# Doing Transparent and Reproducible Quantitative Sociology

by

Pierson Browne

A thesis

presented to the University of Waterloo

in fulfillment of the

thesis requirement for the degree of

Doctor of Philosophy

in

Sociology

Waterloo, Ontario, Canada, 2023

## Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

| | |
|---|---|
| Supervisor | Dr. John McLevey<br>Associate Professor |
| Internal Member | Dr. Owen Gallupe<br>Associate Professor |
| Internal Member | Dr. Peter Carrington<br>Adjunct Professor |
| Internal-External | Dr. Kathryn Plaisance<br>Associate Professor |
| External Examiner | Dr. Fedor Dokshin<br>Assistant Professor |
| Chair | Dr. Michael Drescher<br>Associate Professor |

## Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners. I understand that my thesis may be made electronically available to the public.

# Statement of Contributions

## Chapter 1 – Introduction: The Foundational Cycle

Except where otherwise noted, the introductory chapter is wholly comprised of the dissertation author's original prose and diagrams.

## Chapter 2 – Causal Inference: Development and Method

Except where otherwise noted, Chapter 2 is wholly comprised of the dissertation author's original prose and diagrams.

## Chapter 3 – A Review of Causal Methods in Contemporary Sociology

Except where otherwise noted, Chapter 3 is wholly comprised of the dissertation author's original prose and diagrams.

## Chapter 4 – Reproducibility and Principled Data Processing

Chapter 4 is a reproduction of a chapter titled "Reproducibility and Principled Data Processing" from *The Handbook of Computational Social Science, Volume 2*, coauthored by John McLevey, Tyler Crick, and the dissertation author (2021). McLevey (in an authorial role) and Crick have consented to the chapter's inclusion in this dissertation. McLevey (in his capacity as supervisor to the dissertation author) has cleared the chapter for inclusion in this dissertation.

The dissertation author was involved in developing the general arguments advanced in the chapter in collaboration with John McLevey. Of the chapter's eight sections, the dissertation author was responsible for drafting the prose in the 'Adhocracy, black boxes, and the collaboration problem,' 'Principled data processing,' 'Introducing `pdpp`,' 'How to use `pdpp`,' 'Sample workflow', and 'Conclusion' sections. Combined, these sections account for over half the chapter's total length. The dissertation author was also involved in editing and revising all other sections of the article. The dissertation author was also responsible for producing all of the diagrams, figures, and code for the handbook chapter.

This chapter appears here in almost exactly the same form as it was published in the handbook. Aside from a small set of stylistic alterations (font, layout, formatting, etc.), the concatenation of the references list with the rest of the dissertation's, and a handful of copy edits, the only significant change is the addition of the 'Foundational Cycle' diagram on page 2.

The dissertation author was also the lead designer and developer for the `pdpp` software package (Browne et al. 2021) – a focal point of the chapter. Tyler Crick and Rachel Wood were co-developers on the project, and John McLevey was a co-designer.

## Chapter 5 – Ameliorative Arguments for Bayesian Inference in Sociology

The penultimate section of Chapter 5 contains several references to code and statistical models the dissertation author originally developed as part of a Graduate Research Scholarship with John McLevey. Those models were included in two chapters of *Doing Computational Social Science* (McLevey 2021) and are included with permission. None of the prose or visualizations therefrom have been reproduced, either in whole or in part. The underlying statistical model from the textbook was merely used here as an example of a Bayesian model. This was done to demonstrate how the `pyKrusch` (Browne 2021) software package can automatically produce detailed visualizations of extant Bayesian models whose specification was not necessarily designed to facilitate said visualization.

Except where otherwise noted, the remainder of chapter 5 is comprised of the dissertation author's original prose, code, and diagrams.

## Chapter 6 – Conclusion: Bringing The Foundational Cycle into Being

Except where otherwise noted, the concluding chapter is wholly comprised of the dissertation author's original prose and diagrams.

# Abstract

The ongoing replication crisis (Baker 2016; Gelman and Loken 2016; Freese and Peterson 2017; Wiggins and Christopherson 2019; Bird 2020; Colling and Szűcs 2021) has laid bare quantitative sociology's need for better standards of transparency and reproducibility in all published research. This dissertation's core contribution is the proposal and articulation of a 'foundational cycle' of three interrelated methodological practices: Causal Inference (Rubin 1974; Pearl 2009b, 2009a; Pearl, Glymour, and Jewell 2016), Principled Data Processing (Ball 2016a, 2016b; Barrett 2022), and Bayesian Inference (Bayes 1763; Jaynes 2003). By enshrining the principles and practices of the 'foundational cycle,' researchers ensure that transparency and reproducibility is woven into each critical juncture of the research project – this permits other researchers to comprehend and reconstruct all aspects contributing to the published findings.

The first of the dissertation's four substantive chapters contributes an account of the development of causal inference with a particular focus on the role the 'graphical' paradigm played in motivating the development (and, later, dismantling) of causal methods in quantitative sociology. It also provides a brief description of Judea Pearl's theory of inferred causation (Pearl 2009b) and argues that quantitative sociology should adopt it as the baseline model for the purposes of causal transparency and reproducibility.

The second substantive chapter – which builds directly on that of the first – addresses a gap in extant sociological literature about the prevalence of explicit causal methodology in the field. This chapter contributes a review of the causal methods employed in the quantitative articles published in 'top' sociological journals in the year 2022 (see: Jacobs 2016). The review, which examined 283 quantitative sociological articles (out of a total 574 in the review's corpus), found that – as judged by the criteria articulated in Pearl (2009b) – only 5 among them were 'causally adequate.'

The third substantive chapter's contribution takes the form of a software-based implementation of Patrick Ball's Principled Data Processing framework (Ball 2016b, 2016a), which is designed to permit the development and maintenance of transparent, reproducible data processing pipelines, even in the context of large, distributed, collaborative, and technically-complex research efforts. The software package, titled `pdpp` (Browne et al. 2021), is an accessibility-oriented iteration on Ball's original framework.

The fourth and final substantive chapter makes two contributions: the first is the development and articulation of an 'ameliorative' class of argumentation designed to address gaps in Gelman's typology of arguments in favour of Bayesian inference (Gelman 2008) – extant modes of argumentation focus on 'winning' the debate between Frequentism and Bayesianism, whereas 'ameliorative' arguments seek to address Frequentists' concerns and trepidations about the Bayesian paradigm or the transition thereto. The second contribution is an ameliorative argument reified as a software package titled `pyKrusch` (Browne 2021), which automates the creation of – and builds upon the functionality of – John Kruschke's Bayesian dependency structure diagrams (Kruschke 2014).

# Acknowledgements

If, around the time I began my doctoral studies, someone had informed me that I'd eventually write my doctoral dissertation on the topic of quantitative methodology, I'd have likely laughed the messenger out of the room. I arrived at the University of Waterloo with nary a scrap of experience in statistics or programming – that I now view computational social science as my primary research area speaks volumes about the quality of the mentorship and camaraderie I've enjoyed during my time here. Inevitably, words will fail me in my attempts to convey my appreciation to everyone who played a part in this dissertation; I shall, nevertheless, do my utmost.[1]

The person most responsible for my dramatic transformation is my supervisor, John McLevey. Throughout, John has been instrumental in shaping and guiding my forays into domains I had not previously thought accessible; I can trace the genesis of each chapter in this dissertation back to a specific suggestion John made to explore some body of work I'd not yet encountered (his urgings, clearly, bore fruit). What's more: John's Netlab – and everyone who has been a part of it – has been an unfaltering source of intellectual stimulation, challenge, exchange, and reification. Each student who has passed through the lab is changed for it, and I believe the lab's transformative power is a reflection of the scholarly environment John has exactingly nurtured over the years – it, and John's, reputation are well-earned. My time at Netlab has been a privilege and an honour – my gratitude to you, John, is boundless. It's been one heck of a ride.

I think it fitting that the two professors most directly responsible for initiating my trek into the world of computational social science – Peter Carrington and Owen Gallupe – should now sit upon this dissertation's committee. Those of us who had the good fortune to take one or both of their graduate-level courses – on Social Network Analysis and Social Statistics, respectively – still speak of them with hushed reverence; we know how lucky we were. I'd like to express my most profound appreciation for their contributions not only to this dissertation, but to the academic journeys that I and many of my peers have embarked upon.

I owe an especially deep debt of gratitude to Jennifer Whitson, whose tireless effort and advocacy on my behalf was *sine qua non* for my time at Waterloo. From moving mountains to smooth my path here, to leading me into fascinating new arenas of scholarship and methodology, to welcoming me into the institution's burgeoning community of games research, to quite literally putting a roof over my head – I'm deeply thankful for everything: I'll never forget your (and Aron's!) generosity.

As heteroclite as my path through higher education may have been, each step in the process has built on the groundwork painstakingly laid through the collective efforts of a multitude of professors and educators – too many to name here in their entirety. I'd like to convey my particular gratitude to Mia Consalvo, Martin Cooke, Johan Koskinen, Lennart Nacke, Felan Parker, Katie Plaisance, Neil Randall, Bart Simon, and David Tindall. Thank you all for everything you've done; for me, and for all of the aspiring scholars who have been lucky enough to pass under your aegis.

I'd be remiss if I didn't highlight the myriad ways in which my development as a scholar of both computational social science *and* of play has been shaped by four sterling individuals: Jeff Barrett, Tyler Crick, Sasha Graham and Sascha Lecours. Each of you possess the ability to effortlessly prompt me (and many others, I'm sure) to consider interesting problems from new perspectives – I struggle to think of a higher compliment. It's been my distinct privilege to sit across the gaming table from each of you.

There's nothing like a global pandemic to bring into sharp relief the indispensability of community for any graduate-level researcher. I'd like to thank and recognize Jillian Anderson, Joel Becker, Diana Bianchin, Laine Bourassa, Betsy Bray, Travis Dill, Judy Ehrentraut, Brittany Etmanski, Will Fast, Alexis Hill, Yasmin Koop-Monteiro, Chris Lawrence, Nicole Lepine, Steve McColl, Matthew Perks, Brian Schram, Sabrina Sgandurra, Elise Vist, Alex de Witt, and Rachel Wood for your willingness to collaborate, confer, and – in many cases – commiserate.

A deeply personal round of thanks goes to Sasha Lecours, Siobhan Geary, Solomon the Cat, and Blue the Cat for seeing me through the darkest days of COVID-19. We said we'd weather the storm together, and we did! Perhaps not all in the same room, though – much to Solomon's ceaseless vexation.

Oliver; Jess: you have been two of the most encouraging and steadfast allies a person could possibly hope for. You constantly astound me with your curiosity, your willingness to dive headlong into topics both foreign and byzantine, and your joy at the revelation of information that sheds new light on an open topic in your respective realms. Your advice and wisdom about the world beyond the ivory tower have been without peer, and have been a lone beacon of comfort at

[1] Throughout, lists of names have been alphabetized by surname.

many times when the horizon was at its most caliginous. You've bestowed upon me perhaps the most singular honour of all: a page of my unhinged scribbling about causal modelling still resides on your fridge to this day. It – all of it – means the world to me. Thank you, thank you, thank you.

If they did not do so before, it is here that worlds fail me most finally and utterly. Mom, Dad: nothing I could say can possibly convey the true depth and breadth of my gratitude for all that you have done and continue to do. Your support is the backbone that has allowed this entire effort to coalesce, even when all appeared lost. Many times over the years, I've had cause to wonder aloud how I could possibly repay you: your reply has always been the same: "pay it forward."

It is a debt I will joyfully service for the rest of my time on this earth.

# Dedication

*To Brian and Amy.*

# Table of Contents

# List of Figures

# Chapter 1 – Introduction: The Foundational Cycle

Pierson Browne

Quantitative Sociology, and quantitative social science more generally, is in crisis. At a moment when society and academia alike are in desperate need of trustworthy research results, quantitative social scientists find themselves mired in the midst of ongoing controversy about the reliability of their findings (H. M. Collins 1992; King 1995; Freese 2007; Begley and Ellis 2012; Collaboration 2015; Gelman and Loken 2016; Baker 2016; Freese and Peterson 2017; Gigerenzer 2018; Pridemore, Makel, and Plucker 2018; Christensen, Freese, and Miguel 2019; Clayton 2021). Although the causes of this crisis are myriad, the crisis in sociology stems from – and has been permitted to fester because of – the lack of collective concern for reproducibility and transparency in our research (Freese 2007; Freese and Peterson 2017; Freese, Rauf, and Voelkel 2022).

This dissertation's primary contribution is the proposal and articulation of a cycle of three interdependent practices, each component of which addresses multiple pressing problems in contemporary quantitative sociological methodology by ensuring that their respective domain of the scientific process is transparent and reproducible. They are:

- **Causal Inference (CI)**, a framework for deriving mathematical implications from systems of qualitative assumptions, without which trustworthy statistical inference about empirical effect is impossible.
  - *CI addresses the problem of using predictive models to make interventional and counterfactual claims* (Pearl and Mackenzie 2018) *and reduces the improper use and interpretation of control variables* (McElreath 2020).
- **Principled Data Processing (PDP)**, a set of practices and guidelines which collectively ensure that the data processing pipeline in any research project is transparent, auditable, scalable, and reproducible.
  - *PDP addresses the problem of opaque, un-reproducible data processing pipelines* (Ball 2016b) *and is especially useful in collaborative research involving large and/or complex datasets* (Gentzkow and Shapiro 2014).
- **Bayesian Inference**, an approach to statistical inference that demands transparent model specification and effect estimation, accomplished by specifying exact probabilities for hypotheses under consideration.
  - *Bayesian inference addresses the confusion between sampling probability and inferential probability* (Clayton 2021) *and, more generally, the uncritical application of statistical rituals* (Gigerenzer 2004).

