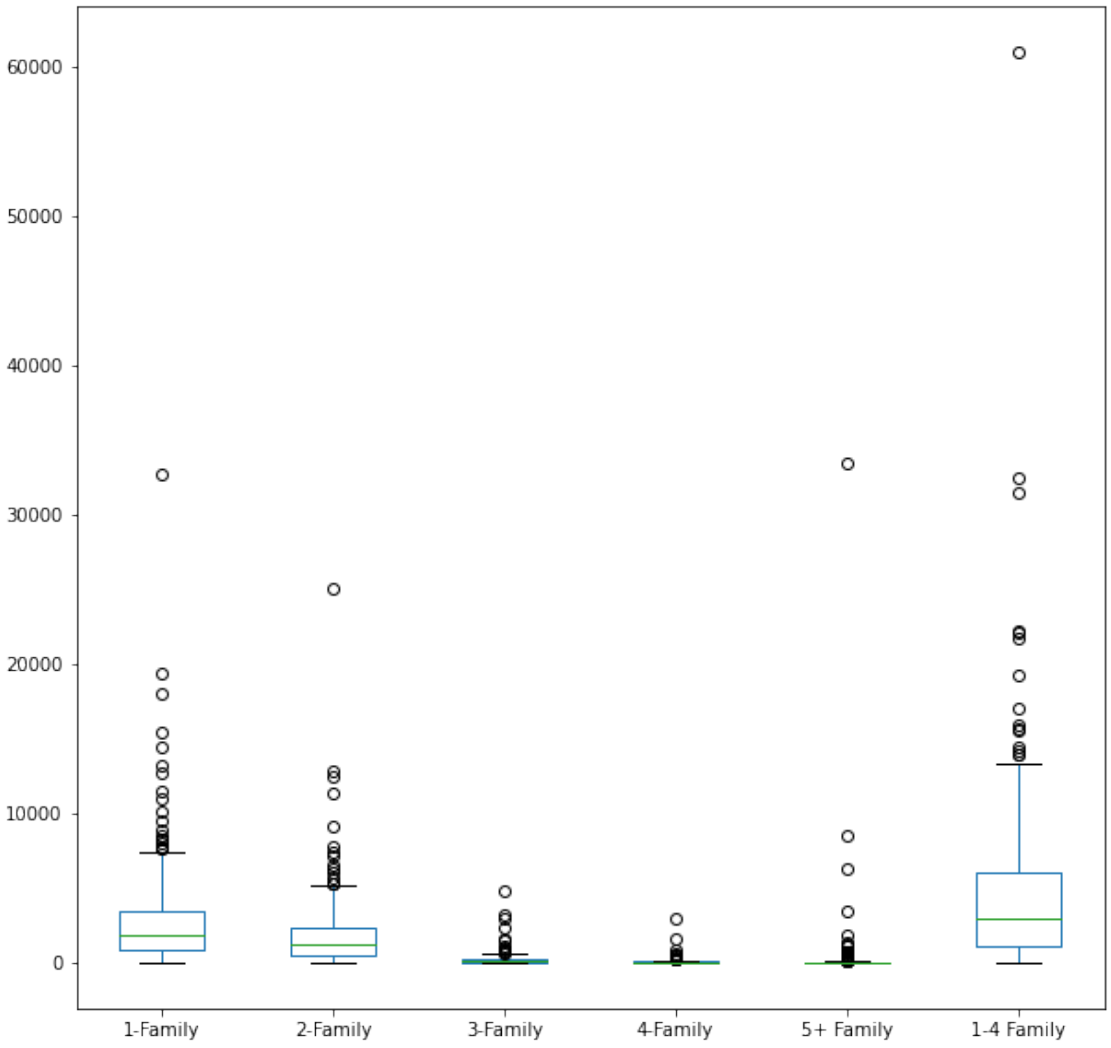
```
339       WOBURN    3112.0    4419.0     105.0      86.0       66.0
340    WORCESTER   15411.0   11373.0    4775.0     922.0     3447.0
341  WORTHINGTON     271.0     178.0       NaN       NaN        NaN
342     WRENTHAM    1585.0    1866.0       NaN       NaN       19.0
343     YARMOUTH   12767.0    1225.0       NaN       NaN      121.0

     1-4 Family
0        4604.0
1        5669.0
2        3265.0
3        1002.0
4        8717.0
..          …
339      7722.0
340     32481.0
341       449.0
342      3451.0
343     13992.0

[344 rows x 7 columns]
```

[17]:
```
buildings.plot.box(figsize=(10,10))
```

[17]:
```
<AxesSubplot:>
```



Visited Cities

[18]:
```
buildings[buildings['CITY'].isin(visited)]
```

[18]:

	CITY	1-Family	2-Family	3-Family	4-Family	5+ Family	1-4 Family
35	BOSTON	32736.0	25047.0	277.0	2928.0	33505.0	60988.0
48	CAMBRIDGE	3971.0	3678.0	281.0	336.0	8559.0	8266.0
144	LAWRENCE	5266.0	1060.0	NaN	NaN	1231.0	6326.0
175	METHUEN	6372.0	3475.0	229.0	84.0	3.0	10160.0

Looking at the box plot, 1 family buildings are most numerous in most of the cities. We can also see that there are a lot of outliers on the high side of the 1-4 family buildings category. This means we have a couple of candidate cities that have more buildings than most of cities that All In Energy has already visited. Furthermore, since All In Energy does a lot of door-to-door canvassing, knowing the density of each city is useful too. Let's calculate them.

[19]:
```
buildings['density'] = buildings['1-4 Family'] / data['SHAPE_AREA']
buildings
```

[19]:

	CITY	1-Family	2-Family	3-Family	4-Family	5+ Family	1-4 Family	density
0	ABINGTON	2726.0	1795.0	83.0	NaN	35.0	4604.0	0.000174
1	ACTON	2650.0	3019.0	NaN	NaN	34.0	5669.0	0.000108
2	ACUSHNET	2650.0	615.0	NaN	NaN	NaN	3265.0	0.000067
3	ADAMS	1002.0	NaN	NaN	NaN	NaN	1002.0	0.000017
4	AGAWAM	7054.0	1638.0	NaN	25.0	NaN	8717.0	0.000138
..	…	…	…	…	…	…	…	…
339	WOBURN	3112.0	4419.0	105.0	86.0	66.0	7722.0	0.000230
340	WORCESTER	15411.0	11373.0	4775.0	922.0	3447.0	32481.0	0.000326
341	WORTHINGTON	271.0	178.0	NaN	NaN	NaN	449.0	0.000005
342	WRENTHAM	1585.0	1866.0	NaN	NaN	19.0	3451.0	0.000059
343	YARMOUTH	12767.0	1225.0	NaN	NaN	121.0	13992.0	0.000217

[344 rows x 8 columns]

In order to compare both criteria, we can plot 2 different maps and explore it that way.

[20]:
```
cities_shp = gpd.read_file('maps/map cities.shp')
cities_shp = cities_shp.rename(columns={'TOWN':'CITY'})
cities_shp = cities_shp[['CITY', 'geometry']].merge(buildings, on='CITY',␣
 ↪how='left')
cities_shp.to_file("maps/buildings cities.geojson", driver='GeoJSON')
```

[21]:
```
all_in_energy = (42.3497392, -71.1067746)
zoom = 9
geojson_data = json.load(open("maps/buildings cities.geojson",'r'))
buildings_data = values_to_bins(buildings['1-4 Family'], width=1, bins=10,␣
 ↪normalize=True)
density_data = values_to_bins(buildings['density'], width=1, bins=10,␣
 ↪normalize=True)
building_map_data = dict(zip(buildings['CITY'], buildings_data))
```

```
density_map_data = dict(zip(buildings['CITY'], density_data))
for feature in geojson_data['features']:
    properties = feature['properties']
    if not properties['CITY'] in building_map_data:
        building_map_data[properties['CITY']] = 0
    if not properties['CITY'] in density_map_data:
        density_map_data[properties['CITY']] = 0
    feature.update(id=properties['CITY'])
```

[22]:
```
building_layer = Choropleth(
    geo_data=geojson_data,
    choro_data=building_map_data,
    colormap=linear.YlOrRd_04,
    border_color='black',
    style={'fillOpacity': 1})
density_layer = Choropleth(
    geo_data=geojson_data,
    choro_data=density_map_data,
    colormap=linear.YlOrRd_04,
    border_color='black',
    style={'fillOpacity': 1})
marker = Marker(location=all_in_energy, draggable=False)
html_buildings = HTML('''Hover Over Cities''')
html_buildings.layout.margin = '0px 20px 20px 20px'

def update_html_buildings(feature, **kwargs):
     html_buildings.value = '''
     <h3><b>{}</b></h3>
     <h4>1-4 Family buildings: {:.4} buildings</h4>
      '''.format(feature['properties']['CITY'],
        float(feature['properties']['1-4 Family']) if␣
 ↪feature['properties']['1-4 Family'] else 0.0)
html_density = HTML('''Hover Over Cities''')
html_density.layout.margin = '0px 20px 20px 20px'
def update_html_density(feature, **kwargs):
     html_density.value = '''
     <h3><b>{}</b></h3>
     <h4>Density: {:.4} buildings/sqr miles</h4>
      '''.format(feature['properties']['CITY'],
        float(feature['properties']['density']) if␣
 ↪feature['properties']['density'] else 0.0)
building_layer.on_hover(update_html_buildings)
density_layer.on_hover(update_html_density)
control_buildings = WidgetControl(widget=html_buildings, position='topright')
control_density = WidgetControl(widget=html_density, position='topright')
```

