# From Social Science to what?

Ask a dozen data scientists to define the field and you’ll get at least 13 answers, 12 from the scientists themelves and one from the cheeky scientist who classifies all those answers and comes up with a meta-answer. It’s a new field and frankly, it is not clear whether it is a field in its own right or a transdisciplinary practice meant to inform existing fields. One thing is for sure, though: people produce a lot of data.

Unfortunately, defining social science is probably just as hard as defining data science. Clearly sociology and political science belong in there, but what about linguistics? Psychology? Ok, maybe social psychology? It turns out, articulating the boundaries around specific scientific practices is itself a recipe for contention. Personally, I like to say I’m promiscuous with theories and conservative with claims. We can, and should, find our ideas from a variety of sources all around us, but our claims about the world should be much more cautious. Within this context, I see social science as any practice that makes claims about groups of individuals *that could be* testable or falsifiable under clearly defined conditions. This is not to say that these conditions will be met. In fact, sometimes, it is pretty impossible to meet those conditions in real life with humans, because humans are clever and meddlesome creatures but also because researchers aspire to be ethical and understand that we are both, in the grand scheme of things, researchers and potential subjects.

We must deal not only with insights from our data but feedback from the population about which such insights are formed. Imagine we download tweets for a few days and then do a modest data analysis on these tweets. We chunk them into hourly bins. So all the tweets from 7:00-7:59 get an average score for their sentiment, the one’s from 8:00-8:59 represent another average sentiment score. We do this for a long time and discover that, on average, the tweets with the happiest sentiment are in the morning. So far, so good. We write them up, publlish them, and make headlines. All of a sudden people start tweeting more in the morning because it seesm that morning tweets are related to happiness. But maybe it was just people who are well-rested or people who have a job to get up for? All of a sudden, the interest in our finding changed the nature of the population (albeit only ever so slightly) in such a way that our conclusion no longer holds the way it did.

This is where the cold analyitc findings of data science meet the twisty, cautious, feedback loop world of social science. And now, under the right conditions the two can be mutually informative. We are not merely analyzing a world created by others, but informing that world thorugh measuring different parts of it and reporting those measurements off to some people. Soemtimes we are reporting to the public, but more often we are reporting to decision makers with a need to have a question answered. xx.ramble()

This book is not meant to be an encyclopaedic reference for data sceince and nor is it meant to be a cutting edge book in the latest practices. It’s meant to help you learn and do. For this, it means we have to take things step-by-step with few assumptions and a lot of patience. Two things we want to avoid in this book: information overload and impostor syndrome.

Information overload is an experience of having more information available than can be processed for the current task. It’s like having a menu with 40 pages of entrees, many of which have names or ingredients you have never heard of. We want to avoid introducing too many concepts before being able to do something.

Related to information overload is imposter syndrome. Imposter symdrome takes ahold when people, typically students, judge themselves against the accomplishments of others (often unfairly) and believe that they simply do not measure up or should not be in the program there are in. First, as someone who has done admissions at my university a few times, let’s just say that if you were given an offer, you fit in. Second, the loudest student is not necessarily the most competent (and often they are not). I raise imposter syndrome here because when you’re given too much information it is often the case that you will think it is you, the reader, who has the problem. Actually, it is probably the writer. Let’s face it, on first reading virtually no one will understand or remember everything. Yet, some people set up really high expectations or believe they should be able to get it immediately. Then when they don’t, some peopel feel that maybe they do not belong. Let’s say this right now: you belong. If you can read this book, you can code. Do not let the haters and the show offs get to you.

Often times, people will focus on the skills they are strongest at to the deteriment of skills they often need. If I had a dollar for every pretty but uninterpreted data visualisation I’ve seen I could pay someone else to write this book. Fortunately, social scientists are often very skilled with interpretation. We focus on unintended consequences, externalities, biases, historical artefacts, spurious correlations, and many, many contingencies behind the data. The social scientist is thus often playing the part of a skeptic rather than a sage. This makes social scientists really great candidates for learning data science techniques –they are interested almost by nature in interpretation– and weary of cavalier interpretations regardless of how impressive the data was.

This skepticism makes for a great data scientist because it makes every step of the process a satisfying puzzle. Whether it is how to count the number of sentences in a text or how to detect polarisation in conversations, each step has to be done with care and caution. Further, when done right each step can be a source of interest and insight.

In this book we weave in this concern for care with practical skills in programming. The result, I hope, is a book you’ll use and return to when asking and answering questions important to you and your research. By the end you will have an impressive ensemble of data science skills and hopefully a clear sense both of how do to use these skills and also how to interpret the insights that come from applying data science skills to social science data.

# (PO)DIKW - A potential theoretical framework for Data Science

Social science textbooks often start with some of the interesting findings in the field, the key questions facing social scientists or some controversey to be resolved. Along the way, researchers tend to introduce some of the main theories or tenets of the field. If you’re learning sociology, you will hear about Max Weber and rationality or Durkheim and social facts. In Communication, you’re likely to hear about medium theory and McLuhan or perhaps Lazersfeld and Two-Step flow of communication. In economics we hear about Pareto and diminishing utility. But what about data science? What frameworks can we use to guide us here?

To my knowledge, there’s no specific core tenets of data science. We have key thinkers and papers, though these also tend to come from existing domains. And if you are to look for the key papers in data science these end up tending towards being really papers in statistics, machine learning, sociology or econometrics. Yet, when we think of data sceince as a science, it seems that there must be a basis for this? So below is an attempt to consider data science, and particlarly the notion of ‘social data science’ from a theoretical perspective.