This dissertation takes the position that each of these three practices are fundamental to the development and execution of good quantitative sociology in the current intellectual and social-political context. Each is a pillar upon which good quantitative research rests – without all three, the structure of quantitative sociological knowledge should not be considered sturdy. Together, they comprise what I will refer to throughout this dissertation as the 'foundational cycle.' *My description of these three practices as foundational should not be interpreted as a claim that other research practices are somehow subsidiary or unimportant*; *rather, my intention is to highlight research practices that have been largely neglected in spite of their indispensability* (Gentzkow and Shapiro 2014; Pearl, Glymour, and Jewell 2016; Ball 2016b; Pearl and Mackenzie 2018; Lynch and Bartlett 2019; Clayton 2021; McElreath 2021b). That they are absent from the overwhelming majority of quantitative research published to date should be cause for alarm and a reconsideration of how quantitative knowledge is produced and disseminated (Open Science Collaboration 2015; Pearl and Mackenzie 2018; Lynch and Bartlett 2019; Christensen, Freese, and Miguel 2019; McElreath 2020, 2021b; Clayton 2021).

Even though progression between the three pillars of the foundational cycle is – in the context of a given research project – non-linear,[2] they generally align along a temporal flow: a hypothetical project will begin with Causal Inference, proceed through data collection via Principled Data Processing, and derive effect estimates using Bayesian Inference.

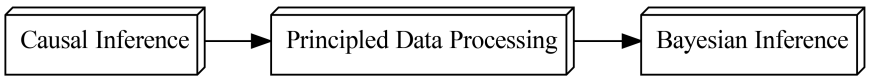


Figure 1: The three practices, arranged temporally.

I have also chosen to cast the three elements as sharing a cyclical relationship – the conclusion of one research project should, in a healthy scientific field, engender further research that directly and constructively incorporates the knowledge derived from its predecessors. The foundational cycle is visualized in figure 2.

The implementation of all three practices will greatly assist quantitative sociologists in ensuring that our research adheres to high standards of:

[2] The same is true for other cyclic models of quantitative scientific research, such as the 'loop' proposed by statistician George Box and his colleagues in the 1960s (Box and Hill 1967; Box 1976, 1980; Blei 2014).

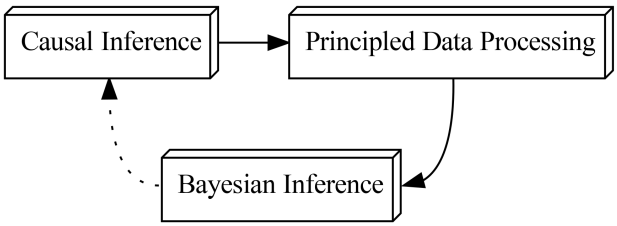


Figure 2: The foundational cycle.

- **transparency**, meaning that every aspect of a research project and its data, assumptions, processing protocols, model specifications, and published results are fully and easily accessible to the academic public.
- **reproducibility**, meaning that a research project's results can be reproduced by any interested party via the use of the project's publicly available data, assumptions, processing protocols, and modelling specifications.[3]

Together, these standards ensure that published research findings are more trustworthy, and that they are comprehensively available for scrutiny and repurposing.[4]

Each of this dissertation's four article-length chapters focuses on one of the above practices. In what follows, I provide a brief overview of each, highlighting their primary contributions.

## Causal Inference

The second and third chapters of this dissertation focus on causal inference in quantitative sociology.

In the second chapter, I review work by Judea Pearl (2009b) and Richard McElreath (2020, 2021b, 2023) that explains why statistical estimates of empirical effect are not viable in the absence of a clearly-communicated transparent causal model. The chapter demonstrates this by briefly describing how the vexing effects of Simpson's Paradox on 'traditional' approaches to variable selection for empirical statistical models – such as conditioning on a 'standard set' of control variables or all pretreatment variables – make meaningful inference nearly impossible without the assistance of a transparent causal model. It then provides a brief account of the two primary approaches to causal inference in the 20th Century: **the 'Graphical' Paradigm** (Wright 1921; Burks 1926; Blalock 1962a; Duncan 1975) and **the Potential Outcomes framework** (Neyman and Pearson 1933; Rubin 1974, 1977, 1978; Sekhon 2008). Near the conclusion of the section, I emphasize the factors that conspired to limit or reverse the development and use of causal methods in the social sciences – and especially so in regards to the use of Structural Equation Modelling, a method that largely abandoned causality following a litany of critique in the late 1980s (see: Freedman 1987; Muthen 1987; Cliff 1987; Achen 1987; Fox 1987; Bentler 1987). The final section of the chapter includes a detailed summary of Judea Pearl's theory of inferred causation (2009b), which I believe is the causal framework most likely to be taken up and productively employed in quantitative sociological research.

The third chapter builds directly on the second by addressing a gap in the current sociological literature. During the writing of this dissertation, I was unable to locate any large-scale survey of causal methods in quantitative sociology. Although anecdotal evidence and claims made by authors familiar with the field (see: Lundberg, Johnson, and Stewart 2021) suggest that contemporary quantitative sociologists rarely make explicit causal inferences from observational data, I could find no empirical evidence to this point. To address this gap, chapter 3 provides a cross-sectional review of one year's worth of publications from 11 'top' sociological journals.[5] Using the data gathered during this review, the chapter asks whether 'top' sociological journals require that the statistical inferences they publish 1) are explicitly causal, and 2)

[3] Note that 'reproduction,' for the purposes of this dissertation, is not synonymous with 'replication.' The distinction between the two is given greater consideration later in this introduction; for the time being, suffice to say that 'replication' implies the creation of a new dataset under similar conditions as the original, whereas 'reproduction' uses the original unaltered dataset. A similar distinction between the two terms was articulated in Freese, Rauf, and Voelkel (2022).

[4] The two points listed above are a generalized subset of the four principles Patrick Ball articulates in his description of Principled Data Processing (Ball 2016b). The remaining two principles – which are covered in more detail in this dissertation's chapter on Principled Data Processing – are 'auditability' and 'scalability.' Both should be considered meritorious qualities of any high-quality sociological research, but they are omitted here because they do not map onto each of the three elements of the foundational cycle.

[5] 7 of which are 'top' overall, 2 of which represent 'top' journals from the United Kingdom, with the final two representing the 'top' of Canada's sociological output.

use adequate causal notation.[6] It concludes by assessing the progress of Structural Equation Models' gradual return to their graphical-focused, explicitly-causal roots.

## Reproducibility and Principled Data Processing

As quantitative social scientists increasingly come to rely on ever-expanding datasets comprised of text corpora, digital traces, web scraping results (Salganik 2019), and other sources of 'big' data, large-scale data processing efforts are beginning to consume the lion's share of effort expended in quantitative social science research projects (Gentzkow and Shapiro 2014) – often dwarfing accompanying literature reviews and statistical analyses. This has necessitated an expansion of the typical team responsible for a social scientific study; where one or two authors may have been the norm in decades past, large teams are increasingly appearing in data-intensive social science research (Wuchty, Jones, and Uzzi 2007). The increased team size, when combined with the typical 'ad-hocratic' (Mintzberg and McHugh 1985; McLevey, Browne, and Crick 2021) nature of social scientific data processing and the increasing prevalence of bespoke scientific software, has led to an eruption of 'seat-of-your-pants' science; quantitative social science data processing pipelines are becoming so complex that they can be difficult to understand and diagnose problems in, even for the research teams responsible for them in the first place (Gentzkow and Shapiro 2014; Ball 2016a, 2016b)

The fourth chapter of this dissertation casts these issues as arising from the absence of principles enshrining transparency and reproducibility[7] in quantitative social scientific research. The need for such principles stems, in part, from social scientists' lack of training in building and maintaining software projects with large teams of contributors (Gentzkow and Shapiro 2014).[8] The chapter presents a remedy in the form of 'Principled Data Processing' (Ball 2016b), a set of guidelines, procedures, and templates intended to keep code and data processing pipelines transparent, auditable, reproducible, and scalable (Barrett 2022). It contributes to ongoing efforts to broaden access to the tools needed to facilitate principled data processing by describing and demonstrating the use of `pdpp` (Browne et al. 2021), a Python package that automates most aspects of maintaining a principled data processing pipeline.[9]

## Ameliorative Arguments for Bayesian Inference in Sociology

In the fifth chapter, I seek to resolve the apparent contradiction between the Frequentist paradigm's well-publicized inadequacies (Clayton 2021) and the Bayesian paradigm's failure to assert itself as the dominant mode of statistical inference in sociology (Lynch and Bartlett 2019). In so doing, I expand upon existing arguments about the conditions that have hampered the adoption of Bayesian inference, including the need for considerable computing power (McGrayne 2011), bespoke model complexity (Blei 2014), and a willingness to incorporate prior knowledge in statistical models (Efron 1986). Drawing upon what scant survey data exists (Natanegara et al. 2014; Faya et al. 2021; Clark et al. 2022), I contend that existing modes of argumentation for Bayesian inference have focused on Frequentism's shortcomings and/or Bayes' relative advantages (Gelman 2008). In so doing, they have ignored the barriers survey respondents most frequently evoked: the lack of knowledge, resources, and time needed to progress from mere awareness of Bayes to being competent users of Bayesian methods (Natanegara et al. 2014; Faya et al. 2021; Clark et al. 2022). In short: the Bayesian ship is self-evidently more seaworthy than the Frequentists' floundering counterpart, but Bayesians have been so busy comparing vessels that they've largely neglected to help anyone aboard.

With an eye to remedying this oversight, I use this chapter to propose an 'Ameliorative' mode of argumentation for Bayesian inference – one which directly addresses Frequentists' concerns by either pointing towards existing resources or developing them in response. In this way, ameliorative arguments are designed to help reduce or eliminate the barriers hampering Frequentists' attempts to adopt Bayesian inference. I conclude the chapter by offering an ameliorative argument of my own making: `pyKrusch` (Browne 2021), a Python package that automates and expands on 'Kruschke Diagrams' (Kruschke 2014) for Bayesian models specified in the `PyMC3` probabilistic programming environment (Salvatier, Wiecki, and Fonnesbeck 2016).[10] Among other things, the Kruschke diagrams that `pyKrusch` produces make model assumptions transparent and easy to understand, critique, and refine.

[6] Which can include potential outcomes notation, the $do(\cdot)$ operator, or causal graphs (Pearl 2009b). See chapter 2 for more information on this point.
[7] As distinct from 'replicability'; see the 'Replication vs. Reproduction' subsection below.
[8] This is not something any social scientist should be faulted for; it is not generally a known problem among the departments they were trained in or belong to. A researcher from a computer science or engineering department, by way of contrast, may have received substantial training and experience managing such projects.
[9] Installation instructions for `pdpp` can be found in Appendix D. Source code for the package can be found in Appendix C.
[10] Installation instructions for `pyKrusch` can be found in Appendix F. Source code for the package can be found in Appendix E.

# The Replication Bugbear

Although rumblings of unease had been present throughout sociology for decades prior, the 'Replication Crisis' as we know it today was galvanized in academics' minds shortly after the release of the Open Science Collaboration's Reproducibility Project (Collaboration 2015; Wiggins and Christopherson 2019). The OSC's report detailed a wide-reaching coordinated effort between hundreds of psychologists, the purpose of which was to determine how likely any given result published in field-leading psychological journals was to successfully replicate.[11] The results were damning: of the 100 attempted replications, only 36 produced significant results.[12] Even in cases of 'successful' replication, the OSC's replication studies reported effect sizes that were – on average – about half as large as the originally-reported findings (Wiggins and Christopherson 2019; Clayton 2021). Subsequent efforts have revealed that most quantitative publications suffer from similarly dubious replicability, and the problem isn't limited to the social sciences – pharmacological research and preclinical cancer studies are among the worst offenders (Clayton 2021). The floodgates are now open: no field of empirical quantitative inquiry has been spared.

My intention in evoking the replication crisis is not to centre it as the primary motivation for this dissertation: far from it. As Pashler and Wagenmakers (2012) point out, replication isn't really the crux of the issue – the OSC's report was an inciting event that precipitated a much broader 'crisis of confidence'.[13] It would be better to think of my use of the replication crisis as both a convenient and salient 'canary in the mineshaft': I am neither directly motivated by, nor desire to directly remedy, the replication crisis. I am, however, willing to interpret it as a strong sign that many things in quantitative sociology have gone badly off the rails, just as they have in other quantitative sciences.

Due to the structure of the sandwich dissertation, none of the substantive chapters contained herein are able to accommodate a nuanced description of my take on the replication crisis. To do so would occupy large chunks of limited space (not to mention readers' attention spans), and ultimately force the inclusion of redundant sections in each of the four bundled-together chapters. As such, I wish to devote a portion of this introductory chapter to the topic. Before proceeding any further, I've opted to nip a lexical issue in the bud: **replication** vs. **reproduction**.