[23]:
```
choro_buildings = Map(center=all_in_energy, zoom=zoom)
choro_density = Map(center=all_in_energy, zoom=zoom)
choro_buildings.add_layer(marker)
choro_density.add_layer(marker)
choro_buildings.add_layer(building_layer)
choro_density.add_layer(density_layer)
choro_buildings.add_control(control_buildings)
choro_density.add_control(control_density)
```

2.5 Map of Total Qualified Buildings in Cities

[24]:
```
choro_buildings
```

```
Map(center=[42.3497392, -71.1067746], controls=(ZoomControl(options=['position', 'zoom_in_text
```

2.6 Map of City Density

[25]:
```
choro_density
```

```
Map(center=[42.3497392, -71.1067746], controls=(ZoomControl(options=['position', 'zoom_in_text
```

From both maps we can see that Boston is surrounded by fairly dense cities, some of which have a lot of buildings for their size. This is good news because it means that All In Energy does not need to go far to get more customers.

[26]:
```
most_buildings = buildings[~buildings['CITY'].isin(visited)].sort_values('1-4␣
↪Family', ascending=False).head(10)[['CITY', '1-4 Family']]
most_buildings
```

[26]:

	CITY	1-4 Family
340	WORCESTER	32481.0
273	SPRINGFIELD	31476.0
154	LOWELL	22211.0
20	BARNSTABLE	22057.0
200	NEWTON	21732.0
194	NEW BEDFORD	19236.0
235	QUINCY	17080.0
93	FALL RIVER	15970.0
94	FALMOUTH	15717.0
157	LYNN	15530.0

[27]:
```
most_dense = buildings[~buildings['CITY'].isin(visited)].sort_values('density',␣
↪ascending=False).head(10)[['CITY', 'density']]
most_dense
```

[27]:

	CITY	density
266	SOMERVILLE	0.001246
91	EVERETT	0.000874
338	WINTHROP	0.000773
240	REVERE	0.000708
56	CHELSEA	0.000613
154	LOWELL	0.000590
159	MALDEN	0.000559
26	BELMONT	0.000554
162	MARBLEHEAD	0.000530
157	LYNN	0.000519

2.6.1 3. The City with The Most People Speaking a Second Language

One of All In Energy's goals is to provide energy efficiency programs to the underserved communities. One type of such community is that of people who speak a language other than English, such as Spanish. Targeting cities with high percentage of Spanish speakers will definitely help achieve the goal.

[28]:
```
spanish = data[['CITY', 'Spanish Speaker', 'Latino or Hispanic']].copy()
spanish.sample(10)
```

[28]:

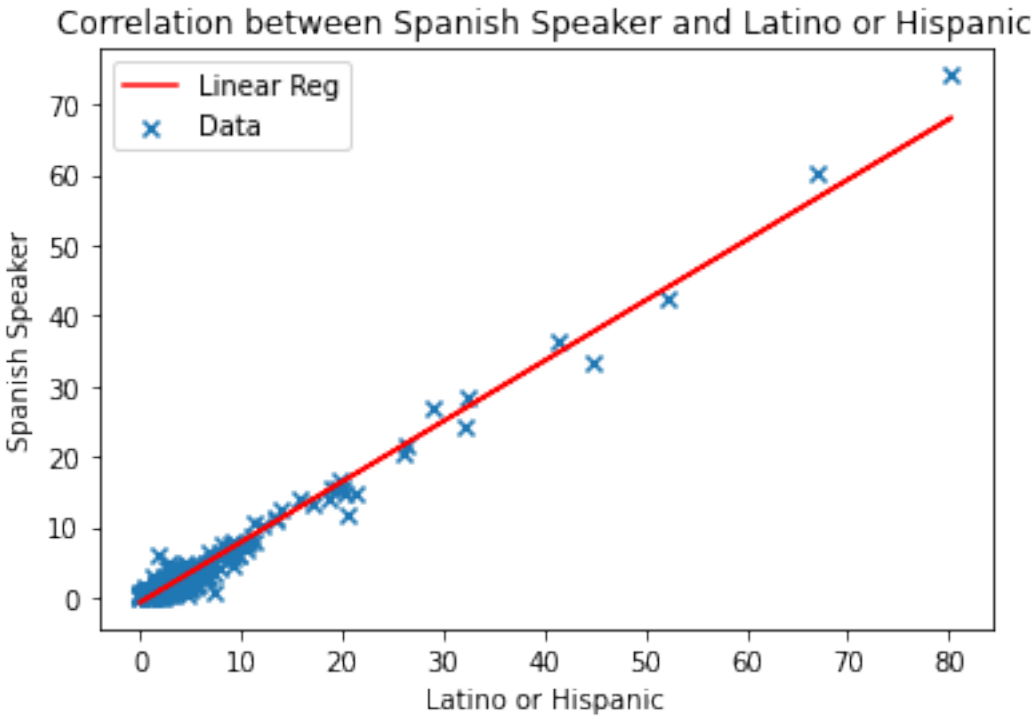
	CITY	Spanish Speaker	Latino or Hispanic
30	BEVERLY	3.06	4.48
38	BOXFORD	0.72	1.15
205	NORTH BROOKFIELD	1.16	1.01
251	SALISBURY	0.99	3.45
136	HUDSON	4.19	6.42
220	PAXTON	1.76	2.73
285	TAUNTON	4.33	7.86
310	WELLFLEET	0.29	0.34
101	GARDNER	5.86	9.17
246	ROWLEY	0.32	0.95

Since the dataset has 2 possible columns that we can use to determine the criteria, we can find their correlation. If they strongly correlate, we can just use either one. Let's use linear regression to find the correlation and use the correlation coefficient to determine the accurate result.

[29]:
```
no_na_spanish = spanish.dropna()
plt.scatter(no_na_spanish['Latino or Hispanic'], no_na_spanish['Spanish␣
 ↪Speaker'], marker='x', label='Data')
m, b = np.polyfit(no_na_spanish['Latino or Hispanic'], no_na_spanish['Spanish␣
 ↪Speaker'], 1)
plt.plot(no_na_spanish['Latino or Hispanic'], m*no_na_spanish['Latino or␣
 ↪Hispanic'] + b, color='r', label='Linear Reg')
plt.legend()
plt.xlabel('Latino or Hispanic')
plt.ylabel('Spanish Speaker')
```

```
plt.title('Correlation between Spanish Speaker and Latino or Hispanic')
```

[29]:
```
Text(0.5, 1.0, 'Correlation between Spanish Speaker and Latino or Hispanic')
```



[30]:
```
spanish.corr()
```

[30]:

	Spanish Speaker	Latino or Hispanic
Spanish Speaker	1.00000	0.98702
Latino or Hispanic	0.98702	1.00000

Both the graph and the correlation coefficient show that the Spanish speaker column and the Latino or Hispanic column have a strong correlation. Almost a perfect correlation, in fact. This means we can eliminate one of them. Since the Spanish speaker column has more complete data, let's drop the Latino or Hispanic column.

[31]:
```
spanish = spanish[['CITY', 'Spanish Speaker']].fillna(0.0)
spanish.sample(10)
```

[31]:

	CITY	Spanish Speaker
72	DEERFIELD	1.66
311	WENDELL	4.19
312	WENHAM	3.78
68	DALTON	3.91
338	WINTHROP	7.40

```
183        MILTON             3.66
29    BERNARDSTON             0.25
333  WILLIAMSTOWN             4.85
120        HANSON             0.87
147         LENOX             2.77
```

There is a problem. The data here is represented in a percentage of each city population, which means that 1% in Boston will not represent the same count as 1% in Cambridge. We need to convert them into actual population counts. I downloaded estimated population data in MA cities from UMass Donahue which will be sufficient for our purposes.