Below I draw upon an existing framework from information visualisation called DIKW, or Data - Information - Knowledge - Wisdom. Yet, I preface this with PO to stand for Phenomena and Operationalisation. This is because the world is not filled with data. It is filled with phenomena, which we convert to data through operationalisation. Then once operationalised through measurement or encoding, we can see how it first becomes data and then serves as the basis for information, knowledge, and ultimately, wisdom.

## What is data?

It turns out that it is even pretty difficult to define data. Originally it meant that which is given or self-evident, from the latin *datum*. In 1946 it was first used to refer to transmissable and storable computer information and less than a decade later was the first use of ‘data processing’. Data is a plural but is often written as a singular (which it is here). This is because data is seen as a mass noun, like information or rice. We would say the ‘the rice is cooked’, by which we almost never mean a single specific grain of rice. Similarly ‘the data is being processed’ would not refer to a single measurement. While data is a mass noun, we can make it a particular by referring to it as a set. Thus we can say “data is reliable” and “data sets are reliable”.

A 21st Century data science should certainly not take data as a given. We have had over a Century of critical thought that should remind people how much goes into any given measurement. For example, let’s say we want to measure a frequency of tweeting by gender. Easy - count all the male names and the female names. Or wait, what about androgynous names like ‘Alex’ or ‘Lee’? What about accounts that don’t list names? What about group accounts? Even the concept of gender itself is not as easy as considering a binary. Animals can be hermaphrodites and switch genders under some conditions, like differing water temperature or crowding (for some frog species). Individual people can be transgender or present as ‘non-binary’. People can be ‘assigned male at birth’ but definitely feel female. Some people can have extra sex chromasomes or be ‘intersex’ with some of both male and female organs. All of a sudden, what we thought was a simple binary distinction turns out to be anything but. Adding on top of that is the matter of how closely gender follows sex. Some people believe that gender is determined by sex, whereas many others think that biological sex tends to nudge people towards one gender or another but history and culture do much more to determine it.

Right from the get go we are confronted with the first issue in data science, even before we get to data: How to operationalise? Some say the world is full of data. While it is true that the world contains huge amounts of data, the world *is not reducible to data*. Rather, everything we observe is **phenomena**. Everything just is. This perspective comes from William James’ notion of radical empiricism, or the idea that everything is real. Dreams are real, myths are real, fantasies are real. If you can think of it, it is real for it has been a thought, even if it has been nothing else. On the other hand, the substance of things might not be what they seem. I am not saying a dragon in a dream is an actual winged scaled lizard. But it definitely is an impression created by the mind that the observing mind recognizes as a dragon. James likes to use the notion of a stick in the water. At the surface of the water, the stick might appear bent. Is it? Yes and no - the stick as an impression in the mind’s eye is indeed as bent as it is observed. But the material stick itself might be quite straight. The water makes it appear bent. The bend is not a falsehood; it is a perspective. If you were studying light diffraction in liquids this particlar bend may be of interest. If you were trying to select the straightest stick from a few stuck in the water, this might not be the best measurement.

What radical empiricism asks us to confront is the chasm between phenomenon and operationalisation. The world as given is filled with phenomena, a “blooming, buzzing confusion” in James’ eyes. Lately, scholars have sought to undermine this notion of James’. They note that babies do not arrive in the world as a ‘blank slate’. Almost as soon as babies open their eyes, it seems they can identify facees, have phobias, understand some basic mechanics and proprioception (i.e. know where their body is in space xx). But this somewhat misunderstands James’ point. Ironically, it means that James’ point can be made more forcefully as a consequence. Even the youngest child will already have some cognitive biases in place that intervene in our ability to see the world as it truly is. We as humans have capacities, some of which appear innate and others learned, which enable us to encode the world. Our perceptions are given to us and then strengthened and learned as time goes on. Our perceptions might work for the world at one scale, but not at another. Trying to see the world as it is involves carefully translating phenomena into data rather than simply assuming the world is data. Thinking of phenomena and data as different is an essential skill for thinking critically about the world and thus, for doing research that will lead to new insights and challenge existing understandings.

The way that we get from phenomena to data is to **operationalise**. Operationalisation is an extremely delicate and important step in any data science effort. Actually, it is a pretty important part of any scientific research. However, for some disciplines what needs to be operationalised is already given, either because of the nature of the discipline or the nature of the phenomenon. Consider the challenges of operationalisation in psychology. We might ask “Do extroverts need as much sleep after a party as introverts?” We know that extroverts are more likely to enjoy the company of others than introverts. In fact, I have heard mention of extroverts as being ‘solar powered’ while introverts are like having ‘rechargable batteries’. But what is an extrovert? Well, it is someone who scores above a certain threshold on an extroversion scale. And why that scale? Because it is seen as reliable. But what about those just below the cut off, are they extroverts? Maybe, but for the purposes of this analysis, no. In this case we simply defer to the work of prior methodologists and give a convenient answer: because they said so. Now the question then actually becomes a little more tedious: “Do those who score greater than one standard deviation above the mean on a standardised extroversion scale report needing significantly more sleep than those who score one standard deviations below the mean?” Not only do we not say this because it sounds tedious, but because we often take “an extrovert” to mean “someone who scores greater than one standard deviation from the mean on a standardized extroversion scale”. Is this how everyday people see extroverts? Not at all, but it is a way to provide a consistent measurement. It is an operationalisation of the concept.