## Replication vs. Reproduction

In the literature on the replication crisis, the terms 'replication' and 'reproduction' are often used interchangeably. Since the two words are near-synonyms, this shouldn't pose much of a problem, but it does rob us of some needed specificity.[14] Therefore, for the purposes of this dissertation:

- **Replication** is the process of re-running the experimental or observational procedures described in a published study to *gather new data under similar conditions,* and analyzing the data using the same methods employed in the original. Replication, in this sense, is primarily designed to assess the *generalizability* of published effects.
- **Reproduction**, by way of contrast, involves following the procedure described in a published study *using that study's dataset* to understand how the published results were generated. Reproduction, in this sense, is useful for examining a published study's internal validity and how sensitive it is to modelling choices.[15]

## Beyond Replication

The habitual publication of successful replication studies isn't a sufficient condition for a healthy quantitative social scientific field (Gelman 2016). Neither, I would argue, are they a necessary condition. I do not dismiss the institution of replication studies using the same logic espoused in certain Frequentist circles, who claim that replication studies aren't necessary because $p$-values can be interpreted as the probability of observing the associated result in any given replication effort (Gigerenzer 2018). I reject this view because it is, unfortunately, mistaken: neither a $p$-value nor a confidence interval contains any information about the probability of some finding being upheld in a replication study.

[11] Meaning, in this context, the probability that researchers would produce similar (and significant) results by carefully reproducing the experimental or observational protocol articulated in the original study.

[12] 'Significance', here, is taken to mean a $p$-value of 0.05 or lower. In the interest of forthrightness, I should point out that 3 of the 100 original studies selected for replication were published in spite of producing non-significant results.

[13] Pashler and Wagenmakers are specifically discussing the field of psychology, but I contend that their claims are equally applicable to any of the other social scientific disciplines.

[14] Throughout, I stick to the replication/reproduction dichotomy, even when it contradicts some established work on the topic (see: Collaboration 2015).

[15] e.g. parameter values, prior distributions, model specification, information criterion, etc. Reproduction is a necessary condition for examining model sensitivity, which is something we should all get used to doing – for our own findings and for others'.

A $p$-value merely tells us how unlikely the null is in light of the observed data, and confidence intervals merely tell us at what values a null hypothesis would not have been rejected.

My basis for viewing replication studies as superfluous is that replication is only required (or, for that matter, an intelligible concept) when using Frequentist null-hypothesis significance testing. On the surface, this claim might seem obvious: the critical ranges for most Frequentist test statistics are only meaningful in aggregate. This is to say that the paradigm's logic is as applicable to the long-run frequency of an event (e.g. flipping a coin) as it is to the long-run frequency of a significant finding (at the $\alpha = 0.05$ significance level, 5% of all statistical efforts will falsely reject the null hypothesis, necessitating endless repetitions of the same study). This statistical Uroboros stems from Frequentism's unwillingness to directly assign probability to the value that all sociologists wish to access: the 'inferential probability' (Clayton 2021) of one or more latent variables.[16] Directly assessing the inferential probability of a proposition, as is done in Bayesian inference, obviates the need for a 'replication' study, as the results speak for themselves and directly assign probability to the value we care about. As a result, Bayesian inference simply incorporates past findings in new studies of the same effect (McElreath 2020).

Throughout this dissertation, I evoke the replication crisis as a way of showing how untenable the status quo in sociology has become. The replication crisis, however, is a symptom – not a cause – of a much larger and more nebulous set of problems. As such, my goal in this dissertation is not to encourage more or better replication studies. Quantitative sociology shouldn't need replication studies at all, and by enshrining the three foundational pillars as standard components of quantitative sociological research, we can shed the replication albatross. Rather than focusing on making research replicable, our aim should be to make research 'continuable.'

## From Replication to Continuity

At risk of being exposed as a hopeless cynic, I would argue that most published quantitative research is not primarily intended to advance human knowledge about a given subject. The purpose of publication is, in many cases, to advance the authors' careers and – according to influential accounts (see: Bourdieu 1988; R. Collins 1998) – to accumulate status and prestige relative to their peers in an intellectual field. This leaves us with the tragically utilitarian view wherein the quality of a research paper is merely a tactic for increasing the probability of being published in a high-status journal. With this accomplished, the vast majority of the publication's utility to the scientific community has been expended. A rare few publications may eventually find themselves pooled with similar papers for the purposes of meta-analysis, and a vanishingly small fraction will contain newsworthy or field-shifting findings and go on to engender interest and investment from beyond the ivory tower. Most, however, will merely pad someone's CV, boost someone's H-index, or – optimistically – end up in a laundry-list literature review (McElreath 2021b). Quantitative articles published in sociology journals are no different; most are moribund. Lacking an active future, most publications are dead on arrival. The publication perishes so that its authors' careers do not.

Currently, the social sciences rely on the 'replication' doctrine (King 1995) for validity: it emphasizes the classification of research results as either 'generalizable' or 'statistical anomaly,' and uses successful replication as a heuristic to sort one from the other. This is a destructive approach to science: a published result can, at best, replicate. All that happens, in this case, is that the status quo continues (as any press-worthiness of the finding will have been garnered by the original result). This, combined with the grim reality of the publish-or-perish doctrine, heavily discourages researchers from pursuing replication studies (Baker 2016; Collaboration 2015), which leaves sociology in thrall to a doctrine whose drawbacks we suffer from, but from which we draw none of the benefits.

We can do better. Although I am yet to encounter a synthesis of these strands of thought unified under a single banner, several authors have already begun to agitate for a comprehensive re-imagining of how knowledge in the social sciences is generated, built upon, and integrated (Gelman and Shalizi 2013; Lundberg, Johnson, and Stewart 2021). For ease of discussion, I have decided to apply the moniker 'continuable' to research that is conducted so as to facilitate future constructive research. The 'continuity' doctrine is closely aligned with the Bayesian paradigm in that it views all knowledge as provisional, and emphasizes the role of new research in explicitly (and mathematically) incorporating and refining results from past research (Gelman et al. 2020; Clayton 2021; McElreath 2021b). It is constructive: a published result can be directly built upon in both a philosophical and a statistical sense. New research contributes to our understanding of the measured effect, and near-perfect 'replications' of the original findings serve to reduce uncertainty in the original result, refining our estimation of the effect size. Results that do not agree can be interpreted either as cause for a re-evaluation

[16] Defined as the probability of a hypothesis conditional on some observed data ($P[H|D]$). See the fifth chapter of this dissertation for more detail.

of the methods and techniques used in the research (past or present), as components of a unified estimate of a variable's distribution,[17] or both.

Principled data processing makes our work transparent, reproducible, auditable, and scalable; qualities which, in combination, permit other sociologists to 'open up' the analytical pipeline and critique, tinker, or repurpose any of the data, code, or method – potentially for use in their own research (Ball 2016b; Barrett 2022). Without the widespread adoption of such principles, continuable research will remain a pipe dream.

Causal inference is the means by which we can isolate and measure the effects we care about (Pearl 2009b; Pearl, Glymour, and Jewell 2016; Pearl and Mackenzie 2018; McElreath 2021b; Alves 2022). A properly-isolated causal effect should be applicable in any experimental or real-world setting. Using Causal Inference, differing results are an important clue about how researchers might best refine their Structural Causal Models or account for covariates that had not been viewed as influential. This is because the estimated causal effect should be largely invariant in the face of different control variables, provided they accurately represent and account for the latent data-generating-process.[18]

To summarize: the benefits of 'continuable' research[19] are myriad, and the pillars upon which the concept rests interact with one another in emergent ways. The possibilities opened up by enshrining the 'foundational cycle' in quantitative sociology are promising. The question, then, becomes one of adoption: if these principles have existed – in one shape or another – for over a decade,[20] why aren't they already considered standard practice in the quantitative social sciences? The issue, as I will argue in the subsequent section, is one of accessibility.

## Accessibility in the Quantitative Social Sciences

As mentioned above, none of the material presented here is truly novel. I am not responsible for developing the Bayesian paradigm, the Principled Data Processing framework, or practical Causal Inference. Most of the intellectual heavy lifting was accomplished long ago. Indeed, any quantitative social scientist with the requisite time, training, and incentive to explore, internalize, and apply Bayesian methods, Principled Data Processing, and Causal Inference would have been able to do so at any time starting in the early 1990s.[21]

The enduring tragedy of these works lies in their inaccessibility.[22] Quantitative social scientists do not generally have sufficient time,[23] training,[24] or incentive.[25] They can hardly be blamed: any methodological prescriptions *must* be accompanied with some set of easing factors capable of lowering the barriers to entry facing those who are not already inclined to implement new methodological approaches on their own initiative. To do otherwise is to obsess over a sanctimonious pipe dream with no realistic prospects of aiding anyone.

In writing this dissertation, I view myself as contributing to this ongoing process of lowering barriers to access to the core insights we need to revolutionize quantitative sociology. From personal experience, anecdotal evidence, and a strong hunch, I'm convinced that a great many early-career social scientists would benefit from the works I wish had existed when I was a 1st-year PhD student. This dissertation is, in effect, an attempt to develop the resource my younger self would have most benefited from when first setting out to self-train in recent methodological advances.[26] I can only hope it makes some small contribution towards the betterment of quantitative sociology.

[17] The net effect here being a compromise estimate with a correspondingly large variance.

[18] This property of causal estimates happens to be especially synergistic with the Bayesian paradigm. Consider the following: if two studies were conducted using a Bayesian paradigm, there would be no need to stop after comparing the two estimates. One could use the other's estimate as a prior, which would permit the overall posterior to be interpreted as a causal estimate that took all of the available data into consideration.

[19] Or whatever else one might wish to call it.

[20] And, in the case of Bayes, the better part of two centuries (Bayes 1763). This is somewhat complicated, however, by the fact that Bayes' theorem is not unique to Bayesian inference. Truly Bayesian approaches to scientific research have a shorter history, in part due to the intractability of analytical Bayes in many domains, and the relatively recent surge of approximate inference methods that are now possible given dramatic increases in computing power (McGrayne 2011; McElreath 2020; Clayton 2021).

[21] PDP was not articulated, as such, until the mid 2010s (Ball 2016b), but the principles upon which it was built have circulated for almost as long as software development has existed as a recognisable body of practice.

[22] See: Clayton (2021) with regards to Jaynes (2003) and R. T. Cox (1946).

[23] See: the ceaseless pressure of publish-or-perish in search of vanishingly rare tenure-track positions (Kiai 2019; Van Dalen 2021; Furnham 2021; Amutuhaire 2022).

[24] Most quantitative social scientists do not have formal training in mathematics and only scratch the surface of statistics, computer science, and machine learning (Gentzkow and Shapiro 2014; McLevey 2021).

[25] Many, I'm sure, would view methodological soul-searching as representing a distraction from conducting research and publishing results.

[26] This resource would have also been useful for my supervisor and peers, or so I'm told.

# Chapter 2 – Causal Inference: Development and Method

Pierson Browne

> We should not be doing any statistics at all without some transparently-communicated causal model that justifies the statistical analysis and the estimands. [...] Too often, statistics in the sciences are just causal salad. They're just a bunch of variables and they're thrown into some machine, some coefficients come out, and those coefficients are given a causal interpretation. This must stop. It is never acceptable.
>
> – Richard McElreath 2023

> Any causal premise that is cast in standard probability expressions, void of graphs, counterfactual subscripts, or $do(\cdot)$ operators, can be safely discarded as inadequate. Consequently, any article describing an empirical investigation that does not commence with expressions involving graphs, counterfactual subscripts, or $do(\cdot)$ can be safely proclaimed as inadequately written.
>
> – Judea Pearl 2009, 334

# Introduction

In this chapter, I take the position that Causal Inference should be considered an essential component of all quantitative inferential sociology.[27] As of the publication of this dissertation, sociologists enjoy access to multiple mathematically-rigorous causal 'languages': counterfactual notation drawn from Jerzy Neyman and Donald B. Rubin's 'Potential-Outcomes' framework (Rubin 1974, 1977, 1978, 1980a, 1981, 1986, 1990b), structural equations using Judea Pearl's $do(\cdot)$ operator (Pearl 1994, 1995, 2009b; Pearl and Verma 1995; Pearl, Glymour, and Jewell 2016; Pearl and Mackenzie 2018), or Structural Causal Models (Morgan and Winship 2015; Pearl, Glymour, and Jewell 2016; Pearl and Mackenzie 2018) – all are capable of articulating the causal assumptions motivating empirical quantitative analysis. What sociologists should not enjoy, however, is any leniency for failing to clearly and unambiguously communicate said causal assumptions.

This chapter will proceed by:

1. Discussing Simpson's Paradox, which demonstrates the perils inherent to interpreting any estimate of statistical effect in the absence of a transparent causal model.
2. Offering an account of the development of the main approaches to causal inference, showing how the approaches thereto that emerged throughout the 20th century either failed to gain widespread support in sociology or were subsequently driven out of the field.
3. Providing a technical account of Judea Pearl's graphical theory of inferred causation. This section also argues that Pearl's causal inference is the most suitable causal framework for quantitative sociological research.

Causal inference is the first component of the 'foundational cycle' (see Figure 3) espoused throughout this dissertation. The causal assumptions made therein have implications for all downstream research activities: when we neglect to state them clearly and systematically, we do so at our peril.