[32]:
```
population = pd.read_csv('MA cities population.csv', dtype={'Population':
 ↪'int32'})
population.sample(10)
```

[32]:
```
            CITY  Population
28    ROCHESTER         5687
115    HARDWICK         3057
53     WRENTHAM        12023
248   TEWKSBURY        31178
127   HAVERHILL        64014
326    BUCKLAND         1850
286        PERU          834
321       HEATH          695
230  WASHINGTON          541
52         STOW         7234
```

[33]:
```
spanish = spanish.merge(population, on='CITY', how='left')
spanish = spanish.fillna(0.0)
spanish['Spanish Speaker Population'] = spanish['Spanish Speaker'] / 100 *␣
 ↪spanish['Population']
spanish
```

[33]:
```
             CITY  Spanish Speaker  Population  Spanish Speaker Population
0       ABINGTON             1.36     16668.0                    226.6848
1          ACTON             1.89     23662.0                    447.2118
2       ACUSHNET             1.78     10625.0                    189.1250
3          ADAMS             1.18      8010.0                     94.5180
4         AGAWAM             3.78     28613.0                   1081.5714
..           ...              ...         ...                        ...
339       WOBURN             3.48     40228.0                   1399.9344
340    WORCESTER            14.92    185428.0                  27665.8576
341  WORTHINGTON             0.25      1175.0                      2.9375
342     WRENTHAM             1.95     12023.0                    234.4485
343     YARMOUTH             1.58     23203.0                    366.6074

[344 rows x 4 columns]
```

[34]:
```
cities_shp = gpd.read_file('maps/map cities.shp')
cities_shp = cities_shp.rename(columns={'TOWN':'CITY'})
cities_shp = cities_shp[['CITY', 'geometry']].merge(spanish, on='CITY',␣
 ↪how='left')
cities_shp.to_file("maps/spanish cities.geojson", driver='GeoJSON')
```

[35]:
```
all_in_energy = (42.3497392, -71.1067746)
zoom = 9
spanish_map = Map(center=all_in_energy, zoom=zoom)
geojson_data = json.load(open("maps/spanish cities.geojson",'r'))
spanish_data = values_to_bins(spanish['Spanish Speaker Population'], width=1,␣
 ↪bins=10, normalize=True)
maps_data = dict(zip(spanish['CITY'], spanish_data))
for feature in geojson_data['features']:
    properties = feature['properties']
    if not properties['CITY'] in maps_data:
        maps_data[properties['CITY']] = 0
    feature.update(id=properties['CITY'])
spanish_layer = Choropleth(
    geo_data=geojson_data,
    choro_data=maps_data,
    colormap=linear.YlOrRd_04,
    border_color='black',
    style={'fillOpacity': 1})
marker = Marker(location=all_in_energy, draggable=False)
html_spanish = HTML('''Hover Over Cities''')
html_spanish.layout.margin = '0px 20px 20px 20px'
control_spanish = WidgetControl(widget=html_spanish, position='topright')
spanish_map.add_control(control_spanish)
def update_html_spanish(feature, **kwargs):
     html_spanish.value = '''
     <h3><b>{}</b></h3>
     <h4>Spanish Speaker: {} People</h4>
      '''.format(feature['properties']['CITY'],
        int(feature['properties']['Spanish Speaker Population']) if␣
 ↪feature['properties']['Spanish Speaker Population'] else -1.0)
spanish_layer.on_hover(update_html_spanish)
spanish_map.add_layer(marker)
spanish_map.add_layer(spanish_layer)
```

[36]:
```
spanish_map
```

```
Map(center=[42.3497392, -71.1067746], controls=(ZoomControl(options=['position', 'zoom_in_text
```

Unfortunately, there are only a few of cities that have a substantial amount of Spanish speakers. However, this also makes the decision a little bit easier since there will be one less variable to worry about.

[37]:
```
most_spanish = spanish[~spanish['CITY'].isin(visited)].sort_values('Spanish␣
↪Speaker Population', ascending=False).head(10)[['CITY', 'Spanish Speaker␣
↪Population']]
most_spanish
```

[37]:

	CITY	Spanish Speaker Population
273	SPRINGFIELD	51196.8798
157	LYNN	34183.3875
340	WORCESTER	27665.8576
56	CHELSEA	23925.1320
132	HOLYOKE	17065.7718
154	LOWELL	15484.0815
240	REVERE	15104.5758
194	NEW BEDFORD	14666.8294
98	FRAMINGHAM	10529.8640
91	EVERETT	10005.5454

2.6.2 4. The City with The Most Renters

In research from 2019 it was concluded that renters from low income demographics are likely to be less energy efficient than other groups due to lack of education on such topics. That is why this criterion will impact a lot of decisions.

[38]:
```
owner_renter = data[['CITY', 'OWNER', 'RENTER']].fillna(0).astype({'OWNER':
↪'int32', 'RENTER':'int32'}).copy()
owner_renter
```

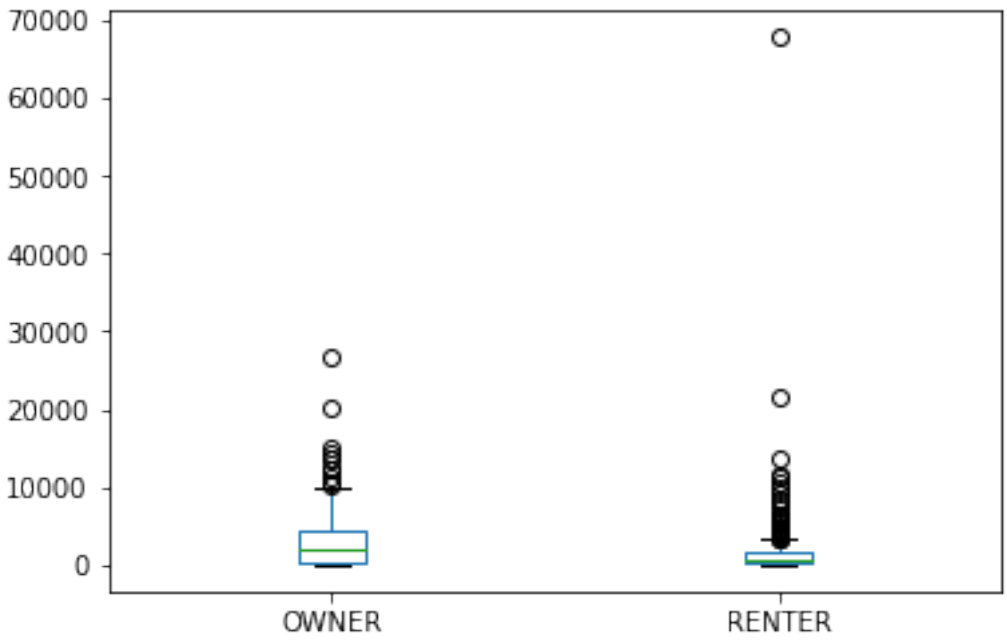
[38]:

	CITY	OWNER	RENTER
0	ABINGTON	3847	783
1	ACTON	4512	1172
2	ACUSHNET	228	3037
3	ADAMS	846	155
4	AGAWAM	7533	1118
..	…	…	…
339	WOBURN	4986	2679
340	WORCESTER	14295	21630
341	WORTHINGTON	120	328
342	WRENTHAM	2909	561
343	YARMOUTH	6716	7372

[344 rows x 3 columns]

[39]:
```
owner_renter.plot.box()
```

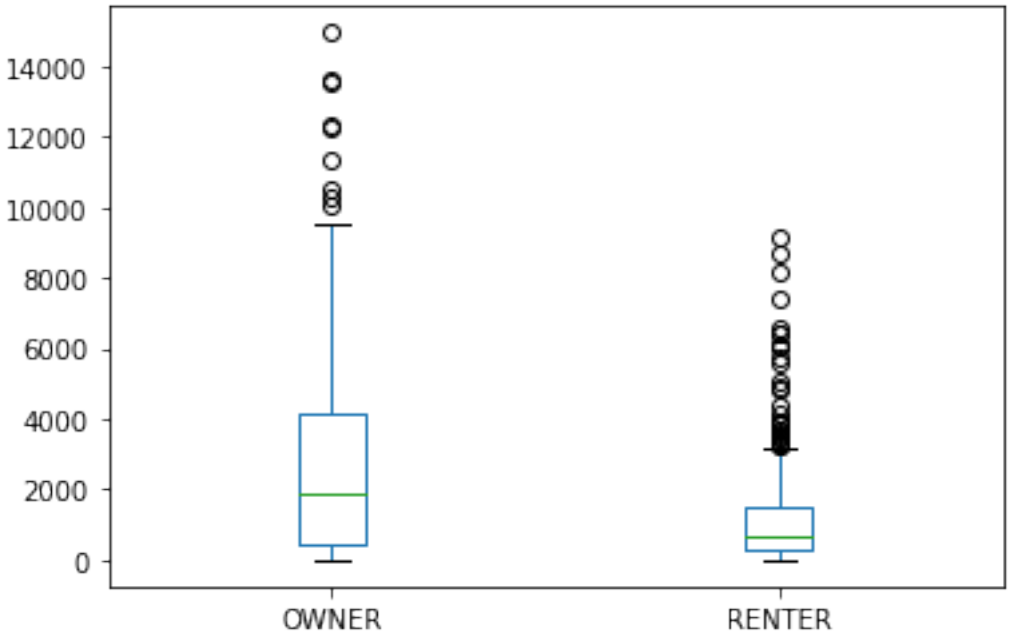
[39]:
```
<AxesSubplot:>
```



The graph above shows the distribution of renters and owners. There is an outlier in the graph that makes it harder to analyze. Let's cut some top outliers to see most of the data on the small end.