Psychology tends to have it a little easier than data science in this regard. First, psychometrics tends to focus on scales. There is a recipe for creating a scale: make up a large series of intelligible questions, ask them to people, use a statistical routine to identify which questions seem to be grouped together most of the time (where people who answer yes to one answer yes to the other), then confirm that those questions tend to be answered in the same direction. Create a new variable based on those questions and publish it. If the colleagues who review the paper think that the author(s) went through the steps correctly and the scale makes sense (or explains some other variable) then we are good, our scale is validated, and our concept is operationalised. Thus, while eHarmony (a matchmaking service) says they have models for marital happiness, what they actually measure are the variables that predict to a specific psychometric scale: the Dyadic Adjustment Scale first developed by Spanier in the late 1970s. It has questions like “xx” and “xx”. However, marketing the Dyadic Adjustmet Scale is a lot less appealing than marketing marital bliss.

It is less common to see data scientists deploying psychometric questionnaires than it is to see them analysing large blocks of text. Rather than seeking to put people in a lab and isolate a single variable, data scientists treat the world as their lab and try to detect signal from the noise. But in doing so, data scientists are inevitably confronted with operationalisation. For example, imagine you are trying to create a social network of people who email each other in an organization. You want to know whether there are clusters of social groups in the firm that can be identified from email traffic. Does sending a lone email count as a relationship? What about an email and a reply? If you then plot a network of all the emails and replies, it might look like a big, dense hairball; virtually everyone is connected to everyone! Instead, we might *operationalise* a relationship as involving two separate instances where people send a message to one another. Each instance could be a threaded conversation. Now, is this the ‘right’ way to operationalise a relationship? It’s hard to say. In some cases, we might say yes and others no. What is being contested is not the existence of email nor the existence of relationships but whether the *specific* critiera of measured email accurately signals a *specific* kind of relationship.

In a similar vein, in criminology we are periodically confronted with spikes in crimes. For example, in 2016 there was a spike in cases of sexual assault in the UK. Was that year particularly notable for sexual predators? No, it was notable because the #metoo movement enabled women to feel empowered to speak about things that were heretofore kept quiet. It’s not that the numbers of sexual assaults were increasing. In fact, it is strongly the case that those numbers are gradually decreasing. What changed was the number of *reported* cases. Simply by leaving out the word ‘reported’, journalists can create the impression that a spike in reported cases of sexual assault is due to a spike in the crime rather than a spike in confidence / security for victims.

A key concern with operationalisation is when people confuse data for phenomena. Gender is the social manifestation of sex dimorphism, but when we take gender to mean sex we forget about all the edge cases and exceptions. When we take crime statistics as data, we can forget about the many unreported crimes that occur. Consider that for all the drugs that are caught at the border, an *order of magnitude* more are not caught. Thus, measuring drug use by the amount of drugs seized could lead to some very tenuous numbers.

As social creatures we do not live outside of the world from which our data emerges. But it is sometimes easy to forget that because we are wrapped up in our practice. For example, take the notion of being a Latinx person. This is now a very clear identity, with conferences, groups, protected census categories and so forth. Yet Latino and Latina categories only emerged in the Twentieth Century, primarily in America as a means of identifying a group of individuals that evidently marked an important group, yet would not otherwise be identifiable by racial categories. Latinx persons can identify racially in a number of ways. But on the other hand, the label is contested by some. There as never been a Latino state or Latina nationality. Thus it is unsurprising that the vast majority of Brazilians would refer to themselves as Brazilian and not Latin American xx. But does Latinx have explanatory power? Yes, indeed.

Explanatory power, however, doesn’t come from data itself. It comes from **insight** about data. But how do we get from data to insight? That’s the second half of this section. Here we draw upon a framework from information visualisation, calld DIKW.

## From Data to Wisdom.

Above we discussed the transition from phenomenon to data. This involves the use of operationalisation to identify how to encode phenomena. Data then is a product of measured observation.

An often repeated framework in information visualization is that there is a hierarchy from data to wisdom.

* **Data** refers to that which was measured and encoded in some means.
* **Information** refers to a presentaiton of that data that signals differences we would understand.
* **Knowledge** is being able to understand the interrelatedness of the information (i.e. signals). If we can convey information in a graphic to another person we can give them knowledge.
* **Wisdom** is challenging to define. In this domain I like Alberto Cairo’s definition (from “The Functional Art”): Wisdom is “deep understanding of acquired knowledge, when we not only “get it,” but when new information blends with prior experience so completely that it makes us beter at knowing what to do in other situations, even if they are only loosely related to the information from which our original knowledge came. Just as not all the information we absorb leads to knowledge, not all of the knowledge we acquire leads to wisdom" (P. 17).

We are trying to turn data into knowledge by identifying information (statistically or otherwise) that we can convey to an audience. We make the researchers wise as they understand how this knowledge relates to their existing frame of reference and possible future phenomena.

Going from data to information is, loosely speaking, a matter of signal detection or pattern detection. Imagine opening up a corrupted file on a computer and seeing each of the three series of numbers below:

0 1 0 1 0 1 0 1 0 1 0 1

1 1 1 1 1 1 0 1 1 1 1 1

0 0 1 0 1 1 1 0 0 1 0 1 xx

What was in the original file? The first one is a repeating sequence. If there’s a 0, the next number is a 1. The second one appears to be nothing but 1 except for a single 0, the third one appears to be random or a bit of a jumble. Information here is about understanding the signals from the noise. In the first case, there is not much information after the first few digits. In the second one, there is an anomoly in an otherwise steady state. In the third one it appears to be random.

In all three cases there is some information to be gleaned. The first series is repeating, the second is consistent with some fluctuation and the third appears to be random. Thus, we can use this information to communicate something to someone else. Yet, what if it turns out that the third one actually means something? If we understand the use of binary we can start to ask, is this representative of a type of information? What else uses zeros and ones and is this the application of such an encoding? The third sequence might be the beginning of a string of numbers representing something else. Wisdom is about understanding how the knowledge that we gain can be applied to other domains.