# Data is Never Enough

At present, the social sciences[28] are continuing to buy into the opportunities provided by the advent of 'big data' (Burrows and Savage 2014; Halford and Savage 2017). Overwhelmingly, the focus of big-data research has been on making accurate

[27] Note that my position here differs from that espoused by Pearl and McElreath in the quotes above. Both (McElreath 2021b, 2020; Pearl 2009b) have claimed that *all* empirical quantitative research should be considered incomplete in the absence of a transparent causal model. This, McElreath contends, is true even in the case of descriptive and demographic research – in which case the causal model is not one of effect, but rather a causal model of an observation (McElreath 2021b). While I agree that all research – including, in my opinion, qualitative research – is strengthened by the explicit communication of causal assumptions, I am only convinced of causal models' absolute necessity in cases where the effect of one variable on another is being estimated. As such, I have largely limited the claims I make in this chapter and the subsequent chapter to works of inferential empirical quantitative sociology – those, in other words, that estimate one or more statistical effects.

[28] Alongside many other fields.

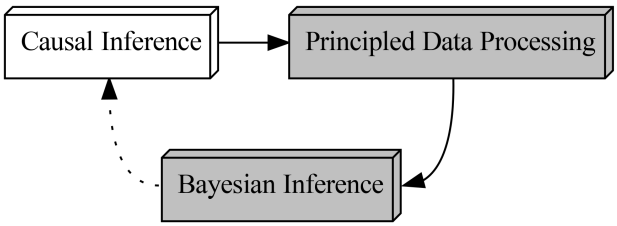


Figure 3: The foundational cycle, highlighting Causal Inference.

predictions.[29] While a prediction-centric paradigm may be appropriate for the questions data science seeks to answer, it most certainly is *not* appropriate for sociology.[30] To illustrate why this is the case, consider the following:

## A Tale of Two Education Systems

Imagine, if you will, that the fictional state of Springfield wishes to know whether the availability of robotics learning kits for elementary school students (which we'll track using binary variable $R$, for 'Robotics Kits') has a positive influence on a school's chances of meeting certain student performance thresholds (which we'll track using binary variable $Y$, for 'Yes, the desired thresholds were met').[31] It'd be fair to say that the city of Springfield is interested in the effect of $R$ on $Y$, which we could render as follows:

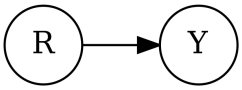


Figure 4: Graph of effect of R on Y

Fortuitously, a handful of schools under their purview began distributing robotics kits to students last year in a bid to enhance their learning experience. Springfield collects information about the two variables from each of their schools.[32] The data, once collected, looks like this:

| $R \rightarrow Y$ | |
|---|---|
| $R = 0$ (Kits Unavailable) | 33 in 149 met threshold (22%) |
| $R = 1$ (Kits Available) | 62 in 151 met threshold (42%) |

Judging by the foregoing table, Springfield might feel safe in drawing the conclusion that the availability of robotics kits improves student acumen overall – but Springfield's educational administrators would be premature in presuming that they can trust their results. In isolation, it's indisputable that the 'robotics kits' variable ($R$) correlates positively with educational outcomes ($Y$), but it might not be true that robotics kits *cause* positive educational outcomes. Springfield decides that their results might be clearer or more convincing if they stratify their data using another variable: they decide to go with neighbourhood wealth ($W$). With the addition of another dimension, the new dataset looks like this:

| $R \rightarrow Y$, stratified on $W$ | $W = 0$ (Lower-Income) | $W = 1$ (Higher-Income) |
|---|---|---|
| $R = 0$ (Kits Unavailable) | 16 in 130 met threshold (12%) | 17 in 19 met threshold (89%) |

[29] Or, in many cases, 'retrodictions' (McElreath 2020).

[30] Although it may be difficult to convince readers at this point, I argue that prediction is an inappropriate basis for social scientific inquiry because sociologists are explicitly in the business of asking and answering *interventional* or *counterfactual* questions. As Judea Pearl has repeatedly shown, one cannot use information about prediction alone to answer questions about intervention or counterfactuals (Pearl and Mackenzie 2018).

[31] The example used in this section is hypothetical and exceedingly simple, but was inspired by real-world referents where the provision of robotics kits to elementary schools was done without explicit consideration of causal factors and their potential to confound any conclusions drawn therefrom (Aurini et al. 2017; Stokes et al. 2022).

[32] For the purposes of this demonstration, I programmed a brief simulation of a data-generating process using Python. The code for the simulation can be found in Appendix B.

| $R \to Y$, stratified on $W$ | $W = 0$ (Lower-Income) | $W = 1$ (Higher-Income) |
|---|---|---|
| $R = 1$ (Kits Available) | 3 in 56 met threshold (6%) | 59 in 95 met threshold (62%) |

Once stratified on $W$, a very different picture emerges: for each of the two wealth brackets, the availability of robotics kits is associated with a *negative* impact on the probability of a given school meeting the student performance threshold. Even so, the effect of $R$ on $Y$ remains positive overall.

The above is a toy example of Simpson's Paradox (Simpson 1951), a statistical conundrum which describes a dataset where the raw correlation between two variables of interest – typically, a treatment and an outcome – runs in one direction across an entire population of interest, but is reversed for all subgroups of the same population. The scenario Simpson describes is a paradox precisely because it defies our (completely correct) intuition that an effect cannot truly be positive for an entire population and yet be negative for each component thereof. Although such a paradoxical combination of effects poses no issues in the mathematical realm, such contradictions are not possible in the real world: a treatment cannot truly have a negative impact on each group while having a positive effect overall. As reassuring as it may be to learn that our intuition here has not failed us, we're nevertheless left asking: which is the true effect? Does the effect $R \to Y$ reveal the truth, or do we need to account for $W$ in order to produce an accurate estimate?

Data alone can't tell us if our model has adjusted for the 'correct' set of variables (Pearl 2009b). Even with an infinite or near-infinite sample size, stratifying a sample can cause the apparent effect of one variable upon another to reverse its direction (Pearl 2009b, 78). What's more, there is no upper bound on this vexing behaviour as the effect-flipping process can continue *ad infinitum*: the addition of a fourth variable could flip the effect back to its original direction, the addition of a fifth could reverse it again, and so on and so forth. The analyst can never be certain if the addition of another variable will cause the effect's sign to flip yet again. None of the standard techniques for model comparison are even remotely helpful in this regard: $R^2$ scores, goodness-of-fit, information criterion, minimum description length, sensitivity analysis, bootstrapping… All are equally useless at indicating whether or not the model is minimally sufficient to estimate the effect of interest. This doesn't necessarily imply that *all* data-only effect sizes are guaranteed to be misleading: even absent a guiding principle for variable selection, it's still possible to stumble across an admissible answer – it's just highly unlikely (McElreath 2021a).[33]

The solution to Simpson's Paradox is to make assumptions about the structure of the data generating process. In our toy example above, the contradiction was brought to light when the dataset was conditioned on a third variable. This could mean that the third variable, $W$, is a common cause of both $R$ and $Y$. In such a case, we can say that $W$ confounds the effect of $R$ on $Y$, as in Figure 5:

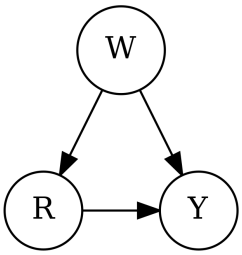


Figure 5: A confounded SCM, where $W$ confounds $R$'s effect on $Y$

One plausible explanation for this data generating process is that $W$ affects both $R$ and $Y$ because wealthier parents are more able to contribute to a school's discretionary funds (for purchases such as robotics kits) *and* are more able to provide

[33] This means that the ultimate decision about which variables to include is, more often than not, informed by precedent or what feels 'right.' Pearl writes: "Despite a century of analysis, questions of regressor selection or adjustment for covariates continue to be decided informally, case by case, with the decision resting on folklore and intuition rather than hard mathematics. The standard statistical literature is remarkably silent on this issue. Aside from nothing that one should not adjust for a covariate that is affected by the putative cause ($X$), the literature provides no guidelines as to what covariates might be admissible for adjustment and what assumptions would be needed for making such a determination formally. The reason for this silence is clear: the solution to Simpson's paradox and the covariate selection problem […] rests on causal assumptions, and such assumptions cannot be expressed formally in the language of statistics" (2009b, 138–39).

their children with academic assistance (in the form of tutoring, for example).[34] The effect of robotics kits on academic performance, then, is negative: $W = 1$ causes both $P(R = 1)$ and $P(Y = 1)$ to be higher than in the case of $W = 0$, which is the true cause of the observed positive correlation between $R$ and $W$. Once stratified on $W$, the effect of $R$ on $Y$ is negative; the State of Springfield should therefore refrain from distributing the kits.

We can, however, imagine another explanation of the data wherein neighbourhoods whose schools offer robotics kits are attractive destinations for wealthy out-of-state families.[35] For this data-generating process, we can say that $R$ 'causes' $W$, and both, in turn, cause $Y$:

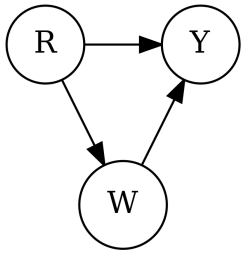


Figure 6: An unconfounded SCM, where $W$ mediates $R$'s effect on $Y$

If we assume that this non-confounding graph accurately represents the underlying data generating process, then we can use the data to confidently make a series of related claims:

1. The direct, unmediated effect of $R$ on $Y$ is negative – absent any other mechanisms, robotics kits negatively impact a school's academic performance.
2. The effect of $R$ on $W$ is positive.
3. The effect of $W$ on $Y$ is positive.
4. The indirect effect $R \rightarrow W \rightarrow Y$ is stronger than the direct effect $R \rightarrow Y$, which implies that:
5. The overall effect of $R$ on $Y$ is positive.

By reversing the direction of a single edge, we've turned the policy implication on its head. This leaves us with the question: which of the two diagrams represents the real data-generating process?

- If the first diagram is accurate:
  - It is essential that the state of Springfield stratifies its data on neighbourhood wealth, lest it allow the effect of $R$ on $Y$ to be exposed to confounding bias from $W$.
  - Robotics kits should be discouraged, as they distract children from more important academic activities.
- If the second diagram has the right of it:
  - It is essential that Springfield does *not* stratify on neighbourhood wealth, lest it close off one of the pathways by which $R$ legitimately influences the value of $Y$ (via $W$).[36]
  - The State of Springfield should encourage schools to distribute robotics kits as this will improve neighbourhood wealth (by attracting wealthy families to the state), which will in turn improve school performance.[37]

[34] If this is an accurate model, then it explains the raw data's predictive power: $R$ correlates positively with $Y$ thanks to $W$'s influence on both. If all one cares about is teasing out properties of the data as-observed, one doesn't need to know anything about $W$ or the role it plays in the data generating process. Imagine, for a moment, that the State of Springfield were only interested in determining which schools they should target for some sort of advanced placement program, and that their data about certain schools' overall academic performances ($Y$) and wealth ($W$) was missing – or, to be more precise, missing completely at random. If Springfield had data about $R$ for each of the schools with missing $Y$ and $W$ data, could they use this data to maximize their chances of recruiting high-performing students? Turns out, the answer is unequivocally yes! Despite ambiguity about the effect of $R$ on $Y$, the raw correlation is still reliable and can be used for answering predictive questions. Something similar is at work in the case of severe multicollinearity: even though a linear model with highly multicollinear predictor variables produces inferentially meaningless estimates of effect size, the model can still be used to make very accurate predictions (McElreath 2020).

[35] A somewhat implausible setup, admittedly.

[36] Adjusting for $W$ doesn't 'bias' the estimate, per se, but it does produce an estimate of a different effect. Thus far, we've been interested in the overall effect of $R$ on $Y$ – also known as the Average Treatment Effect, or ATE (Pearl 2009b). Adjusting for $W$ closes off the mediating pathway it intercepts, which gives us the Controlled Direct Effect, or CDE (Pearl 2009b).

[37] This would be a supremely cynical ploy, as the robotics kits will disadvantage extant students.

In light of this, should the State of Springfield's data analysts stratify the data on $W$, or leave the data un-segregated? As discussed above, there's no way to choose between the two on the basis of data alone. One possible solution to the problem involves running a Randomized Controlled Trial (RCT).

Enter the neighbouring State of Shelbyville which, despite being equally well-funded, has not yet permitted any of their schools to distribute robotics kits to their students. Dismayed by Springfield's paradoxical findings, they decide to run a pilot program and take action based on the results therefrom. Shelbyville places each school into one of two groups ('treatment' and 'control') *completely at random*. They provide the 'treatment' schools with robotics kits to distribute to their students, and leave the 'control' schools as they are.

By forcing the 'robotics kits' variable for any given school to take on a prescribed value ($R = r$, where $r = 1$ for schools that received robotics kits, and $r = 0$ for those that did not), Shelbyville has effectively severed the causal relationship between any potential cause of $R$ (such as $W$) and $R$. Assuming the former of the two candidate causal models is accurate, Shelbyville's RCT ensures that schools in wealthier neighbourhoods are now just as likely to be given robotics kits as schools in less affluent neighbourhoods (see Figure 7).