[40]:
```
owner_renter[owner_renter['RENTER'] < 10000].plot.box()
```

[40]: <AxesSubplot:>



Now it's better. We can see that the data is skewed to the smaller numbers. This is consistent with our maps on density and buildings that told us that most of the cities have a firly homogeneous population count in general. However, comparing owners and renters, the group of owners has more common value on the lower end than the group of renters does.

[41]:
```
cities_shp = gpd.read_file('maps/map cities.shp')
cities_shp = cities_shp.rename(columns={'TOWN':'CITY'})
cities_shp = cities_shp[['CITY', 'geometry']].merge(owner_renter, on='CITY',␣
 ↪how='left')
cities_shp.to_file("maps/renter cities.geojson", driver='GeoJSON')
```

[42]:
```
all_in_energy = (42.3497392, -71.1067746)
zoom = 9
renter_map = Map(center=all_in_energy, zoom=zoom)
geojson_data = json.load(open("maps/renter cities.geojson",'r'))
renter_data = values_to_bins(owner_renter['RENTER'], width=1, bins=10,␣
 ↪normalize=True)
maps_data = dict(zip(owner_renter['CITY'], renter_data))
for feature in geojson_data['features']:
    properties = feature['properties']
    if not properties['CITY'] in maps_data:
        maps_data[properties['CITY']] = 0
    feature.update(id=properties['CITY'])
renter_layer = Choropleth(
    geo_data=geojson_data,
    choro_data=maps_data,
    colormap=linear.YlOrRd_04,
    border_color='black',
    style={'fillOpacity': 1})
marker = Marker(location=all_in_energy, draggable=False)
html_renter = HTML('''Hover Over Cities''')
html_renter.layout.margin = '0px 20px 20px 20px'
control_renter = WidgetControl(widget=html_renter, position='topright')
renter_map.add_control(control_renter)
def update_html_renter(feature, **kwargs):
     html_renter.value = '''
     <h3><b>{}</b></h3>
     <h4>Renters: {} People</h4>
      '''.format(feature['properties']['CITY'],
        feature['properties']['RENTER'] if feature['properties']['RENTER'] else␣
 ↪-1.0)
renter_layer.on_hover(update_html_renter)
renter_map.add_layer(marker)
renter_map.add_layer(renter_layer)
```

[43]:
```
renter_map
```

```
Map(center=[42.3497392, -71.1067746], controls=(ZoomControl(options=['position', 'zoom_in_text
```

The renter map above shows that cities around Boston have pretty much the same number of renters. Again, this means we have a lot of options in picking surrounding cities to go with.

[44]:
```
most_renters = owner_renter[~buildings['CITY'].isin(visited)].
 ↪sort_values('RENTER', ascending=False).head(10)[['CITY', 'RENTER']]
most_renters
```

[44]:

	CITY	RENTER
340	WORCESTER	21630
20	BARNSTABLE	13656
273	SPRINGFIELD	11438
300	WALTHAM	10511
154	LOWELL	10000
73	DENNIS	9163
94	FALMOUTH	8663
170	MEDFORD	8177
343	YARMOUTH	7372
93	FALL RIVER	6543

2.7 D. Choosing The Best City

After we have the 10 best cities for each criterion, now we should be able to pick the best one. First, let's put all of them all into the same table based on each criterion.

[45]:
```
mosts = [most_buildings, most_dense, most_spanish, most_renters]
most_of_most = most_distance[['CITY']].rename(columns={'CITY':'Distance'})
for criteria in mosts:
    most_of_most[criteria.columns[1]] = criteria['CITY'].to_list()
most_of_most
```

[45]:

	Distance	1-4 Family	density	Spanish Speaker Population	\
266	SOMERVILLE	WORCESTER	SOMERVILLE	SPRINGFIELD	
45	BROOKLINE	SPRINGFIELD	EVERETT	LYNN	
170	MEDFORD	LOWELL	WINTHROP	WORCESTER	
276	STONEHAM	BARNSTABLE	REVERE	CHELSEA	
183	MILTON	NEWTON	CHELSEA	HOLYOKE	
336	WINCHESTER	NEW BEDFORD	LOWELL	LOWELL	
159	MALDEN	QUINCY	MALDEN	REVERE	
172	MELROSE	FALL RIVER	BELMONT	NEW BEDFORD	
238	READING	FALMOUTH	MARBLEHEAD	FRAMINGHAM	
91	EVERETT	LYNN	LYNN	EVERETT	

	RENTER

```
266    WORCESTER
45    BARNSTABLE
170  SPRINGFIELD
276      WALTHAM
183       LOWELL
336       DENNIS
159     FALMOUTH
172      MEDFORD
238     YARMOUTH
91    FALL RIVER
```

Unfortunately, there is no single city that is the best at every single criterion. So we need to come up with a way to find which city is the closest to being the best at every criterion. One way to do that is by giving each city a score. Below, I will give each city a score based on its positions in each list. For example, Somerville is the first in distance and density, so Somerville will have score 20.

[46]:
```
highest_buffer = {}
for col in most_of_most.columns:
    score = 10
    current_cities = most_of_most[col].values
    for city in current_cities:
        if not city in highest_buffer:
            highest_buffer[city] = 0
        highest_buffer[city] += score
        score -= 1
highest = []
for key,value in highest_buffer.items():
    highest.append([key, value])
highest_df = pd.DataFrame(highest, columns=['CITY', 'Score'])
highest_df.sort_values('Score', ascending=False)
```

[46]:

	CITY	Score
10	WORCESTER	28
11	SPRINGFIELD	27
12	LOWELL	24
0	SOMERVILLE	20
13	BARNSTABLE	16
22	CHELSEA	13
2	MEDFORD	11
21	REVERE	11
19	LYNN	11
9	EVERETT	11
1	BROOKLINE	9
20	WINTHROP	8
15	NEW BEDFORD	8
6	MALDEN	8
27	WALTHAM	7

3	STONEHAM	7
4	MILTON	6
18	FALMOUTH	6
14	NEWTON	6
25	HOLYOKE	6
5	WINCHESTER	5
28	DENNIS	5
16	QUINCY	4
17	FALL RIVER	4
7	MELROSE	3
23	BELMONT	3
8	READING	2
26	FRAMINGHAM	2
24	MARBLEHEAD	2
29	YARMOUTH	2

2.8 E. Conclusion

In today's world where data is a commodity, we can try to answer questions using data that is already there out in the world. For this particular example, we managed to answer the question "Which city is the best to expand to?" by using housing and demographics data from each city. It is Worcester. However, there is a problem. The way the score was constructed assumes that all of the criteria have equal importance. So, the next improvement should be finding out a way to differentiate the importance of criteria if needed.

Jimmy Hikmatullah Emerson College 2020

Appendix: Examination

3 Computer System Examination

This is C program that I wrote to start the Shell Lab.

```c
/*
 * tsh - A tiny shell program with job control
 *
 * Exam portion of Plan of Concentration.
 * Supervised by Jim Mahoney
 *
 * Jimmy Hikmatullah
 *
 * Emerson College
 * 2020
 */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <ctype.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <errno.h>

/* Misc manifest constants */
#define MAXLINE    1024   /* max line size */
#define MAXARGS     128   /* max args on a command line */
#define MAXJOBS      16   /* max jobs at any point in time */
#define MAXJID    1<<16   /* max job ID */

/* Job states */
#define UNDEF 0 /* undefined */
#define FG 1    /* running in foreground */
#define BG 2    /* running in background */
#define ST 3    /* stopped */

/*
 * Jobs states: FG (foreground), BG (background), ST (stopped)
 * Job state transitions and enabling actions:
 *     FG -> ST  : ctrl-z
 *     ST -> FG  : fg command
 *     ST -> BG  : bg command
 *     BG -> FG  : fg command
 * At most 1 job can be in the FG state.
 */

/* Global variables */
extern char **environ;      /* defined in libc */
char prompt[] = "tsh> ";    /* command line prompt (DO NOT
       CHANGE) */
int verbose = 0;            /* if true, print additional output
        */
int nextjid = 1;            /* next job ID to allocate */
char sbuf[MAXLINE];         /* for composing sprintf messages *
       /

struct job_t {              /* The job struct */
    pid_t pid;              /* job PID */
    int jid;                /* job ID [1, 2, ...] */
    int state;              /* UNDEF, BG, FG, or ST */
    char cmdline[MAXLINE];  /* command line */
};
struct job_t jobs[MAXJOBS]; /* The job list */
/* End global variables */