The point here is that understanding what is data does not have to happen right away and it might not be obvious just by looking at the data. First, we have to identify the information from that data and then understand how that information relates to its context, thus creating knowledge. Finally, if we can transport that knowledge from one context to another, we are getting closer to wisdom.

Many times in social data science, we see people rush to create information but in doing so, rush right past contextual features that would create knowledge. The analyst was smart, but not necessarily wise. We can see this today in the world of analytics. We can learn how many tweets were sent, classify the number of words, see which ones are most common, plot the time when people are most active and so forth. For example, trying to learn about the experience in a hotel by sorting the words used in reviews might be illustrative, but not necessarily insightful. It is when these data are situated in their context that it becomes the most interesting. There in the hotel reviews we might learn of a specific underperforming employee or some notable external factor like roadworks that could make the difference.

This is not to give short shrift to data analytic techniques. This is from social science to data science after all, not the other way around. The techniques for clustering, sorting, combining and otherwise managing information are a necessary precursor to the sorts of insights that we get from learning about information in its context. Why? Because they allow us to see the data at different scales. These are the data wrangling and data analysis tasks that help bring insight. But the true insight comes not from being able to *look at scale* (as in ‘look at all this data I can manage on a server’), but being able to *see at scale* (as in ‘now once we clean, filter, and plot, the pattern is obvious’).

So, first I’m here to show you how to look and then, hopefully, I can help show you how to see. And to start, let’s think about what it means to manage data systematically, that is to say, what does it mean…to code?

# Beyond the interface

This is a book about some of the fundamental skills for social data science [SDS]. At the risk of being terribly reductive, social sciences tend to focus on individuals as thinking persons, with *agency*, and aggregations of persons, as *structure*. It might be how individuals alter their behaviors in markets based on market signals, how political movements form and position themselves, or users on social media platforms have their behavior nudged. In these cases as in many more, much of the activity associated with these phenomena that we consider social is now mediated. There’s traces of tickets to events, often with specific names and accounts, held on servers bby concert vendors. There’s data on electricity use monitored using smart home gadgets. Companies regularly clean and distribute data sets as challenges. Even the social sciences are getting in on this. Matt Salganik at Princeton ran the ‘Fragile Familes’ challenge in 2018 to see how data on children at two waves could predict measures such as grades or psychometric values in a third wave. Wikipedia offer a wealth of open access data and even some corporate social media allow programmatic access to streams of content.

There is indeed an explosion of data available or at least created. Some of this data is not available because of privacy or privitisation. But we can still think about this data as being able to make claims about the world.

The goal of this book is to get you thinking and doing things differently. In particlar, we want you to think beyond the interface. Facebook is not merely the page with the long newsfeed and the blue banner at the top. It is also a data controller (that’s the legal term). It hosts data, presents it in a particularly curated way, regulates who gets to see what data and who doesn’t. But as a user of Facebook’s products, one might be inclined to think Facebook *is* the newsfeed. But it’s not. The newsfeed is just a representation of the data being held by Facebook. It’s a form of algoriithmic curation. And that curation is based on decisions, ideas, and tests. But it’s also historically contingent…it did not have to be that way. And it is but one view to the lived experience that is captured in these posts, replies, photos, and videos. To think beyond the interface is to think - how can the data we see and record be represented in a different format, a format that can help us answer questions about the world.

The way people often think beyond the interface is to translate data from one representation to another. We might start with a list of friends. That’s a single, unordered, column of data: “Alice”, “Bob”, “Chuck”, and “Dee”. Now, imagine layering on just one more kind of data. It could be birthdays, it could be their profile summary, it could be their friendships between each other or where they live. In each case, you could represent the people different. We could see that Alice and Bob live in the same city or that Chuck and Dee are friends with each other. We might discover that they all have birthdays in the spring or that you message these friends most in the summer and the winter but not the spring or fall.

Answering these questions is available with the data already on Facebook, but not in the form that Facebook provides. Unfortunately, Facebook do not provide easy programmatic access to data. In fact, they have explicitly sought to close off access to data where possible (xx Hogan 2018). But Faebook still works well as an example and who knows. Maybe they will hire you as a reserch scientist someday and from within the company you might have access to all kinds of information. So imagine instead Alice, Bob, Chuck and Dee all send each other eamil instead. We can still layer this data, use it to see patterns at a different sccale and to ask questions about the people in particular or their contexts in general.

The point is to think “beyond the interface” and in doing so think about what other reprensetations of people are available and when we observe these representations what else will we learn. Sometimes we will see patterns, sometimes we won’t. Some of those patterns will be coincidences, but some will point to an interesting feature about the data and one worth probing further. To do this, however, involves applying some programming skills to the data at hand. Programming certainly isn’t the only way to do this, but it is quite powerful.

When we think of data science, people often think of machine learning. But data science involves considerably more than machine learning. That’s not to dismiss ML. It’s amazingly powerful and sometimes completely bonkers. But getting the data ready for machine learning, first through data access and then through data wrangling, are critical parts of the puzzle. They are often the less glamourous parts, but they are also some of the most carefully done and intensely scrutinised. This is because they are the parts of the research process where the researcher must exercise the most judgment. This is not an either/or proposition. But it is to suggest that some parts of data science can keep algorithms as black boxes. Machine learning can be done in a bit of a black box fashion. When we get results that make sense or seem to be useful given our various goodness of fit metrics, we can often ignore the specifics of the process that got us there. This is not to be naive about them, but to understand our limits.