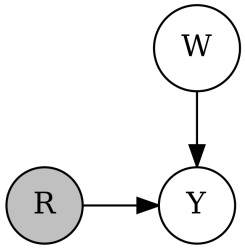


Figure 7: SCM of a Randomized Controlled Trial for R, Y, and W, where the confounding effect of W has been eliminated.

The other candidate model, however, remains unaltered. Since the presence of robotics kits attracts wealthy families to the neighbourhood, the experimental distribution thereof shouldn't impact said wealthy families' decision-making processes. Despite the experimental control, the relationships between $R$, $Y$, and $W$ in the 'mediation' graph are unaltered (see Figure 8).

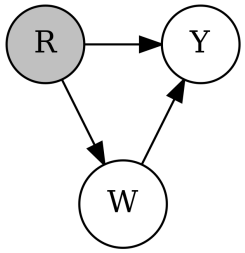


Figure 8: SCM of a Randomized Controlled Trial for R, Y, and W, where the mediating effect of W is unaltered.

Assuming that there are only two possible candidate models for explaining the relationship between $R$, $Y$, and $W$, Shelbyville's RCT should provide evidence for one over the other. Shelbyville distributes the kits, collects the data,[38] and produces the following results:

| $R \to Y$ | |
|---|---|
| $R = 0$ (Kits Unavailable) | 56 in 146 met threshold (38%) |
| $R = 1$ (Kits Available) | 39 in 154 met threshold (25%) |

[38] These results were simulated from the same set of probabilities used to generate the data in the Springfield section. The code for both Shelbyville and Springfield can be found in Appendix B.

Shelbyville's research showed that robotics kits have a *negative* impact on educational outcomes, and are best avoided.[39]

### You Always Need a Causal Model

While Randomized Controlled Trials are effective at circumventing bias in some cases, they aren't a one-size-fits-all solution to the problems incumbent upon making meaningful inferences. While perfectly-followed experimental treatments can eliminate the effect of confounding variables (whilst leaving mediators intact), they only reliably do so when adherence to the assigned treatment is complete. If, for example, certain Shelbyville schools that were assigned to the control group $R = 0$ decided to defy their assigned treatment and purchase robotics kits for students anyways, the validity of the experimental assumptions would be called into question. In such cases, the experimental treatment does *not* sever the links between the treatment and its antecedents, instead creating an imperfect instrumental variable – in this case, causal inference is still possible, provided a fully-specified causal model is available (Pearl 2009b; Morgan and Winship 2015). In many other cases, randomized experiments are simply impractical, unfeasible, or amoral. Imagine, for example, if instead of assessing the impact of robotics kits, the State of Shelbyville was interested in knowing about the impact of Rubella outbreaks on academic performance.[40] Randomly assigning certain children to be infected with virulent straits of the Rubella virus is plainly unethical. Finally, some kinds of policy-relevant questions cannot be satisfactorily answered using experimental data. If one wishes, for example, to allow a mediator to take on the value it would have 'naturally' (as if the treatment had not been intervened upon), then experimental data alone is insufficient (Pearl and Mackenzie 2018). The questions which sociologists ask – and seek to answer – demand a general-purpose approach to causality; one that does not rely on experimental trials or methods that seek to emulate them to the exclusion of observational data.

This is the primary reason why we should not trust the results of inferential quantitative sociology that is done in the absence of a transparent causal model. Researcher integrity is not the problem here. The good news is that we, as a field, aren't starting from a standstill: Sociologists already encode causal assumptions in their published work as a matter of habit. The bad news is that these assumptions are almost never transparently communicated to readers, nor are the implications of said assumptions habitually accounted for.[41] The silver lining, if there is one, is that sociologists shoulder little of the blame for this sorry state of affairs. As we shall see in the next section, the development of causal methodology in the 20th century was a fraught endeavour.

## How We Got to Now

The vast majority of the causal calculus available today can be traced back to one of two 'languages' developed to articulate causal claims (Pearl 2009b, 135): the 'Potential Outcomes' framework, and the 'Graphical' paradigm. In this section, I present a brief account of both paradigms' (highly nonlinear) development and uptake.[42]

### The 'Graphical' Paradigm

The graphical paradigm is usually referred to via synechdoche; a loose constellation of terms – including "Path Analysis," "Structural Equation Model", "Structural Causal Model", "Causal Graph", "Causal Directed Acyclic Graph (DAG)" – are frequently used to refer to the lineage of observational work attributed to authors such as Wright Wright (1934), Burks (1926), Haavelmo (1943), Blalock (1961), Duncan (1975) and Pearl (2009b).[43] Of particular interest in this section are the terms "Path Analysis" and "Structural Equation Modelling" which – despite their distinctions (Duncan 1975, 51) – are

[39] This can be interpreted as evidence for the former of the two data generating processes: the one in which $W$ is a confounder. This aligns with the causal model I used in the simulation.

[40] A not unreasonable query, given that as of the publication of this dissertation, Canadian society is still grappling with the impact of the COVID pandemic on students' academic development.

[41] Evidence to this point will be presented the subsequent chapter of this dissertation.

[42] The overall direction and focus of this account was inspired by the works of Judea Pearl (2009b), Pearl and Mackenzie (2018), and Morgan and Winship (2015).

[43] This account is, by necessity, abridged. I have omitted reams of material that might be of interest to readers looking for a more complete history of the 'graphical' causal paradigm. In particular, I have neglected to include the full breadth of the 1987 furore over SEM, as it is much broader and much more technical than I have room to do justice to here. Readers interested in knowing more are encouraged to consult the other articles in the 1987 special issue of *Journal of Educational Statistics*, including Hope (1987), Achen (1987), Bentler (1987), Cliff (1987), and so on. Notable among the articles contained therein is John Fox's (Fox 1987) impassioned defense of SEM.

often used as near-synonyms in the literature.[44]

The first verifiable emergence of mathematically-sound principles for causal inference took place nearly simultaneously in two different parts of the United States of America, by two researchers who barely knew of one another and never met (Pearl and Mackenzie 2018). The two researchers – Stanford psychology student Barbara Burks and Sewall Wright of the U.S. Department of Agriculture – separately devised similar graphical methods for expressing their assumptions about causal linkages in a system of variables.

Burks' first use of a graphical model can be found in her 1926 article entitled "On The Inadequacy of the Partial and Multiple Correlation Technique" (1926), where she used findings from her research on the heritability of intelligence to point out flaws in standard statistical practice at the time. Although Burks' graphical model followed Wright's original by several years, there is reason to believe that she developed her graphical representation independently of Wright (Pearl and Mackenzie 2018).[45] What is indisputable is that Burks was the first to use a graphical method to represent the effect of a mediator, which she used to explain why partial correlation[46] would frequently produce spurious causal estimates if causal structure was not taken into account (see: Figure 9).[47]

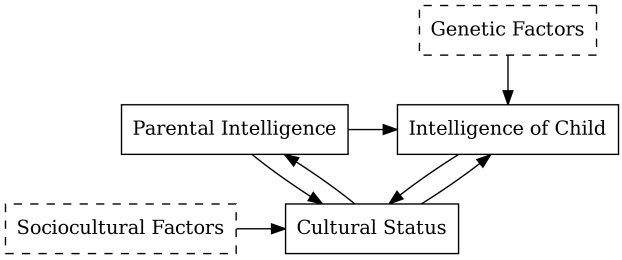


Figure 9: Causal model of Child Intelligence; adapted from Burks (1926).

First published 6 years before Burks' equally groundbreaking work, Wright's 'Path Analysis' is often cited as the methodological progenitor of two successor methods that remain influential to this day: Sociology's 'Structural Equation Modelling' and the 'Simultaneous Equation Model' still habitually used in Economics.[48] Wright's original impetus for developing Path Analysis was his desire to understand the mechanisms by which Guinea Pig coloration patterns were determined (Wright 1920). Wright wanted to know how much of a Guinea Pig's pattern variation was attributable to hereditary factors (which were observable), as opposed to developmental and environmental factors (which were not). He encoded his causal assumptions in a Path Diagram, with each path between variables representing a structural coefficient determined entirely by its antecedent variables and paths.

Wright's original Path Diagram contained the first use of graphical criteria as the basis for estimating causal effects. With this new graphical model in hand, Wright turned his attention to all manner of Guinea Pig-related quandaries, including the effect of litter size and gestation period on guinea pig weight at birth (Wright 1921).

Despite the miraculous capabilities of Burks' and Wright's graphical methods, both were burdened by their reliance on a

[44] SEM can be productively characterized as an extension of Path Analysis. While Path Analysis only permits the explicit consideration of observed variables, SEM typically includes a measurement model and allows for unobserved and latent variables to enter into the analysis (Duncan 1975). Although I would have preferred to disambiguate between the two throughout this chapter, the semantic overlap in the literature on both subjects made it necessary to use the terms interchangeably. While doing so is inaccurate in the strictest sense, the conflation of SEM and Path Analysis is not consequential for the purposes of this chapter: both are burdened by the same adherence to assumptions of linear additivity – at least in their original formulations.

[45] There's reason to believe that Burks' use of diagrams was not inspired by Wright. Pearl and MacKenzie (2018, 308) write: "Burks may actually have invented path diagrams independently of Sewall Wright, who preceded her by only six years. We can say for sure that she didn't learn them in any class. Figure 9.2 is the first appearance of a path diagram outside of Sewall Wright's work and the first ever in the social or behavioural sciences."

[46] Or, equivalently, a coefficient from a linear regression model, albeit with a different standardization.

[47] Keen-eyed readers may have noticed that, due to the presence of double-headed paths between two of the variables, Burks' path diagram doesn't take the form of a directed acyclic graph.

[48] In 1943, economist Trygve Haavelmo posited the use of 'Systems of Simultaneous Equations' as a means of making causal claims by emulating the effect of hypothetical experiments (1943). Haavelmo's contribution was mathematically similar to Wright's, but the former lacked the graphical emphasis of the latter: this preference for sequentially-listed structural equations in place of graphical depictions continues in the field of economics to this day (Heckman 1992), where it is known as 'Simultaneous Equation Modelling.' Although Haavelmo's original manuscript does not evoke causality directly, his stern warnings on pages 3 through 5 makes clear that modeling assumptions should closely follow theory and researcher knowledge, and that mathematical estimation should be subservient to both (1943, 3–5). He also explicitly states that his method is a surrogate for experimental studies, capable of emulating hypothetical experiments using observational data (Haavelmo 1943, 4, 12).

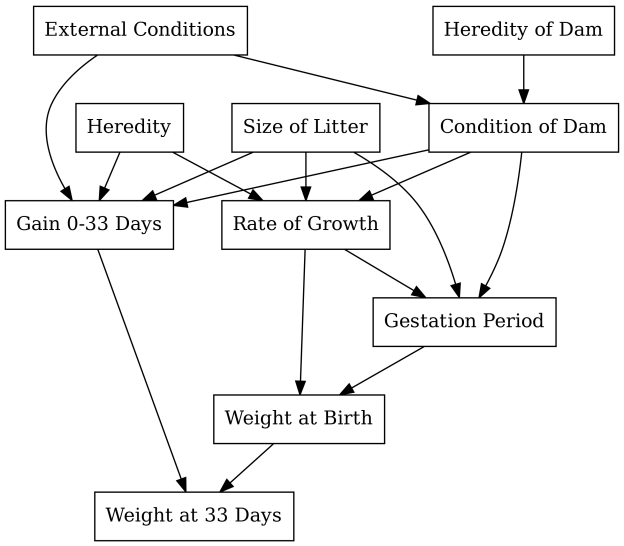


Figure 10: Causal model of Guinea Pig Weight; adapted from Wright (1921).

strict assumption of linearity.[49] While the assumption of linearity was arguably appropriate for the questions being asked in their original formulations, the assumption of linearity would not fit so easily into the contexts that Path Analysis' successors would be applied in. The assumption was habitually made anyways, despite a mounting body of evidence that it was often inappropriate or invalidating. As such, the assumption of linearity was a weak pillar upon which path analysis rested. It would take 60 years for the pillar to become overstressed and fail – when it did, it would set the graphical paradigm back by decades.

Starting in the mid 1950s, Sociology joined the burgeoning movement towards explicit causal methods. This move was precipitated, in part, by Simon (1954), Blalock (1961, 1963; Blalock 1962a), and Duncan's (1975) efforts to combine Wright's (Wright 1920, 1921) and Haavelmo's (Haavelmo 1943) methods into what we now know of as "Structural Equation Modelling" (SEM).[50] It was here that Wright's rigid graphical system was married to additions that accounted for measurement indeterminacy (in the form of SEM's measurement model component) and latent variables (Wright's original formulation of path analysis could not explicitly model latent variables). Parallel developments in economics (Haavelmo 1943; Heckman 1992) had proceeded in the absence of graphical models, preferring to rely solely on lists of equations to describe structure. By way of contrast: Blalock and Duncan's articulations of SEM retained Wright's emphasis on the use of graphs as a tool for visualizing modelling assumptions. The founders of SEM also retained Wright's orientation towards the act of model-making: for them, a SEM's structure was built atop the researcher's domain knowledge, and was thus primarily qualitative in nature (Duncan 1975).[51] Once built, the graphical models could be distilled into a series of structural equations whose parameters could be solved for using observed data and standard algebraic rules.