```

```
60
61 /* Function prototypes */
62
63 /* Here are the functions that you will implement */
64 void eval(char *cmdline);
65 int builtin_cmd(char **argv);
66 void do_bgfg(char **argv);
67 void waitfg(pid_t pid);
68
69 void sigchld_handler(int sig);
70 void sigtstp_handler(int sig);
71 void sigint_handler(int sig);
72
73 /* Here are helper routines that we've provided for you */
74 int parseline(const char *cmdline, char **argv);
75 void sigquit_handler(int sig);
76
77 void clearjob(struct job_t *job);
78 void initjobs(struct job_t *jobs);
79 int maxjid(struct job_t *jobs);
80 int addjob(struct job_t *jobs, pid_t pid, int state, char *
        cmdline);
81 int deletejob(struct job_t *jobs, pid_t pid);
82 pid_t fgpid(struct job_t *jobs);
83 struct job_t *getjobpid(struct job_t *jobs, pid_t pid);
84 struct job_t *getjobjid(struct job_t *jobs, int jid);
85 int pid2jid(pid_t pid);
86 void listjobs(struct job_t *jobs);
87
88 void usage(void);
89 void unix_error(char *msg);
90 void app_error(char *msg);
91 typedef void handler_t(int);
92 handler_t *Signal(int signum, handler_t *handler);
93
94 /*
95  * main - The shell's main routine
96  */
97 int main(int argc, char **argv){
98   char c;
99   char cmdline[MAXLINE];
100   int emit_prompt = 1; /* emit prompt (default) */
101
102 /* Redirect stderr to stdout (so that driver will get all
      output
103   * on the pipe connected to stdout) */
104 dup2(1, 2);
105
106 /* Parse the command line */
107 while ((c = getopt(argc, argv, "hvp")) != EOF) {
108     switch (c) {
109     case 'h':             /* print help message */
110         usage();
111   break;
112     case 'v':             /* emit additional diagnostic info
      */
113         verbose = 1;
114   break;
115     case 'p':             /* don't print a prompt */
116         emit_prompt = 0;  /* handy for automatic testing */
117   break;
118 default:
119           usage();
120 }
121   }
122
123   /* Install the signal handlers */
124
125   /* These are the ones you will need to implement */
126   Signal(SIGINT,  sigint_handler);   /* ctrl-c */
127   Signal(SIGTSTP, sigtstp_handler);  /* ctrl-z */
128   Signal(SIGCHLD, sigchld_handler);  /* Terminated or stopped
      child */
129
130   /* This one provides a clean way to kill the shell */
131   Signal(SIGQUIT, sigquit_handler);
132
133   /* Initialize the job list */
134   initjobs(jobs);
135
136   /* Execute the shell's read/eval loop */
137   while (1) {
138
139       /* Read command line */
140       if (emit_prompt) {
141           printf("%s", prompt);
142           fflush(stdout);
```

```
143         }
144         if ((fgets(cmdline, MAXLINE, stdin) == NULL) && ferror(
        stdin))
145             app_error("fgets error");
146         if (feof(stdin)) { /* End of file (ctrl-d) */
147             fflush(stdout);
148             exit(0);
149         }
150
151         /* Evaluate the command line */
152         eval(cmdline);
153         fflush(stdout);
154         fflush(stdout);
155     }
156
157     exit(0); /* control never reaches here */
158 }
159
160 /*
161  * eval - Evaluate the command line that the user has just
        typed in
162  *
163  * If the user has requested a built-in command (quit, jobs, bg
        or fg)
164  * then execute it immediately. Otherwise, fork a child process
        and
165  * run the job in the context of the child. If the job is
        running in
166  * the foreground, wait for it to terminate and then return.
        Note:
167  * each child process must have a unique process group ID so
        that our
168  * background children don't receive SIGINT (SIGTSTP) from the
        kernel
169  * when we type ctrl-c (ctrl-z) at the keyboard.
170 */
171 void eval(char *cmdline){
172   char *argv[MAXARGS];
173   int isBG = parseline(cmdline, argv);
174   if (!builtin_cmd(argv)) {
175     // Fork the command if it is not built-in command
176     pid_t newPid = fork();
177     if (newPid == 0) {
178       // child process runs here
179       if (execv(argv[0], argv) < 0)
180       printf("No Command Found! Try to add /bin/ \n");
181       exit(0);
182     }
183     int state = isBG ? BG : FG;
184     addjob(jobs, newPid, state, cmdline);
185     if (!isBG) {
186       // if foreground process, wait until it finishes.
187       wait(NULL);
188     }
189     return;
190   }
191   return;
192 }
193
194 /*
195  * parseline - Parse the command line and build the argv array.
196  *
197  * Characters enclosed in single quotes are treated as a single
198  * argument.  Return true if the user has requested a BG job,
        false if
199  * the user has requested a FG job.
200  */
201 int parseline(const char *cmdline, char **argv)
202 {
203     static char array[MAXLINE]; /* holds local copy of command
        line */
204     char *buf = array;          /* ptr that traverses command
        line */
205     char *delim;                /* points to first space
        delimiter */
206     int argc;                   /* number of args */
207     int bg;                     /* background job? */
208
209     strcpy(buf, cmdline);
210     buf[strlen(buf)-1] = ' ';  /* replace trailing '\n' with
        space */
211     while (*buf && (*buf == ' ')) /* ignore leading spaces */
212   buf++;
213
214     /* Build the argv list */
215     argc = 0;
216     if (*buf == '\'') {
217   buf++;
```

```
218   delim = strchr(buf, '\'');
219     }
220     else {
221   delim = strchr(buf, ' ');
222     }
223
224     while (delim) {
225   argv[argc++] = buf;
226   *delim = '\0';
227   buf = delim + 1;
228   while (*buf && (*buf == ' ')) /* ignore spaces */
229           buf++;
230
231   if (*buf == '\'') {
232       buf++;
233       delim = strchr(buf, '\'');
234   }
235   else {
236       delim = strchr(buf, ' ');
237   }
238     }
239     argv[argc] = NULL;
240
241     if (argc == 0)  /* ignore blank line */
242   return 1;
243
244     /* should the job run in the background? */
245     if ((bg = (*argv[argc-1] == '&')) != 0) {
246   argv[--argc] = NULL;
247     }
248     return bg;
249 }
250
251 /*
252  * builtin_cmd - If the user has typed a built-in command then
        execute
253  *     it immediately.
254  */
255 int builtin_cmd(char **argv)
256 {
257   if (strcmp("quit", argv[0]) == 0) {
258     exit(0);
259   } else if (strcmp("jobs", argv[0]) == 0) {
260     listjobs(jobs);
261     return 1;
262   }
263   return 0;     /* not a builtin command */
264 }
265
266 /*
267  * do_bgfg - Execute the builtin bg and fg commands
268  */
269 void do_bgfg(char **argv)
270 {
271   if (strcmp(argv[0], "bg") == 0) {
272     // execute bg function
273   }
274 }
275
276 /*
277  * waitfg - Block until process pid is no longer the foreground
         process
278  */
279 void waitfg(pid_t pid)
280 {
281     return;
282 }
283
284 /*****************
285  * Signal handlers
286  *****************/
287
288 /*
289  * sigchld_handler - The kernel sends a SIGCHLD to the shell
        whenever
290  *     a child job terminates (becomes a zombie), or stops
        because it
291  *     received a SIGSTOP or SIGTSTP signal. The handler reaps
        all
292  *     available zombie children, but doesn't wait for any
        other
293  *     currently running children to terminate.
294  */
295 void sigchld_handler(int sig)
296 {
297   return;
298 }
299
```

```
300 /*
301  * sigint_handler - The kernel sends a SIGINT to the shell
         whenver the
302  *     user types ctrl-c at the keyboard.  Catch it and send it
         along
303  *     to the foreground job.
304  */
305 void sigint_handler(int sig)
306 {
307   pid_t currentPid = fgpid(jobs);
308   if (!currentPid) {
309     exit(0);
310   }
311   deletejob(jobs, currentPid);
312   kill(currentPid, sig);
313   return;
314 }
315
316 /*
317  * sigtstp_handler - The kernel sends a SIGTSTP to the shell
         whenever
318  *      the user types ctrl-z at the keyboard. Catch it and
         suspend the
319  *      foreground job by sending it a SIGTSTP.
320  */
321 void sigtstp_handler(int sig)
322 {
323   pid_t currentPid = fgpid(jobs);
324   if (!currentPid) {
325     exit(0);
326   }
327   deletejob(jobs, currentPid);
328   kill(currentPid, sig);
329   return;
330 }
331
332 /*********************
333  * End signal handlers
334  *********************/
335
336 /***********************************************
337  * Helper routines that manipulate the job list
338  **********************************************/
339
340 /* clearjob - Clear the entries in a job struct */
341 void clearjob(struct job_t *job) {
342     job->pid = 0;
343     job->jid = 0;
344     job->state = UNDEF;
345     job->cmdline[0] = '\0';
346 }
347
348 /* initjobs - Initialize the job list */
349 void initjobs(struct job_t *jobs) {
350     int i;
351
352     for (i = 0; i < MAXJOBS; i++)
353   clearjob(&jobs[i]);
354 }
355
356 /* maxjid - Returns largest allocated job ID */
357 int maxjid(struct job_t *jobs)
358 {
359     int i, max=0;
360
361     for (i = 0; i < MAXJOBS; i++)
362   if (jobs[i].jid > max)
363       max = jobs[i].jid;
364     return max;
365 }
366
367 /* addjob - Add a job to the job list */
368 int addjob(struct job_t *jobs, pid_t pid, int state, char *
         cmdline)
369 {
370     int i;
371
372     if (pid < 1)
373   return 0;
374
375     for (i = 0; i < MAXJOBS; i++) {
376   if (jobs[i].pid == 0) {
377       jobs[i].pid = pid;
378       jobs[i].state = state;
379       jobs[i].jid = nextjid++;
380       if (nextjid > MAXJOBS)
381     nextjid = 1;
382       strcpy(jobs[i].cmdline, cmdline);
```

```
383         if(verbose){
384           printf("Added job [%d] %d %s\n", jobs[i].jid, jobs[i
        ].pid, jobs[i].cmdline);
385             }
386             return 1;
387   }
388     }
389     printf("Tried to create too many jobs\n");
390     return 0;
391 }
392
393 /* deletejob - Delete a job whose PID=pid from the job list */
394 int deletejob(struct job_t *jobs, pid_t pid)
395 {
396     int i;
397
398     if (pid < 1)
399   return 0;
400
401     for (i = 0; i < MAXJOBS; i++) {
402   if (jobs[i].pid == pid) {
403       clearjob(&jobs[i]);
404       nextjid = maxjid(jobs)+1;
405       return 1;
406   }
407     }
408     return 0;
409 }
410
411 /* fgpid - Return PID of current foreground job, 0 if no such
        job */
412 pid_t fgpid(struct job_t *jobs) {
413     int i;
414
415     for (i = 0; i < MAXJOBS; i++)
416   if (jobs[i].state == FG)
417       return jobs[i].pid;
418     return 0;
419 }
420
421 /* getjobpid  - Find a job (by PID) on the job list */
422 struct job_t *getjobpid(struct job_t *jobs, pid_t pid) {
423     int i;
424
425     if (pid < 1)
426   return NULL;
427     for (i = 0; i < MAXJOBS; i++)
428   if (jobs[i].pid == pid)
429       return &jobs[i];
430     return NULL;
431 }
432
433 /* getjobjid  - Find a job (by JID) on the job list */
434 struct job_t *getjobjid(struct job_t *jobs, int jid)
435 {
436     int i;
437
438     if (jid < 1)
439   return NULL;
440     for (i = 0; i < MAXJOBS; i++)
441   if (jobs[i].jid == jid)
442       return &jobs[i];
443     return NULL;
444 }
445
446 /* pid2jid - Map process ID to job ID */
447 int pid2jid(pid_t pid)
448 {
449     int i;
450
451     if (pid < 1)
452   return 0;
453     for (i = 0; i < MAXJOBS; i++)
454   if (jobs[i].pid == pid) {
455             return jobs[i].jid;
456         }
457     return 0;
458 }
459
460 /* listjobs - Print the job list */
461 void listjobs(struct job_t *jobs)
462 {
463     int i;
464
465     for (i = 0; i < MAXJOBS; i++) {
466   if (jobs[i].pid != 0) {
467       printf("[%d] (%d) ", jobs[i].jid, jobs[i].pid);
468       switch (jobs[i].state) {
```

```
469     case BG:
470         printf("Running ");
471         break;
472     case FG:
473         printf("Foreground ");
474         break;
475     case ST:
476         printf("Stopped ");
477         break;
478       default:
479         printf("listjobs: Internal error: job[%d].state=%d ",
480          i, jobs[i].state);
481       }
482       printf("%s", jobs[i].cmdline);
483   }
484     }
485 }
486 /*****************************
487  * end job list helper routines
488  *****************************/
489
490
491 /************************
492  * Other helper routines
493  ************************/
494
495 /*
496  * usage - print a help message
497  */
498 void usage(void)
499 {
500     printf("Usage: shell [-hvp]\n");
501     printf("   -h   print this message\n");
502     printf("   -v   print additional diagnostic information\n")
        ;
503     printf("   -p   do not emit a command prompt\n");
504     exit(1);
505 }
506
507 /*
508  * unix_error - unix-style error routine
509  */
510 void unix_error(char *msg)
511 {
512     fprintf(stdout, "%s: %s\n", msg, strerror(errno));
513     exit(1);
514 }
515
516 /*
517  * app_error - application-style error routine
518  */
519 void app_error(char *msg)
520 {
521     fprintf(stdout, "%s\n", msg);
522     exit(1);
523 }
524
525 /*
526  * Signal - wrapper for the sigaction function
527  */
528 handler_t *Signal(int signum, handler_t *handler)
529 {
530     struct sigaction action, old_action;
531
532     action.sa_handler = handler;
533     sigemptyset(&action.sa_mask); /* block sigs of type being
          handled */
534     action.sa_flags = SA_RESTART; /* restart syscalls if
          possible */
535
536     if (sigaction(signum, &action, &old_action) < 0)
537   unix_error("Signal error");
538     return (old_action.sa_handler);
539 }
540
541 /*
542  * sigquit_handler - The driver program can gracefully
          terminate the
543  *    child shell by sending it a SIGQUIT signal.
544  */
545 void sigquit_handler(int sig)
546 {
547     printf("Terminating after receipt of SIGQUIT signal\n");
548     exit(1);
549 }
```