On the other hand, there’s virtually nothing black box about data access and wrangling. In the code that follows there will be a handful of places where we simply ask the computer (i.e. “call a method”) in a way where it does not really matter if we understand the precise implementation of a task. But for the most part, every operation must be carefully told and deliberately stipulated. In the pages that follow, I will be discussing not merely programming. It is better thought of as the art of asking and answering systematic questions. The thing is, most programmers I know do not have perfect or even great memories. While coding is like a language, its one where there are lots of phrases that are hard to rattle off by heart and remember when needed. Instead, people make use of a huge volume of online resources, books like this one, searching the web, asking questions in forums and in chat rooms. The programming will come with time. And with some small successes behind you, you too will start to have ever greater successes.

The challenge is not how to program, but *what* to program. Often when problem solving code issues with students, I would ask ‘what do you want to do’ not ‘do you know how to use this or that method’. Because when we break it down, it often comes together coherently, and then getting to code to run is more a matter of fiddling with syntax than anything else. On the other hand, sometimes, data is structured in a really peculiar way or simple a way not amenable to the sorts of questions you want to answer. But by keeping an eye on what question we want to answer we can more directly focus both the data cleaning and the potential analytical approaches.

Much of this work thus centers around the twin instruments of the social scientist’s trade: *operationalisation* and *research questions*. Strangely though, we won’t be talking too much about these topics in detail for a while. Although research questions come as second nature for a seasoned researcher, students often struggle coming up with good research questions right away. Some have a knack for this, but others often have either a specific skill they simply want to employ or just a domain that they want to learn more about. What we want to foster in the first half of this book is a playful approach to the task of capturing insight from data. Such an approach should allow students to explore data, ask questions, and warm up to the data itself. Then, once comfortable ‘in the sandbox’ so to speak, we can start focusing our questions, checking our biases and thinking about how to put together research that doesn’t simply *describe data* but *explain phenomena*.

## Building a socialscope

Phenomena for social scientists often happens at scales that we do not normally perceive. Like in the example of layering data on to a list of friends, the implication is that we will learn something about the friends by thinking about these friends as a collection or an aggregate. Perhaps this is because we can detect some similarity between all four, or some regularity in the behavior of these friends either with each other or with you, the person who knows these friends. We can say these regularities happen at scale.

Scale is an extremely important concept for data science. Sometimes things only work at small scales or large scales, they work for certain time periods but not others. Scale, in that sense, is like helping to determine how far out we can extend our findings. In history, there is the concept of ideographic and nomothetic writing. Idiographic writing is about the specifics and the details of an event; which minister introduced what bill or how did that person express their ideas on that comment board. Nomothetic writing is about ascertaining the general laws or patterns that extend across contexts. Soemtimes when we zoom out to be totally nomothetic we can learn really fascinating things about the world, but we might actually lose sight of some of the processes, how they are experienced and how people make meaning of the context. Some disciplines are extremely nomothetic, such as physics. Others are extremely idiographic, such as ethnography. We are not here to judge, but to facilitate. For those doing detailed word in text, are they collecting the right text, are they interpreting comments in relation to what came before or after? Are they looking at enough text to tell a story? Enough to train a classifier? Enough to train a neural network? Thinking about building a socialscope means thinking about capturing the right scale. The things you learn with a microscope are very different from what’s learned from a radio array or an orbit-based telescope. All have their place but all similarly have limitations.

Now when using the metaphor of a scope, we are implicitly conjuring ideas about objects; material things in the world. We might think if a microscope as looking at cells or telescope as viewing a planet. But this is where we need to use a little caution in our metaphors. The things in the world that we look at in social science are not always material. Sometimes they are latent, implied, or abstract. For example, consider political orientation. When people vote, they are deciding on their selection between very real candidates and very real established parties. But how well do these parties represent not just the people, but their will? People have political leanings, values, ideas, and concerns that are not always perfectly aligned with parties. It’s okay for this to be the case. Politics is not simply about the contest of wills between clearly stipulated sets of ideas, but the determination of these ideas at the macro level through the processes of electioneering and governance. In this sense, structures such as contests for who to be party leader or who to run for office become tasks in operationalising. We stabilise on a decision based on data. We say things like, “she received the majority of votes, so she should be the representative”. Does that mean she will represent everyone? Does that mean she will be a fair representative? No. It means that we went through a process where a claim was uncertain. Then according to some rules it became less uncertain. The representative then should not be seen as the ‘true’ person that perfectly represents that district. We do not even know that she will be a good representative, though we have some ideas. Maybe it doesn’t matter as long as the other guy did not get to become the representative. In this sense, it is hard to say that the election was necessary. Rather it was contingent. It was contingent on people, their history, and the many factors that went into messaging about candidates. Thus, some of the claims we can make about the election will likely be ideographic. They will tell *the story* of that election. But some will tend to be more nomothetic, like whether people feel a sense that immigrants are taking their job are likely to vote for a candidate who ‘incentivises inequality’ (for example by opposing redistributive funding that might be used to help new immigrants adjust or integrate). This gives us a clearer picture of the various wills and factions within the group.

Strictly speaking, this is not a book on social science research methods. But that’s only partially by design. Rather, to some extent it is by necessity. The social sciences have a set of skills that does not always lend itself to making many of the judgments required for social *data* science. For example, think if how many questions are asked in the way they are because it works for a survey; further, works for a survey over the phone. For questions such as how trolling undermines faith in political institutions, we cannot get a clear answer through a survey or representative sample. We can get a population level estimate of the frequency of the behavior or perhaps public opinion on trolling, but this is not the same thing. Asking the public whether they prefer cake or pie will not lead you to learning how to bake a good pie, only which pies the public define as better than others. We are here to (metaphorically) bake pies.