There is evidence that the early progenitors of SEM were concerned that the method's mathematical properties could come to predominate (Morgan and Winship 2015). Tellingly, Duncan's foundational book (1975) contains hints of his ambivalence towards the very method he helped popularize:

> It is not my purpose to advocate or defend the use of structural equation models in sociology. Indeed, I hold a rather agnostic view about their ultimate utility and vitality in that discipline, fascinating as the models may

[49] There's some lingering ambiguity around the term 'linearity' in statistical practice. Whenever I evoke the term 'linearity' in this chapter, I'm referring to strict 'additive' linearity of the form $y = \alpha + \beta_1 x_1 + \beta_2 x_2 ... \beta_i x_i$. This is the kind of linearity used in Ordinary Least Squares regression. It is incompatible with any form of interaction between the variables – including themselves. The properties of the OLS model are highly desirable, as it allows us to interpret each raw coefficient in a linear regression as an effect size that is independent of the value of any other variable in the regression – and, consequently, as a causal effect, assuming the model is correct and the effect is identifiable (neither of which are trivial assumptions). The assumptions needed to warrant the use of an OLS model, however, are strict and very rarely compatible in empirical sociological settings (Freedman 1987).

[50] Simon's (1954) foundational contribution to graphical causality neither cites nor mentions Wright or Path Analysis; but those who expanded upon Simon's work – such as Blalock (Blalock 1962b) – connect the two lineages.

[51] Though not entirely: as mentioned above, SEMs assumed linear additivity; thus, certain quantitative assumptions were 'baked in.'

> be in purely formal terms. [...] The last thing I want to do is suggest that the structural equation approach comprises a new recipe for conducting sociological research. I hope that no one, upon learning of this book, will write for me the computer program which I use to 'do' structural equation models. I have no such program. What I am trying to say throughout the book is that 'doing' the models consists largely in thinking about what kind of model one wants and can justify in light of the ideas whose validity one is prepared to take responsibility for. (Duncan 1975, viii)

Duncan's disclaimer would prove tragically prophetic. A few years before Duncan published *Introduction to Structural Equation Models*, the 'LISREL' algorithm (Joreskog and Van Thillo 1972) had been developed – it was capable of automating the exact process Duncan had hoped would never be automated. In the years that followed, the creation and interpretation of SEM became a ritualistic affair: practitioners would focus most of their attention on the vagaries of software-based estimation, with comparatively little being paid to their models and the assumptions they implied (Freedman 1987; Muthen 1987; Pearl 2009b; Morgan and Winship 2015; Pearl and Mackenzie 2018). Later in life, Duncan would come to regret having published his foundational work on SEM in the first place, claiming that "Researchers had come away from the book believing that fundamental policy questions about social inequality could be quickly and easily answered with path analysis" (Berk 2004 xvii).

Powered by LISREL, SEM was fast and easy. Cracks in the edifice, however, began to appear when the assumptions powering SEM were accused of flattening much of the non-linear interactive complexity that was an undeniable feature of the real world (Lieberson 1985; Ragin 1987; Arminger and Bohrnstedt 1987; Abbott 1988; Morgan and Winship 2015, 87–88).[52] Since the Path Diagrams of the day merely represented linear additive regression coefficients, they lacked any consistent means of indicating quadratic terms or interaction between variables. At the time, SEM practitioners were keenly aware of these shortcomings: works such as Kenny and Judd (1984) and Busemeyer and Jones (1983) worked towards developing the means to include quadratic terms and interactions, but they did so by manipulating functional forms whilst keeping the parameters linear (Bollen and Pearl 2013).[53] Parallel efforts to formalize the inclusion of non-linear variables (such as count, dichotomous, or categorical variables) succeeded by creating link models that mapped the non-linear variables onto some kind of accompanying continuous variable, producing a linear approximation thereof (Muthén 1984). Nevertheless: the advancements posited in each of these works were still shackled to SEM's assumption of linearity: all efforts merely attempted to cajole nature into being 'linear enough.'

In his withering critique of causal SEM, Karlin (1987) singles out the practice of coercing data and coefficients into linear forms as a major stumbling block: "Multivariate distributions cannot in general be transformed to multivariate normality. Attempts to transform each component variable separately to normality would, in general, alter the correlation structure, obscure the dependence relations among the variables, and disrupt the specific dependencies depicted in the model" (Karlin 1987, 166). Karlin's proposed solution, however, is deeply counterproductive: since, he contends, the models implied by Path Analysis are intractable, the best solution is to dispense with models entirely and proceed with a 'model-free' approach to conducting science – one that seeks to "understand the data interactively by using a battery of displays, indices, and contrasts" (Karlin 1987, 168).[54]

Already pressured by sustained critique, the SEM party would come to a crashing halt around the time D.A. Freedman published *As Others See Us: A Case Study in Path Analysis* (Freedman 1987). Despite the innocuous name, Freedman's critique cut to the quick by targeting SEM's very reliance on assumptions that practitioners seemed to habitually ignore. Freedman noted that the assumptions were almost never justified, and that SEM was in dire need of a comprehensive overhaul. One of Freedman's key points was that SEM's causal claims were wholly based on linear statistical law,[55] and thus were only valid when linear assumptions are respected. Freedman then went on to demonstrate that several celebrated

[52] The pre-1988 works cited here are not specifically aimed at SEM, but at practices that SEM relies upon. Abbott (1988) and Morgan and Winship (2015) describe the impact of these criticisms on SEM.

[53] Tellingly, Kenny and Judd (1984) warn users about using LISREL in combination with their proposed method: "Although we have assumed that nonproduct latent variables are normally distributed, this assumption means that the product latent variables are not. This fact means that in estimation, we should avoid minimizing a loss function that assumes multivariate normality, such as the maximum likelihood function in LISREL" (Kenny and Judd 1984, 208). Their warning anticipates elements of influential criticisms that would be voiced by Karlin (1987) and Freedman (1987) three years hence.

[54] As discussed above: if the entirety of quantitative sociology is content asking and answering exclusively predictive questions, then Karlin's proposed approach is perfectly adequate. If sociology wishes to ask and answer interventional and counterfactual questions – which, I contend in the strongest possible terms, we should – then Karlin's proposed solution is woefully inadequate.

[55] This is a natural and inevitable consequence of SEM/Path Analysis making causal assumptions that are simultaneously structural *and* functional. As Freedman notes: "the path-analytic notion of 'cause' is intimately bound up with linear statistical laws" (Freedman 1987, 108).

SEM publications proceeded with the assumption of linearity without justifying its use.[56] Freedman was, in essence, pleading with sociologists (and other SEM practitioners) to rouse themselves from the stupor brought on by uncritically assuming whatever suited their chosen estimation technique:

> One problem noticeable to a statistician is that investigators do not pay attention to the stochastic assumptions behind the models. It does not seem possible to derive these assumptions from current theory, nor are they easily validated empirically on a case-by-case basis. Also, the sheer technical complexity of the method tends to overwhelm critical judgement. I have no magical solution to offer as a replacement. On the other hand, repeating well-worn errors for lack of anything better to do can hardly be the right course of action. If I am right, it is better to abandon a faulty research paradigm and go back to the drawing boards. (Freedman 1987, 102)

Instead of attending to the underlying issues, SEM practitioners shot the messenger: nobody could accuse them of using misleading path diagrams if they didn't *have* any path diagrams, so the path diagrams began to disappear from SEM publications *en masse* (Morgan and Winship 2015, 87).[57] Similarly, critiques of SEM's linear assumptions ripped away the causal component of the method, leaving the software-powered automatic inference engine unscathed (Pearl 2009b; Morgan and Winship 2015). In this way, Freedman's critique had largely achieved the exact opposite of what he had intended: rather than continuing to make strong causal claims and adopting methods to suit, SEM practitioners generally took the easy way out by simply disavowing that SEM coefficients could be causally meaningful (Pearl 2009b; Morgan and Winship 2015; Pearl and Mackenzie 2018).[58] Bengt Muthen – a leading social scientist at the time – penned a response that summed up the field's reaction to Freedman's critique: "I agree that the credibility is further strained by an unhealthy insistence on strong causal inferences from the analyses. In my view, these statistical analyses have very little to do with causality" (Muthen 1987, 179–80). The solution, in their view, was to simply stop making causal assumptions and, consequently, avoid making causal claims (Muthen 1987).[59]

Bereft of path diagrams and causal assumptions, Structural Equation Modelling took its place among the pantheon of data-driven causally-barren methods of the kind that Karl Pearson and R.A. Fisher would have happily endorsed (Pearl 2009b, 133). Not until the emergence of non-parametric graphical methods for causal inference would SEM be equipped with the means to reclaim its original place as a fully justifiable causal method (Bollen and Pearl 2013). More on that later.

## The 'Potential Outcomes' Paradigm

In literature on the subject, one might see Potential Outcomes referred to as the "Neyman-Rubin Model", the "Rubin Causal Model", or some permutation thereof. Nomenclature notwithstanding, the Potential Outcomes framework emphasizes experimental methodology and is most closely associated with the works of Neyman (1923), Fisher (1935), Cox (1958), Holland (1986), and D.B. Rubin (1974, 1977, 1978, 1981) on the subject.[60]

The very first use of notation that would eventually come to form the bedrock of the 'Potential Outcomes' framework can be found in the writings of Polish mathematician Jerzy Neyman (1923). For his dissertation research, Neyman tasked himself with assessing the yield ($U$) of a variety of crops under different soil conditions. He indexed the crop varieties using $m$, and the various plots of land upon which they were being grown using $i$. His core objective was to measure each imaginable $U_{im}$: the yield $U$ for each combination of crop $m$ and plot $i$. For practical reasons, each plot was only

[56] Discussing Hope's (1984) SEM-based comparative work on education, Freedman notes that "Hope takes for granted that his measurements on the variables are connected through linear statistical laws. He can hardly be blamed for doing so because nearly everyone does the same; but it is precisely the move from rough insight to full-blown path model that seems to counterproductive to me" (Freedman 1987, 120)

[57] A few brave souls responded to the criticisms levelled at path analysis by attempting to shore up the notational deficiencies. Bollen (1995), for example, proposed notation designed to augment the existing path diagrams, allowing them to represent interaction terms and quadratics using a jagged or wavy edge, instead of the straight one typically used to denote a linear relationship. It's a valiant attempt, but his proposed solution doesn't transcend the core issue of linearity. Bollen writes: "Here I develop a 2SLS estimator that applies to general SEMs, including the latent variable and the measurement model. And the 2SLS estimator allows for equations that are nonlinear in the latent or observed variables, *requiring only that they be linear in the parameters*" (1995, 232, emphasis mine). Requiring parametric linearity doesn't help Bollen's proposed solution in the eyes of Freedman's critiques.

[58] Karlin, however, got exactly what he wanted.

[59] Detailed summaries of SEM practitioners' collective response to criticisms of SEM can be found in Pearl (2009b) and Morgan and Winship (2015). It is worth noting that the article containing Bollen's aforementioned attempts to shore up the deficiencies in SEM notation does not mention causality (1995). The world 'causal' appears but once: in the title of another article in the works cited list. Tellingly, this article stems from the pre-1987 causal collapse.

[60] As above, the account I provide here is necessarily abridged. I have elected to focus on Neyman and Rubin's contributions to the potential outcomes framework, as their appearances in contemporary literature (and Rubin in particular) appear most notable, not least of which due to the fact that the paradigm is often referred to using both of their names.

planted with a single crop: as such, it was impossible to directly measure each possible cell in the matrix of $U$ values under examination. If, for example, crop 5 had been planted in plot 7, then $U_{7,5}$ was directly observable, but $U_{7,4}$ was not. Neyman's revolutionary contribution was to use the empirical information available (e.g. $U_{7,5}$) to make informed estimates of *counterfactual* events – he had discovered a way to use strong assumptions about the distribution of crop yields to make predictions about events that could have happened, but did not (e.g. $U_{7,4}$) .

Shortly after Neyman published his groundbreaking work,[61] he had the misfortune of being offered a position at University College London in 1934, which meant that he was forced into close proximity to R.A. Fisher, whose capacity for marginalizing others' good ideas (Pearl and Mackenzie 2018) was rivalled only by his uncanny ability to delude himself into thinking that he was incapable of making mistakes (Rubin 2005).[62] Since Fisher viewed causality merely as strong correlation (namely, where $cov(X, Y) = 1$), he was brutally dismissive of anyone who thought that making counterfactual or causal claims required care, attention, or mathematical tools over and above Fisher's standard (and deeply misguided) approach to statistical inference (Pearl and Mackenzie 2018). Thanks to the fact that the statistical world overwhelmingly nominated R.A. Fisher as the oracular font of statistical wisdom from the mid 1930s to the late 1950s, Neyman's causal notation sat more-or-less dormant for almost 50 years.

Starting in the early 1970s, Neyman's causal writings would gain a second lease on life when statistician Donald Rubin began churning out mathematical treatments of potential outcomes based on adapted versions of Neyman's original notation. In his first major work on the subject, Rubin (1974) articulated general-purpose mathematical notation[63] for counterfactual (or potential) outcomes:

$$Y_i(E) - Y_i(C)$$

Where:

- $Y_i$ is the observed outcome value of interest from 'trial' $i$
- $E$ denotes the assignment of an experimental condition
- $C$ denotes the assignment of a control condition

Since, for any given trial $i$, it is impossible to observe both the treated ($E$) and the untreated ($C$) conditions, Rubin's notation directly compares the observed outcome of an event that did happen with the outcome of an event that did not, but could have (if the condition assignment had been otherwise).