4 Algorithm Examination

This is Jupyter Notebook that I used to demonstrate the algorithm exam solutions.

Algorithm exam

November 23, 2020

1 Algorithm Exam

As part of Plan of Concentration, this algorithm exam is part of the independent. The purpose of the exam is to show my understanding in the topics.

Jimmy Hikmatullah Supervised by Jim Mahoney from Bennington College Emerson College 2020

1. Implement, numerically test, and explain two different sorting algorithms with different O() behaviors on randomly chosen lists of numbers with various sizes n. Use two different programming languages and coding styles. Show graphically that the expecected performance is consistent with your numerical experiments.

[1]:
```
class Extended_list(list):
    '''
        Extending list built-in variable type with counting sort built into it.
    '''
    def __init__(self, input_list):
        list.__init__(self, input_list)
        self.aux_list = [0] * (max(input_list) + 1)
        self.steps = 0
        self.space = len(self.aux_list) + len(input_list)

    def counting_sort(self):
        '''
            Do in-place sort using counting sort algorithm.
        '''
        copy = self[:]
        for element in self:
            self.aux_list[element] += 1
            self.steps += 1
        for i in range(1, len(self.aux_list)):
            self.aux_list[i] = self.aux_list[i - 1] + self.aux_list[i]
            self.steps += 1
        for element in copy:
            self[self.aux_list[element] - 1] = element
            self.aux_list[element] -= 1
            self.steps += 1
```

[2]:
```
# Counting Sort test
test = Extended_list([4,5,7,8,1,2,1,3,4,2,77,64,32])
test
```

[2]:
```
[4, 5, 7, 8, 1, 2, 1, 3, 4, 2, 77, 64, 32]
```

[3]:
```
test.counting_sort()
test
```

[3]:
```
[1, 1, 2, 2, 3, 4, 4, 5, 7, 8, 32, 64, 77]
```

[4]:
```
from json import load
```

[5]:
```
# Load generated unsorted lists
with open('input.json') as json_file:
    data = load(json_file)
```

[6]:
```
# Apply new counting sort method on test file
results = []
for data_point in data:
    result = [max(data_point), len(data_point)]
    array = Extended_list(data_point)
    array.counting_sort()
    result.append(array.steps)
    result.append(array.space)
    results.append(result)
```

[7]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

[8]:
```
countingData = pd.DataFrame(results, columns=['Range', 'Length','Steps',␣
↪'Space'])
countingData
```

[8]:

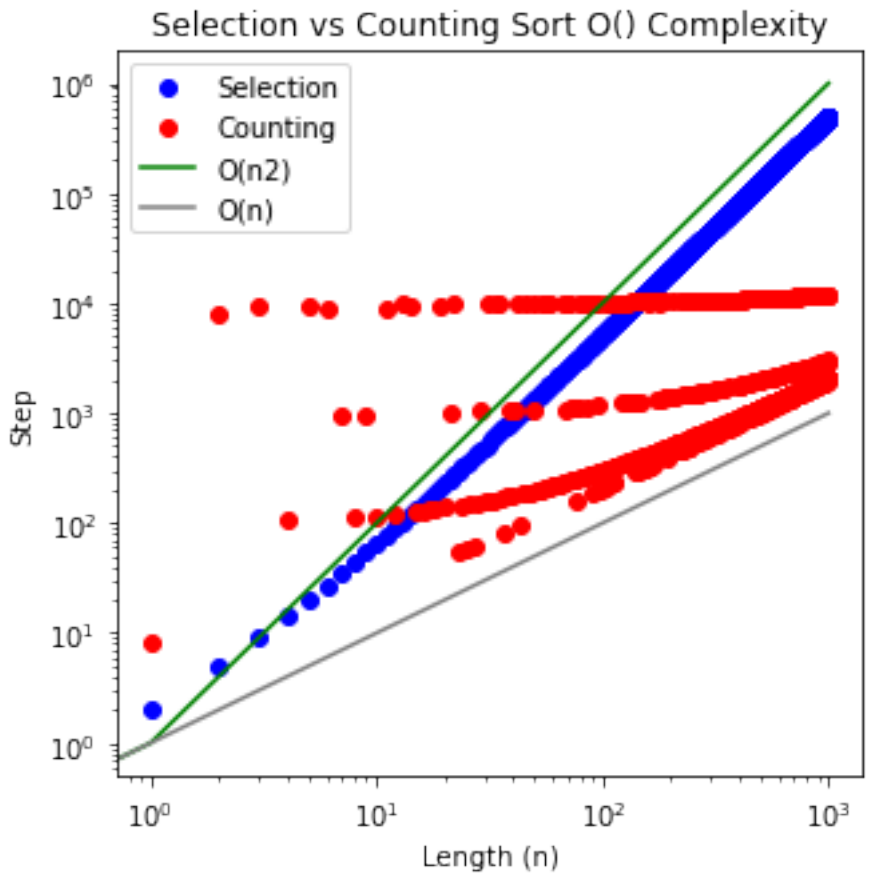
	Range	Length	Steps	Space
0	6	1	8	8
1	7870	2	7874	7873
2	9590	3	9596	9594
3	99	4	107	104
4	9642	5	9652	9648
..	…	…	…	…
994	9	995	1999	1005
995	99	996	2091	1096
996	997	997	2991	1995
997	99	998	2095	1098
998	9	999	2007	1009

```
[999 rows x 4 columns]
```