Imagine if, instead of asking the public which pies the prefer, we go to a recipe site and see which pie is most popular. It’s not quite the same thing. As soon as we go to that recipe site someone is going to start asking questions, tough questions. This is a shame, because we were just talking about pie. But as soon as we shift to making claims about pie those claims are going to get tested. What’s the most popular pie is not necessarily the same as the best. The best pie, by taste, might actually be incredibly difficult to make. Further, the best pie for those whose palette prefers tart will not be the same as those for whom their palette prefers sweet. What about the best pie hot versus cold? Have popular pies changed over the years? In the 1970s I hear that key lime pie was a popular dish. These days salted caramel seems to be all the rage. So again, what’s the best pie?

Answering these questions does not necessarily involve the most complex statistics or the newst algorithms. Sure, give a learning algorithm enough data about you and it might be able to determine the best pie for you personally. But that might not get you any further into making claims about what makes some pies, in general, better than others. Better for what? Sometimes when the what is clearly specified, the how becomes straightforward.

When we say better for what we are already on our way to clearing up our methodology. Better for taste…well taste is pretty subjective, so we might want reports from people. Better for calories? Well, we can measure the calories in a serving. Better for novice bakers? We can look at prep time, number of steps or number of ingredients. Better for budgets, again, same thing - perhaps cost of ingredients. But even then, the cost will vary depending on where you live; fresh berries in the summer in the UK will cost much less than greenhouse fresh berries in the winter. So, even then, while we specified better for what, that might not have been far enough.

What the pie example is really getting at is that *operationalisation* is important. When we operationalise we take some concept of interest and transform it in such a way that we can ask questions about it at the scale in which we are interested.

In many senses, social data science then is the science of operationalisation. It is the science of taking data that is often objective but partial and using it to answer questions that enable us to talk about the world in scales beyond standard human experience. When we classify, we are operationalising concepts as soon as we decide what to train on and how to split the sample. We when are doing network analysis, text mining, regressions, or just descriptives, a large amount of our time will be spent articulating what it is that our data represents and what we think it represents.

Reconciling measurement and phenomena is not exclusive to social data science, but it is central to papers in this field. If we want to measure bullying online we first have to find a way to measure it, then a way to do such measurements at a scale that we think allows for meaningful claims.

# Socialscopes needs scale

Up to this point I have spoke primarily of research design, which seems pretty far from code. But for the rest of the chapter (and indeed the rest of the early part of this book) we will be dealing with the nuts and bolts of programming. I want to suggest that the programming parts are not actually too far from this discussion. In the next couple sections in this chapter I’ll be discussing some of the art of programming. The idea here is to get you thinking about the practice of doing research as a practice of crafting means to see things at scale, as well as to understand what is the right scale.

# Fixed, variable, and Marginal Costs: Why not to build a barn.

I don’t know where I first heard this joke, but I like it and I’ve been known to tell it in my programming classes: > Why don’t you ask a computer scientist for a glass of milk? Because they will build you a barn and fill it with cows just to make the second glass that much easier to get.

The idea behind the joke is that computer scientists will –by default– venture into a level of abstraction that is not really necessary. Get a jug of milk or a carton from the store, you might say. It is quicker, simpler, and definitely cheaper than building a barn. But then would *anyone* go through all the trouble? The idea behind building the barn is not about the second glass of milk but about the glass. That is, all glasses after the first one. Drafting the program to get the first glass would be the fixed cost in this case. The marginal costs is the cost per iteration, in this case, the cost of getting any given glass once the system is set up. The joke works because we perceive the fixed cost to be somehow irrational - in this case the difference between going to the store and buying a carton of milk and literally sourcing some cows (or growing almonds or oats if your guests prefer different kinds of milk).

## From Economics to Data Science

One central part of microeconomic theory is the identification of fixed and marginal costs of production. It’s a very useful set of concepts for programming as well. However, where microeconomics often (but not necessarily) speaks in terms of the value of money or capital, here we can think in terms of human effort or time.

The reason that we work with computers is often to facilitate work that can happen at greater time scales. Processors can do the same repetitive task much more quickly and accurately than a human. That time savings allows us to see the aggregated results from many calculations. This is exciting because it allows us to get a characterisation of things at scales that we otherwise would be unlikely to perceive or do so without great effort. For example, we now have weather stations that can make forecast models on demand for different parts of the country. Instead of one meteorologist per city, we have a system in place of readings that get synced. Public access to these readings means that third party apps can use them to calculate a weather report for anywhere that is near enough to the readings.

The weather reading set up was the same in both case. A few barometers, some thermometers, and so forth. This is the fixed cost. These days that fixed cost includes sending up a satellite. It can be very costly. But the marginal costs can vary considerably. If you need another weather reading in a city between London and Birmingham, you need another meteorologist. However, if you have a series of readings across the country and an algorithm for predicting weather based on it, you simply feed in the readings and out comes the forecast. Is it as good as a meteorologist? Well, not always. It is as good as the algorithm. What we have done is take a task that can scale slowly with human effort and then make it scale quickly using algorithms.

I’ve heard before that computer scientists only know how to count three numbers: , , and . The idea here is that if you have to do it twice, you really have to do it times, and we should be able to specify some algorithm that allows us to do it times. In reality there are still lots of issues with efficiency. But indeed, this is a foundational lesson in programming: > where possible shift marginal costs to fixed costs, but not if it will add overall costs.