In Rubin's 1974 article, he goes onto describe that causal inferences can be drawn from data, provided "there are no extraneous variables that affect $Y$ and systematically differ in the $E$ and $C$ groups" (Rubin 1974, 698) – he shows how this is as true of experimental data as it is of observational data. He writes: "the primary difference is that without randomization there is often a strong suspicion that there are such variables, while with randomization such suspicions are generally not as strong" (Rubin 1974, 698). It is here that Rubin first articulates an early form of what would later come to be called 'ignorability' (Rosenbaum and Rubin 1983), which refers to the property of the treatment assignment mechanism being independent of the outcome variable at every level of both. In other words: if we know that no unadjusted-for variable affected $P(x_i)$ *and* $P(y_i|x_i)$ for any $i$, then the treatment assignment mechanism is 'ignorable' and observational model results can be (cautiously) interpreted as though they were produced by experimental methods.

In this way, the potential outcomes framework and the graphical paradigm were tackling the same problem – that of drawing causal inferences from observational data – but both were doing so from opposite directions: the graphical paradigm sought valid causal mechanisms in observational data *qua* itself, whereas the Neyman-Rubin camp used experimental designs as a starting point and explored the conditions under which observational data could be treated

[61] Pun intended.

[62] My disdain for R.A. Fisher should, by this point, be painfully obvious. While I recognize that Fisher made many contributions to the field of statistics, I believe that Fisher's stultifying effect on the non-mathematical sciences – the yoke of which we are only now beginning to throw off – is not inconsiderable, even in light of his positive contributions. He shows up as a villain in both this chapter and in the chapter on Bayesian Inference for good reason: he actively worked to stamp out advocates of both causal inference and Bayesian probability (McGrayne 2011; Pearl and Mackenzie 2018).

[63] What follows isn't exactly what appears in Rubin's original 1974 article. He uses $j$ for the index subscript, and both instances of $Y$ are lower-case. I have adapted the notation in line with current convention for potential-outcome notation, including Rubin's later work. The general counterfactual form $Y_x(u)$ is often written $Y(x, u)$, as if this stuff wasn't already confusing enough (Pearl 2009b, 243).

as-if randomized (Rubin 2008).[64]

The volume and import of Rubin's contributions would grow throughout the 1970s and 1980s. During this period, Rubin would articulate two further assumptions implicit in the potential-outcomes framework.

The Stable Unit Treatment Value Assumption (SUTVA) merely reflects the assumption that the effect of the treatment doesn't depend on how widely it is applied (Rubin 1980b, 1990a). For example: receiving cataract surgery will alleviate symptoms associated with having cataracts regardless of whether 2 out of 100 people receive it, or 97 out of 100 people receive it. The same might not be so easily said of an after-school tutoring program for 7th-graders: the efficacy of such a program might depend on how many children regularly attend, and so the effect when only 3 children are assigned to the program might be quite different from that when 30 are.[65]

The consistency assumption ensures that any treatment given experimentally will have the same effect if received 'naturally' (VanderWeele and Hernan 2013). Here again: randomly assigned cataract surgery will almost certainly have the same effect as cataract surgery sought voluntarily. The after-school tutoring program, however, might affect children differently depending on whether or not they were forced to attend by experimental treatment, forced to attend by the 'natural' intervention of a parent, or sought the program of their own volition.

It is at this point that this selective, breakneck-speed overview of causal inference in the 20th century comes to an abrupt end. As of the late 1990s, the causal SEM project had foundered under sustained critique.[66] The Potential Outcomes framework – at this point – is well known, but Rubin's work on causality has failed to precipitate the desired sea change among many of the fields that regularly rely on statistical analysis (Pearl 2009b, 135).[67]

In the section that follows, I detail Pearl's attempt to revitalize the study and use of graphical causality by synthesizing the two causal paradigms.

## Structural Causality

In the early 1990s, a relative newcomer stepped into the causal arena. Judea Pearl – by that point already a well-known AI researcher – sought to build upon his existing body of work on Bayesian Networks for probabilistic reasoning (Pearl 1988) by extending it to the realm of formal causal inference (Pearl 1993; Pearl 1995 ).[69] Pearl did so by building on the insights of both the 'Graphical' and the Potential-Outcomes traditions, primarily borrowing the graphical emphasis of the former and the counterfactual notation of the latter. In so doing, Pearl broke with precedent and – arguably – his own mathematical background by presenting causality as a property of systematized *qualitative* knowledge, free from assumptions about distributions, functional forms, or most statistical properties.[70] It can be difficult, at first, to appreciate the enormity of this leap: prior to Pearl, all graphical causal models were burdened by intrinsic functional assumptions that – as discussed in the previous section – seriously harmed their validity when applied in empirical sociological settings.

To SEM scholars such as Blalock Jr (1961) and Duncan (1975), graphs such as Figure 11 were merely short-hand for a series of linear structural equations, which could be summarized as follows:

[64] Something Rubin views as very difficult to achieve: "This article advocates the position that observational studies for causal effects need to be designed to approximate randomized experiments. This enterprise requires careful thought and execution, and not simply running mindless regression programs and looking at coefficients. In most situations, this design effort will be more intellectually demanding than a similar effort for an analogous randomized experiment" (Rubin 2008, 837).

[65] The potential outcome framework is capable of handling cases where the SUTVA is violated. Doing so, however, is a much more odious task, both conceptually and mathematically (Morgan and Winship 2015; Cunningham 2021).

[66] Pearl writes:" Structural equation models are used by many, but their causal interpretation is generally questioned or avoided, even by their leading practitioners. […] The current dominating philosophy treats SEM as just a convenient way to encode density functions (in economics) or covariance information (in social science)" (Pearl 2009b, 135).

[67] In the contemporary words of Judea Pearl: "Currently, potential-outcome models are understood by few and used by even fewer." Notable exceptions here are economics and econometrics, both of which regularly employ the Potential Outcomes model (Imbens 2020). These exceptions do not undercut Pearl's claim that Rubin's work on causality has had limited reach – this is certainly true in Sociology[68]: while Rubin is cited a handful of times (in roughly 1/10th of the inferential quantitative articles reviewed), the vast majority of these nods are for his statistical work pertaining to non-causal topics, e.g. 'Rubin's Rules,' a set of guidelines for multiple imputation (Rubin 2004b).

[69] Readers familiar with the topic may take exception to my characterization of Pearl as standing alone in motivating this most recent advancement in the graphical paradigm of causal inference. Their criticism would be fair: Pearl's contemporaries – many of whom he worked with – were instrumental. For a more complete picture of the collaborative efforts behind the development of Structural Causal Models, interested readers can consult (Barringer, Eliason, and Leahey 2013).

[70] Compared to its predecessors, SCMs are shockingly light on assumptions, but they nevertheless must make a few: details about the Markovian assumptions powering SCMs can be found in (Pearl 2009b).

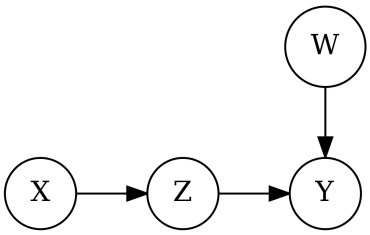


Figure 11: A causal graph.

$$
\begin{aligned}
X &= e_x \\
W &= e_w \\
Z &= \beta_{zx} X + e_z \\
Y &= \beta_{yz} Z + \beta_{yw} W + e_y
\end{aligned}
$$

To Pearl, Figure 11 is called a 'Structural Causal Model' (SCM) and represents much less, and much more. It represents less in the sense that it encodes no assumptions about the functional forms, which can be written like so:

$$
\begin{aligned}
X &= f_x(U_x) \\
W &= f_w(U_w) \\
Z &= f_z(X, U_z) \\
Y &= f_y(Z, W, U_y)
\end{aligned}
$$

Where each of the $f$s above can describe any imaginable relationship between the parameters and the outcome. It is in this sense that SCMs represent far more than their path-based ancestors, as shedding SEM's statistical assumptions permits a degree of flexibility and generality not before seen in any strand of graphical causal modelling.[71] The result is a much more powerful tool for the articulation of causal assumptions and the derivation of the implications thereof (Bollen and Pearl 2013): Pearl writes that "graphs provide a fundamental notational system for concepts and relationships that are not easily expressed in the standard mathematical languages of algebraic equations and probability calculus" (Pearl 2009b, 138).

In the subsections that follow, I provide a brief overview of Pearl's graphical theory of inferred causation.

## Structural Causal Models

### Edges Track Causality

The nodes in an SCM are joined by directional edges that, where present, indicate a causal relationship between nodes. (Pearl, Glymour, and Jewell 2016) use an auditory metaphor: node $A$ determines its value (in part or in whole) by 'listening' to the other node(s) with an edge pointing into node $A$.

All SCMs are Directed Acyclic Graphs (DAGs). 'Directed,' in this case, indicates that all of the edges in the graph are unidirectional, as indicated by the presence of a tail (the edge's origin) and head (the edge's destination). 'Acyclic' refers to the lack of cycles in the graph: this implies that a variable cannot cause itself – directly or indirectly – and that mutual causality is impossible.

[71] The significance of this move can be hard to grasp at first. This is especially so for sociologists, who are used to thinking of statistical relationships in terms of ordinary least-squares regression models. Doing so here runs the risk of needlessly circumscribing the flexibility afforded by Pearl's free-form $f$s. Morgan and Winship write: "An implication of this flexibility deserves particular emphasis, and preexisting knowledge of traditional path models and their implicit additive structural equations can hinder full recognition of its importance. […] even though it may feel natural to want to 'see' a specific arrow present in the causal graph to represent an interaction effect that corresponds to a cross-product term in a regression equation, one must learn to suppress such a desire. The key point, in considering an analysis that utilizes causal graphs, is to drop regression models from one's mind when thinking about identification issues" (2015, 89–90).

### Nodes are Functions

Each node in an SCM represents a variable. They determine their values by 'listening' to each of the other nodes that send an edge to them. For some, it may be helpful to think of each node in a SCM as representing a function whose arguments are output values from other nodes. For a treatment $R$, an outcome $Y$, and a confound $W$, writing a functional-form SCM might produce something akin to Figure 12.[72]

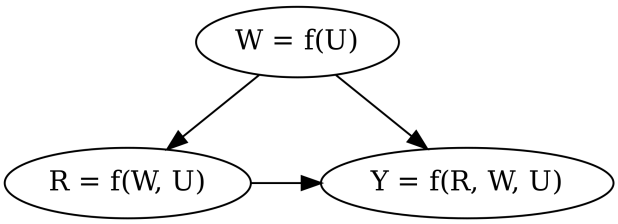


Figure 12: Confounding SCM for R, Y, and W with explicit unobserved influences.

You can read the above as follows:

- $W$ is wholly determined via the influence of unknown, unobserved variables ($U$)
- $R$ is determined in part by the value of $W$ and in part by other unknown, unobserved variables ($U$)
- $Y$ is determined by some function of the values of $R$, $W$, and other unknown, unobserved variables ($U$)

The specific form of the functions ($f$) in the diagram above aren't important for the purposes of causal inference. The edges in an SCM imply causality, but they do not encode any assumptions about the form of the relationship between the variables so joined.[73]

### Correlation vs Causation

There is an intrinsic tension present in every SCM. In the world of causation, directionality matters: causative influence flows *exclusively* from the tail of an edge to its head; never the reverse. In the world of correlation, however, the arrowheads effectively disappear: correlation flows backwards along an edge just as well as it flows forwards. This represents a major issue: as social scientists, we have no choice but to use correlative tools (in the form of statistical models) to estimate causal effects, but we cannot imbue our tools with the ability to differentiate causative effects from mere correlation: only science and scientists can do this (McElreath 2020). Our objective, then, is to use an SCM to judiciously identify a set of variables that – when adjusted for[74] – will prevent any important correlation from 'flowing' erroneously. Adjusting for a node in the context of an SCM effectively alters how that node transmits influence. In most cases, adjustment prevents a node from passing information along. In some cases, however, adjustment can have the opposite effect.

Structural Causal Models can take on a dizzying array of permutations, but each SCM can be decomposed into a set of three different triads, each of which demands a different handling method. I'll cover them below. In each example, I'll propose a setting wherein we are interested in assessing node $A$'s causal influence on node $C$, where $A$ and $C$ share no direct connection but are linked via a third node, $B$. They are:

### The Pipe

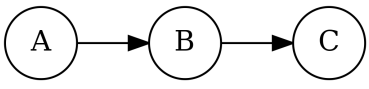


Figure 13: SCM of A, B, and C in a pipe configuration.

Node $A$ causes $B$, which – in turn – causes $C$. In this example, $A$ and $C$ are correlated through $B$'s ability to 'pipe' information from $A$ to $C$. For any pipe consisting of $A \rightarrow B \rightarrow C$ with treatment $A$ and outcome $C$, the indirect influence

[72] Note that each lower-case 'f' in Figure 12 represents a different function.
[73] As discussed above, this is a critical point and should not be overlooked.
[74] Using any valid means of 'adjustment' – including 'controlling', as in the case of a regression model.

of $A$ on $C$ is genuine, and – hence – causal. If we adjust for $B$, $A$ and $C$ would become (conditionally) independent of one another, as all of $A$'s causal influence would be captured by $B$.