[9]:
```
# Load Performance Metrics from JavaScript
selectionData = pd.read_json('output.json')
selectionData = selectionData.rename(columns={0: 'Range', 1: 'Length', 2:␣
↪'Steps', 3: 'Space'})
```

[10]:
```
fig = plt.figure(figsize=(5, 5))
ax = plt.gca()
ax.figure = fig
ax.plot(selectionData['Length'], selectionData['Steps'], 'o', c='blue',␣
↪label='Selection')
ax.plot(countingData['Length'], countingData['Steps'], 'o', c='red',␣
↪label='Counting')
nsquared = np.linspace(0, max(selectionData['Length']),␣
↪len(selectionData['Length']))
ynsquared = nsquared ** 2
ax.plot(nsquared, ynsquared, '-', c='green', label='O(n2)')
ax.plot(nsquared, nsquared, '-', c='gray', label='O(n)')
ax.set_yscale('log')
ax.set_xscale('log')
ax.set_xlabel('Length (n)')
ax.set_ylabel('Step')
ax.legend()
ax.set_title('Selection vs Counting Sort O() Complexity')
```

[10]:
```
Text(0.5, 1.0, 'Selection vs Counting Sort O() Complexity')
```

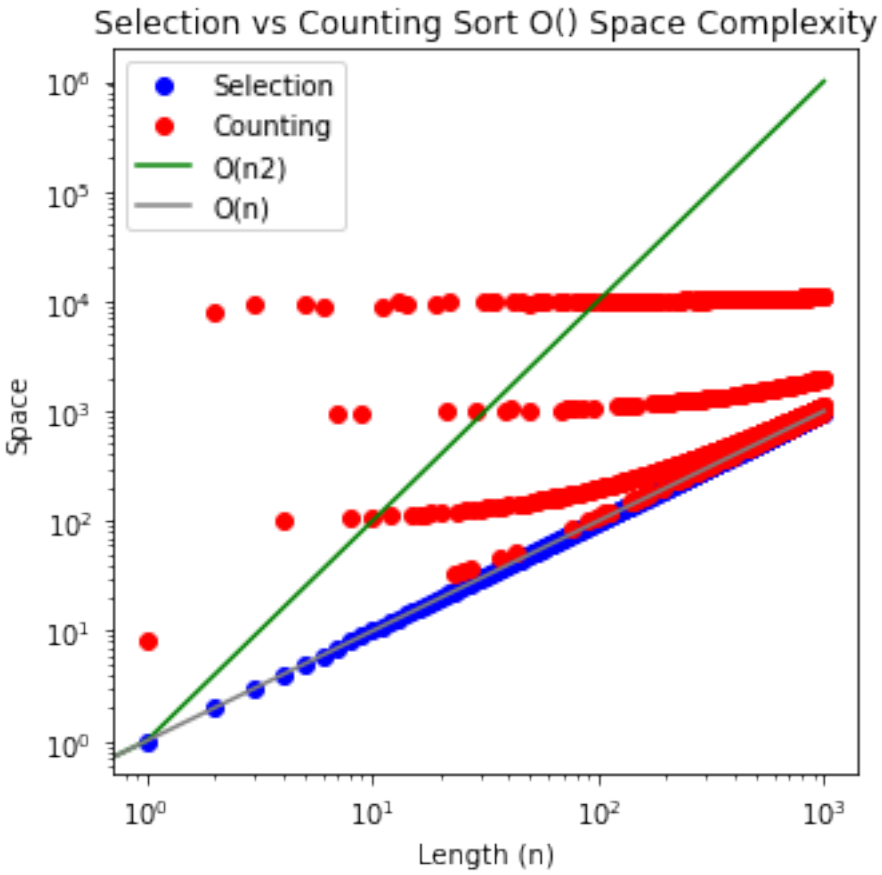


The graph shows both selection and counting performance. The counting sort is more superior than selection sort since counting's complexity is O(n k), where the k is the range of numbers in the list. Specifically when sorting a really long sequence of numbers, like 100000 numbers, the counting sort will have less complexity than selection sort. However it comes with caveats.

[11]:
```
fig = plt.figure(figsize=(5, 5))
ax = plt.gca()
ax.figure = fig
ax.plot(selectionData['Length'], selectionData['Space'], 'o', c='blue',␣
↪label='Selection')
ax.plot(countingData['Length'], countingData['Space'], 'o', c='red',␣
↪label='Counting')
ax.plot(nsquared, ynsquared, '-', c='green', label='O(n2)')
ax.plot(nsquared, nsquared, '-', c='gray', label='O(n)')
ax.set_yscale('log')
ax.set_xscale('log')
ax.set_xlabel('Length (n)')
ax.set_ylabel('Space')
ax.legend()
```

```
ax.set_title('Selection vs Counting Sort O() Space Complexity')
```

[11]: `Text(0.5, 1.0, 'Selection vs Counting Sort O() Space Complexity')`



This graph shows the caveats. Although counting sort takes less steps to sort the list, it takes more space than selection sort. Specifically, when the range of the numbers in the sequence is large. The space complexity for counting sort is O(n k).

2. In a language of your choice, code and explain both a depth first and breadth first search algorithm of a tree of possible moves for a small "sliding block" puzzle. (Using a stack and queue would be a good approach.)

   A sample search might be to get from

   ```
   start        finish

   2 1 3        1 2 3
   5 4 6        4 5 6
   7 8 .        7 8 .
   ```

   where the "." is the empty square and the two possible first moves slide either the 6 or the 8 to bottom right corner.

[12]:
```
from math import sqrt
class Board:
    '''
        Class of board which can be moved and store the current state of the␣
↪board.
    '''
    def __init__(self, init_state):
        self.hole = max(init_state)
        self.size = sqrt(len(init_state))
        self.state = init_state[:]
        self.hole_loc = self.find_hole()
        self.moves_mapper = {
            'up': (0, -1),
            'down': (0, 1),
            'right': (1, 0),
            'left': (-1, 0)
        }

    def all_moves(self):
        '''
            Return all possible move in any given position of the empty square.
        '''
        possible_move = []
        xhole, yhole = self.find_hole_coor()
        for move, coor in self.moves_mapper.items():
            if (0 <= (xhole + coor[0]) <= self.size - 1) and (0 <= (yhole +␣
↪coor[1]) <= self.size - 1):
                possible_move.append(move)
        return possible_move

    def move(self, direction):
        '''
            Change the state of the board based on the input direction
        '''
        if direction in self.moves_mapper:
            coor = self.moves_mapper[direction]
            xhole, yhole = self.find_hole_coor()
            if (0 <= (xhole + coor[0]) <= self.size - 1) and (0 <= (yhole +␣
↪coor[1]) <= self.size - 1):
                future_loc = self.hole_loc + coor[0]
                future_loc = future_loc + int(coor[1] * self.size)
                swap = self.state[future_loc]
                self.state[future_loc] = self.hole
                self.state[self.hole_loc] = swap
                self.hole_loc = future_loc
            else:
                print('Impossible Move!')
```

```
        else:
            print('Invalid Move')

    def compare(self, solved_state):
        '''
            Compare 2 states of the board.

            Return true if the same.
        '''
        return ''.join(str(element) for element in self.state) == ''.
↪join(str(element) for element in solved_state)

    def find_hole(self):
        '''
            Return the index of the hole.
        '''
        for i in range(len(self.state)):
            if self.state[i] == self.hole:
                return i
    def find_hole_coor(self):
        '''
            Return the x and y coordinate based on the hole index.
        '''
        xhole = self.hole_loc % self.size
        yhole = self.hole_loc // self.size
        return xhole, yhole

    def print_board(self):
        '''
            Print the board using regular print() function.
        '''
        for i in range(len(self.state)):
            endline = '\n' if (i % self.size) - 1 == 1 else ' '
            print(self.state[i] if not self.state[i] == self.hole else '.' ,␣
↪end=endline)
        print('\n')
```