These, then are costs to time; either time spent writing repetitive code or time spent running that code. I see this a lot when people first learn loops and functions. They get the revelation that abstraction can help with repitition. See below:

# Name from email  
email1 = "Bernie.Hogan@gmail.com"  
email\_parts = email1.split("@")  
name1 = email\_parts[0]  
  
email2 = "Scott.Hale@oii.ox.ac.uk"  
email\_parts = email2.split("@")  
name2 = email\_parts[0]  
print(name1,name2)

Bernie.Hogan Scott.Hale

# Attempt number 2  
email\_list = ["bernie.hogan@gmail.com","Scott.hale@oii.ox.ac.uk","Taha.Yasseri@oii.ox.ac.uk"]  
names = []  
for email in email\_list:   
 names.append(email.split("@")[0])  
print(names)

['bernie.hogan', 'Scott.hale', 'Taha.Yasseri']

Notice that not only is attempt number 2 shorter, but that it includes a third email address. It takes the operations from the first section, puts them in a for loop and spits out the result in an array. If we were to add a new email to the first attempt, we would have to copy and paste or write new code from scratch. In the second example, we simple add a new email to the list and the rest will work as intended with no extra tweaking. In that sense, the first one adds more fixed costs because we have to write more code. The second one does not because we can use the same code with more data.

In computer science terms we might say that the second example has code that is more abstract, modular, and reusable. These are all things we want out of our code. But what we really want is to make our analyses as complete and efficient as possible. And recall that what we want to minimise is not simply length of code, but time taken in the analysis. So that’s where the joke comes in – building a barn represents an attempt at making the get\_milk routine more reusable, but in such an over-the-top way that it seems like the emphasis on fixed costs have gone too far. Recall, in data science we are often not writing fully fledged production code but code that works for our analysis. Thus, we want to optimise fixed and marginal costs rather than do everything by hand (maximizing marginal costs) or create the super-analyzer 2000 that has every possible feature but is so unwieldy that it might never get off the ground.

## The challenges of maximising fixed costs

Rarely in a computer science text will you hear about the virtues of code that is a one-off, or just hacked together. I do not want to change that, necessarily. However, I do know that I have seen code that is often far more overwrought and abstract than it needs to be. For example, the push towards object-oriented programming is excellent for some tasks, but I often have trouble finding opportunities to articulate user-defined objects to students in ways that actually help their analysis, even if it helps their code architecture.

To help with this so that we are neither going to spend too much time planning and building that barn nor are we going to do it all by hand, I have a series of pointers here. I would love to add more to these but I think it’s a start.

1. **Psuedocode sections of your analysis before embarking upon it**: This way you can discover data you might need or methods that you can treat as black boxes. This is covered later in this chapter.
2. **Refactor your code during your analysis, not after**: Refactor is a way of taking repetitive or messy code and writing it in a way that makes it more elegant and ideally more robust. It is okay to spend an afternoon on your code midway or late in your analysis and just redo the code. Sometimes you only know how to do certain things after having done them, or you only know what is repetitive once you start on them and discover all the commonalities. If you have budgeted time to refactor, it will definitely pay off when you have to come back to the code, extend it, or share it with others.
3. **If you can do it by hand in a few minutes, do it, but document it**: Accept that sometimes it is just easier to recode a few variables by hand than try to find a regular expression or other mechanism for automating this or discovering a more abstract way.
4. **Do not treat your analysis as the basis of a huge project for others**: Time and again, I have seen people slowed down by the sense that they need to create a code base that will not just be used for their analysis but for others in the field. The intention is admirable but the realisation often gets in the way of doing a quality analysis. The best way to get your work reused is to have a paper published that others can cite. Thus, in all cases, I strongly recommend doing one-off analyses for yourself or your group and accepting that this code is not going to go very far. It should still be readable, serviceable, and well commented…for you and your team. But get some projects under your belt first and ideally contribute to other people’s projects to get a sense of what it means to get your software used by others. You will discover a whole host of things you might not have considered (and admittedly that we can’t cover in this book).

Beyond these pointers, we can use the notion of FREE coding as a guide for our code to help us understand what to prioritise and what will be most effective in helping us accomplish our goals. This is covered in the next section.

# Code should be FREE

Below is an explanation of the idea that code must be FREE. It’s a bit of a pun in the coding world, there is a major movement in most technical arenas to make code Free and Open Source. While it is important to familiarise yourself with the notion of free and open source coding, this is a different matter. Here FREE is mnemonic to help you understand how to focus your coding efforts.

* **F**unctional
* **R**obust
* **E**legant
* **E**fficient

## Functional code

Functional code is code that gives the expected result. If you are building a way to calculate a series of numbers or run a slice from a DataFrame, in all cases you want the answer to be correct, or as expected.

In the case of functions and methods, this does not mean that you are meeting all eventualities. It means that you are abiding by the pre-condition / post-condition contract. This contract is important for building modular code. Each module (whether it is a script, a class file or a program in its own right) has a sense of what is the correct input. This is the ‘pre-condition’. If this pre-condition is satisfied then the post-condition will be correcct.

For example, if you give me an integer and I say I will square it, then for every integer I should be able to do this.

def square(number):  
 squarednumber = number \* number   
 return squarednumber

Treated as a ‘black box’, we can say that if you give this function a number then it will return the correct, squared value.

Finally, a note on language. This is functional code in the sense that it functions as expected. There is also a notion of ‘functional programming’, which is a style of programming. That’s not what we mean here. We mean code that gets the correct result.

## Robust code

Code that is funcctional with the precondition might not be functional in other contexts. What if the user sends in a string (which we know cannot be squared)? What if a user sends in a really long integer, longer than you would normally expect, but still an integer? This is where we need to think about how to ensure that our code is not simply functioning, but robust.

One common way to ensure robust code is to check for data types. Another is to use try / catch statements. The most thorough way is to use *unit tests*. In our example, we can make the code more robust by checking that the input is a number. If it is, we square it, and if it is not, we return False.

import numbers   
  
def square(number):  
 if isinstance(number, numbers.Number):  
 squarednumber = number \* number   
 return squarednumber  
 else:  
 return False  
  
square("b")

False

## Elegant

Code that is elegant is code that doesn’t waste space or add extra layers of complexity. This is where our barn building comes in.

If we have to do some data processing and we do type out the same task repeatedly our code is not very elegant. Often times, it is said that reusable code is good code. This does not necessarily mean reusable by someone else. It can also mean reused within the same program. In the example above, our code is somewhat elegant in that it uses a function to perform some action and that function can be reused. But you will also notice that it creates a new variable “squarednumber”. We can simply get rid of that and return the square directly.

def square(number):  
 if isinstance(number, numbers.Number):  
 return number \* number  
 else:  
 return False

We can further think about ways to simplfy our code. For example, we could make the function “powersOf” rather than square and make 2 the default:

def powersOf(number,power = 2):  
 if isinstance(number, numbers.Number):  
 return number \*\* power  
 else:  
 return False

In this second case, the code might not be the most efficient (because it creates a second variable, “power”. But it is elegant in terms of functions. of course, we could go too far in the other direction: we could make a completely general purpose function with hundreds or thousands of arguments. That wouldn’t be elegant either becuase of all the possible contingencies and the fact that our code is likely to be messy and disorganized.

There is an art to thinking about what level of abstraction works best. It is where we shift our skills from being strictly programmatic to thinking of coding as writing.

## Efficient

All else equal, we want our code to be as efficient as possible. However, that’s asssuming the first three things are taken care of. In python, there are a variety of means to make code more efficient. For example, there is the [time and timeit modules](https://github.com/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/01.07-Timing-and-Profiling.ipynb) which can be used to get a sense of how long a task takes.

Overall, efficient code tends to make use of low level python features. For example, “broadcasting” is more efficent than going through a for loop one at a time. But these sorts of tricks tend to beccome known as they are needed. In most cases, the greatest speed bump in code is reading bad code, not in running slow code. That’s obviously not the case for the biggest data, but even then code has to be functional and robust first and foremost and ought to be elegant for the developers seeking to use that code.

xx timeit

# Pseudocode (and pseudo-pseudocode)

Pseudo code is a means by which we articulate what we want to do with code without being too careful syntactically. It’s about clearing away the specifics or abstracting them from the code. Often, well written pseudocode can translate very easily into running code. One of the nice things about pseudocode is that it can help us resolve some fixed/marginal cost decisions. We can refactor pseudocode simply by rewriting it rather than worrying about all the little issues with debugging. When planning some approach to exercises in this book or in your own research, it will often be helpful to use pseudocode first.

## Pseduo-pseduocode?

Pseudocode is not quite computer code. But it is often written in a format that is close to formal. Have a look at Springer’s LINCS books (Lectures in Computer Science) to see that even the pseudocode itself is quite formal. Below we will look at a simple algorithm (the geometric mean) as an example. This will include both pseudocode in a mathematical sense but also in a more informal sense. In all cases, we will want to ensure that the instructions are functional, but it will not include many of the things that make it robust.

Now there’s no real thing as pseudo-pseudocode. But here I want to suggest that pseudocode varies in its syntactic clarity. More formal pseudocode uses specific mathematical symbols or follows the general syntax of a specific language. More informal pseudocode is simply a set of instructions, written in an inconsistent or conversational style. This is not a bad thing. The function of pseudocode is to help you organize your thoughts. If you are trying to organise and writing it in a certain way helps, then don’t fret over its formality. However, when you go to share this with someone else, the more formal, the less likely that there will be ambiguity about what you meant.

Have a look at the following four examples referring to the arithmetic mean. In a way they are all pseudocode but they do vary both in terms of how easily it is said and how much ambiguity is involved.

## Attempt 1. Pseudocode as written word:

Add all of the elements together and divide by the number of elements.

## Attempt 2. Pseudocode as mathematical:

Note that here, someone familiar with the formula could probably detect what is happening. However, if you do not read math, then it can be a challenge. The big E-like character means sum all the elements. The refers to any element and the subscript means that it is done for each element . It is our ‘iterator’. The rest you can probably get yourself, but just a note that it is often convention to place a bar over a variable to imply it is the average of that variable. So means the average (usually taken to be specifically the ‘arithmetic mean’) of .

## Attempt 3. Pseudocode as written code:

get a collection of elements   
get count of elements   
set total at zero   
  
for each element:   
 add value to total  
   
result equals total divided by number of elements

## Attempt 4. Slightly more formal pseudocode (in a Python style):

def average(elements):   
 count = length(elements)  
 total = 0  
 for element in elements:   
 total += element   
 return (total/count)

In all honesty I did not mean to write Attempt 4 as working Python but it is highly likely that the formal code would change little from Attempt 4. The only difference I can see here is that I would first like to check if each of the elements is a number and that getting the length of elements requires a method called len() not length().

# Summary

This chapter spoke primarily about the art of social data science. This involves both the art of good operationalisation and the art of good coding. The reasons we see these as arts is because they often involve the subjective understandings of people. While we often strive for objectivity in science, it should be seen as a form of continual negotiation. The more objective we make something the more it becomes brittle and systematic. The more subjective we make something, the more personal and experiential it is. What we want is to draw upon our personal understandings to come up with a data analysis that is fair to the phenomenon so hopefully we can achieve work that is both valid and reliable. But phenomena change over time as do our understandings of what to label the phenomenon or how to describe what goes into or is excluded from classifying.

This tightrope walk of trying to be objective but not so objective as to ignore contingency is tricky. Accepting its inevitability will help you make better decisions when coding and better conclusions from your analysis. Now with much of this wisdom already stated, let’s begin learning some of the key skills in Python that can help us get data in a form to analyse. Then we can start to explore this data, refine our specific questions and look at ways of creating specific actionable and ideally insightful research projects.