[13]:
```
class State:
    '''
        Class of 'node' which can be used to store the history of
        moves done by a specific starting position.
    '''
    def __init__(self, state, moves=[]):
        self.moves = moves
        self.board = Board(state)
```

[14]:
```
class Searches:
    def __init__(self, start, end, style='depth'):
        self.memory = set()
        self.end = end
        self.counter = 0
        self.buffer = [State(start)]
        self.style = 0 if style == 'depth' else -1

    def search(self):
        '''
            Perfome a search on a given starting state and stop when the end␣
↪state is reached.
        '''
        while self.counter < 100000 and len(self.buffer) > 0:
            current_state = self.buffer.pop(self.style)
            hashable = tuple(current_state.board.state)
            if hashable in self.memory:
                continue
            self.memory.add(hashable)
            for move in current_state.board.all_moves():
                new_state = State(current_state.board.state, current_state.
↪moves[:])
                new_state.board.move(move)
                new_state.moves.append(move)
                if new_state.board.compare(self.end):
                    return new_state
                self.buffer.append(new_state)
            self.counter += 1
        sorted_buffer = sorted(self.buffer, key=lambda x: len(x.moves))
        return sorted_buffer[-1]
```

[15]:
```
start = [
    2,1,3,
    5,4,6,
    7,8,9
]
end = [
    1,2,3,
    4,5,6,
    7,8,9
]
```

[16]:
```
depth_search = Searches(start, end)
breadth_search = Searches(start, end, 'breadth')
search_moves = {
    'breadth': breadth_search.search(),
    'depth': depth_search.search()
```

```
}
```

[17]:
```
for style, states in search_moves.items():
    print('Final State using {}'.format(style))
    print('Moves Length: {} moves'.format(len(states.moves)), end='\n\n')
    states.board.print_board()
```

```
Final State using breadth
Moves Length: 93485 moves

2 4 3
6 8 5
1 . 7


Final State using depth
Moves Length: 16 moves

1 2 3
4 5 6
7 8 .
```

3. Explain what a "hash table" is, and what it's O() behavior looks like. Implement one, use it to make a histogram of a word counts in a large text file, and discuss its performance. The specifics (collision algorithm, hash function, programming language) are all up to you.

[18]:
```
class Hash_Table:
    '''
        Hash table implemention using built-in list data structure.
    '''
    def __init__(self, init_size=10):
        '''
            Initialize the table.
        '''
        self.table = [None] * init_size
        self.filled = 0

    def __setitem__(self, key, value):
        '''
            Set value by given key using built in hash function.
        '''
        hashed = hash(str(key))
        index = hashed % len(self.table)
        if self.table[index]:
            if self.table[index][0] == key:
                self.table[index][1] = value
```

```
                return None
        hashed = hash(str(hashed))
        while self.table[index] != None:
            self.expand()
            index = hashed % len(self.table)
        self.table[index] = [key, value]
        self.filled += 1
        if  self.filled / len(self.table) > 0.2:
            self.expand()

    def expand(self):
        '''
            In order to avoid collision, expand the table 10x bigger.
        '''
        values = list(filter(lambda x: x != None, self.table))
        self.table = [None] * (len(self.table) * 10)
        for value in values:
            hashed = hash(str(value[0]) )
            index = hashed % len(self.table)
            if self.table[index] != None:
                hashed = hash(str(hashed))
                index = hashed % len(self.table)
            if self.table[index] != None:
                print('Collision')
            self.table[index] = [value[0], value[1]]


    def __getitem__(self, key):
        '''
            Return value based on key.
        '''
        hashed = hash(str(key))
        index = hashed % len(self.table)
        try:
            if self.table[index][0] == key:
                return self.table[index][1]
            hashed = hash(str(hashed))
            index = hashed % len(self.table)
            if self.table[index][0] == key:
                return self.table[index][1]
            print('Hash Error')
        except TypeError:
            return None

    def __contains__(self, key):
        '''
            Perform special 'in' logic.
```

```
        '''
        hashed = hash(str(key))
        index = hashed % len(self.table)
        if self.table[index]:
            if self.table[index][0] == key:
                return True
            else:
                hashed = hash(str(hashed))
                index = hashed % len(self.table)
                if self.table[index]:
                    if self.table[index][0] == key:
                        return True
        return False

    def sort(self, reverse=False):
        '''
            Return a sorted list based on value.
        '''
        values = list(filter(lambda x: x != None, self.table))
        return sorted(values, key=lambda x: x[1], reverse=reverse)
```

Load Pride and Prejudice file. Downloaded from Project Gutenberg.

[19]:
```
import re
from datetime import datetime
with open('book.txt') as file:
    book = file.read().replace('\n', ' ')
    book = re.sub(r'[^\w\s]', ' ', book)
    book = book.upper()
```

[20]:
```
start_time = datetime.now()
striped = book.split(' ')
counter = Hash_Table()
for word in striped:
    if len(word) < 1:
        continue
    try:
        if not word in counter:
            counter[word] = 0
        counter[word] += 1
    except TypeError:
        print(word)
end_time_implementation = (datetime.now() - start_time).total_seconds()
```

```
AT
UNDER
```

```
WHAT
DISGUST
PROBABILITY
IDLE
```

[21]:
```
start_time = datetime.now()
counter_built_in = {}
for word in striped:
    if len(word) < 1:
        continue
    try:
        if not word in counter_built_in:
            counter_built_in[word] = 0
        counter_built_in[word] += 1
    except TypeError:
        print(counter_built_in[word])
        print(word)
end_time_built_in = (datetime.now() - start_time).total_seconds()
```

[22]:
```
print('Total seconds using implementation code: {} seconds'.
 ↪format(end_time_implementation))
print('Total seconds using built-in code: {} seconds'.format(end_time_built_in))
```

```
Total seconds using implementation code: 2.681599 seconds
Total seconds using built-in code: 0.07709 seconds
```

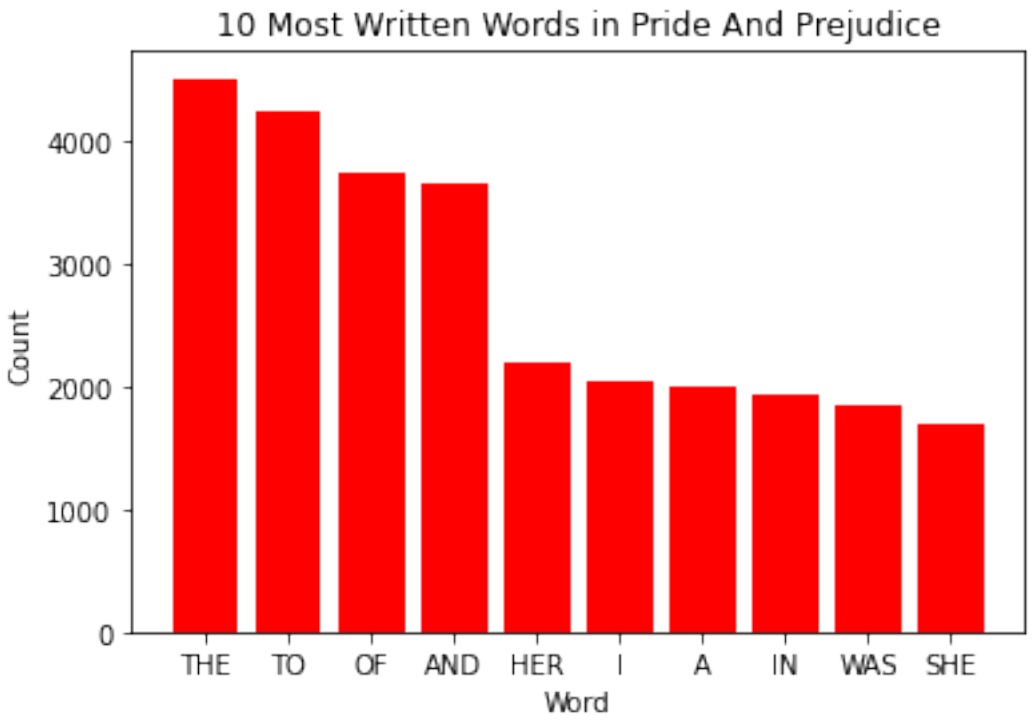
[23]:
```
words = pd.DataFrame(counter.sort(reverse=True), columns=['Word', 'Count'])
```

[24]:
```
x = words['Word'].head(10)
energy = words['Count'].head(10)

x_pos = [i for i, _ in enumerate(x)]

plt.bar(x_pos, energy, color='red')
plt.xlabel("Word")
plt.ylabel("Count")
plt.title("10 Most Written Words in Pride And Prejudice")
plt.xticks(x_pos, x)

plt.show()
```


10 Most Written Words in Pride And Prejudice
Count
4000
3000
2000
1000
0
THE
TO
OF
AND
HER
I
A
IN
WAS
SHE
Word