By way of a brief example: imagine $A$ represents individual's alcohol consumption over the course of a 2-hour bender, $B$ tracks that same individual's blood alcohol level at the end of the two-hour period, and $C$ indicates whether or not that individual crashed their car whilst attempting to drive home immediately following their ablutions. We would expect all three variables to be positively correlated with one another. We would also expect $A$ and $C$ to become independent of one another after adjusting for $B$, as $B$ moderates all of the information transmitted from $A$ to $C$: in this case, once we know someone's blood alcohol level ($B$), we've already learned everything we can about that person's chances of crashing ($C$) – additional information about how much alcohol they consumed ($A$) won't influence our predictions because $A$ is now independent of $C$.

In the example above, one should *not* close the pipe by adjusting for $B$. That doesn't, however, mean that one should *never* close a pipe. Consider the case below:

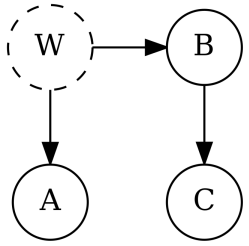


Figure 14: Confounding SCM of A, B, W, and C

In this case, the path from $A$ to $C$ in $A \leftarrow W \rightarrow B \rightarrow C$ is non-causal (as $W$ is the cause of both $A$ and $B$; any apparent causative link between $A$ and $C$ is spurious). If we knew of $W$'s influence, but could not – for some reason – observe it, we could prevent the influence from biasing our estimate by closing the pipe $W \rightarrow B \rightarrow C$, which could be accomplished by adjusting for $B$.

**The Fork**

If we switch the direction of the leftmost edge in the Pipe example above, we end up with the Fork (see Figure 15).

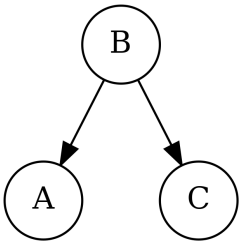


Figure 15: SCM of A, B, and C in a fork configuration.

Absent any further modifications or information, we would expect $A$ and $C$ to be correlated as a result of $B$'s' mutual influence. For any fork consisting of $A \leftarrow B \rightarrow C$ with treatment $A$ and outcome $C$, $A$ does not cause $C$. The correlation between $A$ and $C$ is due to the presence of a common cause $B$ and should not be interpreted causally. Adjusting for $B$ should render $A$ and $C$ (conditionally) independent of one another.

When spurious associations appear, the grim hand of the fork is often at work. Imagine that $A$, in this example, represents the number of assault felonies committed in a given city during a given day. $C$ represents the number of creamsicles sold from ice-cream trucks throughout the same city on the same given day. Taken in isolation, the two appear to be correlated – an incautious researcher might conclude that consuming orange-flavoured desserts somehow induces misanthropic rage. Once we adjust for the common cause, $B$ (which can stand for 'barometric conditions'), $A$ and $C$ will become (conditionally) independent. The explanation here is that warm weather ($B$) causes higher ice cream sales ($C$) *and* higher

violent crime rates ($A$): once we've factored in information about the weather, information about ice cream sales can't tell us anything about violent crime.

In this example, if we were interested in learning about $A$'s influence on $C$, it would be in our interest to close the non-causal path $A \leftarrow B \rightarrow C$ by adjusting for $B$.

#### The Collider

We've saved the trickiest for last. When two variables both directly cause a third, a collider is formed (see Figure 16).

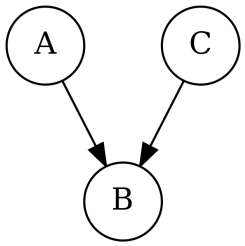


Figure 16: SCM of A, B, and C in a collider configuration.

Distinct from pipes and forks, colliders do not – by default – imply a correlation between nodes $A$ and $C$. Both $A$ and $C$ have a causal influence on $B$, but $A$ and $C$ are (unconditionally) independent; you cannot learn anything about one by examining the other. All of this changes, however, when one adjusts for $B$: by holding $B$ at some constant value, knowledge of $A$ implies some knowledge of $C$ (and vice versa).

Collider bias can be induced when a collider ($B$, in our example) is adjusted for; this frequently occurs when some kind of ranking or recommendation system considers two or more pieces of uncorrelated information. For example: a trusted niche publication puts out an annual 'Best-Of' list which enumerates the editor's top picks for video games that were both enjoyable ($A$, standing for 'A Lot Of Fun') and which did a good job of evoking a Gibsonian aesthetic ($C$ for 'Cyberpunk'). In general, game enjoyability is uncorrelated with adherence to a neuromancer-like setting; learning about one does not provide any information about the other. An induced correlation appears, however, once a game has been included on the publication's 'Best-Of' list ($B$); this is because the publication is only willing to include games that are: - of surpassing enjoyability (very high $A$) and at least middling Gibsonian-ness (average $C$), - of exceptional Gibsonian-ness (very high $C$) and at least passable enjoyability (average $A$), - of at least good Gibsonian-ness (above-average $C$) *and* enjoyability (above-average $A$)

If we assume that enjoyability and Gibsonian-ness are both normally distributed and uncorrelated, games that score very highly on both $A$ and $C$ are *exceedingly* rare. In light of this, knowledge of a game's enjoyability ($A$) and its inclusion on the best-of list ($B$) allows us to deduce something about its Gibsonian-ness ($C$): if it's high on $C$ and included on $B$, then it's likely to be lower on $A$ (and vice versa). This correlation is, of course, non-causal; a game's adherence to a cyberpunk aesthetic does not *cause* the game to be enjoyable. Unlike the Pipe and the Fork, however, Colliders are closed by default. Only when the middle node in a collider is adjusted for does it begin to transmit non-causal information.

Together, the Pipe, the Fork, and the Collider are the building blocks of every directed acyclic graph. Taking their properties into account, we are ready to follow Pearl down a chain of theorems that will eventually bring us to a complete, general-purpose theory of inferred graphical causation.

## *d*-separation

As we saw above, the directional relationship between a set of three nodes implies certain conditional and unconditional dependencies: the Pipe and the Fork imply unconditional dependence (with conditional independence if the middle node is adjusted for), whereas the Collider implies unconditional independence (with conditional dependence if the middle node or any of its descendants is adjusted for). Using just this information, we are now able to make statements about dependence between any two nodes in a given graph, $G$. If the arrangement of nodes and adjustments is such that two nodes are independent of one another, we can say that they are 'directionally separated', or '*d*-separated' for short (Pearl 2009b).

The formal definition, contained in Pearl (2009b, 17), of *d*-separation is as follows:

> A path $p$ is said to be *d*-separated (or blocked) by a set of nodes $Z$ if and only if:
>
> 1. $p$ contains a chain $i$->$m$->$j$ or a fork $i$<-$m$->j such that the middle node is in $Z$
> 2. $p$ contains an inverted fork (or collider) $i$->$m$<-$j$ such that the middle node $m$ is not in $Z$ and such that no descendent of $m$ is in $Z$

The importance of $d$-separation cannot be overstated: in a given causal graph $G$, if any two nodes $A$ and $B$ are *d*-separated by set $Z$, then $A$ and $B$ are independent of one another conditional on $Z$. The corollary also holds: nodes that are not *d*-separated in $G$ are almost certain to be dependent (Pearl 2009b, 18).

One of the most significant byproducts of *d*-separation is that every graphical model carries with it a series of 'testable implications,' in the form of conditional/unconditional dependencies/independencies.[75] In the graphs used to demonstrate the pipe and the fork, for example, we can derive the following testable implications:

1. $A \not\perp\!\!\!\perp C$
2. $A \not\perp\!\!\!\perp B$
3. $B \not\perp\!\!\!\perp C$
4. $A \perp\!\!\!\perp C|B$

In short, the above set of statements claim that all of the variables in both the pipe and the fork configuration are dependent (because they are not *d*-separated). When $A$ and $C$ are conditioned on $B$, however, $A$ and $C$ become *d*-separated, and – thus – conditionally independent.

The pattern of independencies implied by *d*-separation can be tested using observational data believed to be drawn from the data generating process described by the graph. If any of them are violated, the graph can be discarded.[76] In this sense, Pearl's SCMs are subject to a sort of Popperian falsifiability: their veracity can be challenged or refuted using observational data alone (Popper 2009).[77]

On the basis of *d*-separation, Pearl develops a set of rules for making equivalence statements about conditional probabilities in manipulated graphs which he calls 'the *do*-calculus'.

## *do*-calculus

Pearl developed *do*-calculus to address what he viewed as a shortcoming in statistical notation circa 1993 (Pearl 1993, 2009b; Pearl, Glymour, and Jewell 2016). Pearl had observed that those working with structural equations (e.g. $Y = f_Y(X, U)$) had to rely on intuition and context to understand that the equality symbol in such equations wasn't supposed to behave the way most equality symbols do. Structural equations were designed to show that the value of $Y$ depended on some (usually unknown) function of two variables: the endogenous, observed $X$, and the exogenous, unobserved $U$. Manipulating the value of $X$ (as in an experimental setting) would, in such a contrivance, presumably alter the value of $Y$ according to the functional form of $f$. The inverse, however, was not true: inverting $f$ and manipulating $Y$ should not produce any change in $X$, as $X$ does not depend on either $Y$ or $U$ for its value.

The other notational shortcoming was in regards to conditional probabilities. Consider the following statements:

1. The probability of $Y = y$ given the observed value of $X = x$ is $P(Y = y|X = x)$
2. The probability of $Y = y$ given the randomized treatment $X = x$ is $P(Y = y|X = x)$

The mathematical statements of probability may be identical, but do they describe the same probability? How does observing $X = x$ in a natural setting differ from setting it to a fixed value $X = x$ for all participants in an experiment? Pearl's solution to this ambiguity was to introduce the $do(\cdot)$ operator: any variable enclosed inside a $do(\cdot)$ operator is considered 'fixed,' as though by experimental treatment assignment. Using this new operator, the observational probability

[75] For the remainder of the chapter, independence between two variables will be indicated using the following symbol: $\perp\!\!\!\perp$. Dependency will be indicated using a crossed-out version of the independence symbol, like so: $\not\perp\!\!\!\perp$.

[76] For that population or sample, at least – the graph may still be valid for a different population or sample.

[77] In practical settings, measurement error and construct validity can blur the line between independence and dependence somewhat.

expressed above would remain unchanged, but the experimental probability would now be written $P(Y = y|do(X = x))$.[78]

Aside from notational clarity, the $do(\cdot)$ operator permits us to ask causal questions of observational data. For example: "if I observe the probability distribution $P(Y|X)$, what would be the change in the value of $Y$ if I intervened upon $X$ first to make it take on a value of 0, and then 1?" We could write this query as follows:

$$P(Y = y|do(X = 1)) - P(Y = y|do(X = 0))$$

Note that the above equation (which describes the effect of artificially intervening on $X$) describes a set of observations entirely distinct from the $do(\cdot)$-free equivalent (which describes the difference between cases where $X$ naturally took on the value 1 and 0):

$$P(Y = y|X = 1) - P(Y = y|X = 0)$$

Now we arrive at the core of Pearl's theory of inferred graphical causation. Using with the $do(\cdot)$ operator, Pearl developed a system of rules that describe how one can derive causal claims from a given graph, $G$. This system is called the *do*-calculus.

To use the *do*-calculus, one begins by specifying a causal DAG, $G$, describing the data generating process that gave rise to the dataset under analysis.[79] For the purposes of this section, we'll use the DAG in Figure 17, where $X$ and $Y$ are the variables of interest, $W$ is an observed confounder, $U$ is an unobserved confounder, and $Z$ is an observed moderator.

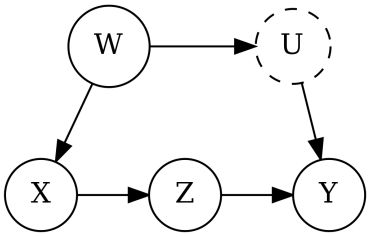


Figure 17: $G$, a causal DAG.

Then, we ask a question about the change in the probability of $Y$ conditional on an intervention on $X$, such as this one:

$$P(Y = y|do(X = 5)) - P(Y = y|do(X = 2))$$

The question, in mathematical form, makes use of the *do*-operator: as such, its value cannot be directly calculated from observational data, as the *do*-operator cannot be evaluated numerically. This is where the *do*-calculus steps in: the *do* calculus consists of three rules that permit manipulation of the desired conditional probability on the basis of graphical criteria.

Each of the rules assumes four arbitrary sets of nodes in a causal graph (labelled here as $A$, $B$, $C$, and $D$), which contain non-overlapping observed nodes in the graph (each observed node appears in at most one set). The upper-case letters are used to refer to the entire set of nodes, whereas lower-case letters are used to represent a specific combination of values taken on by each of the nodes in the set. Lower-case letters with a 'hat' represent intervened-upon variables (as in an experimental treatment), whereas lower-case letters without the 'hat' are merely observed. The 'hat' notation is used here to save space and visually simplify the equations that follow: each hat variable can be interpreted as being wrapped in a $do(\cdot)$ operator, and the un-hatted variables are raw observational probabilities: the statement $P(a|\hat{b}, c, d)$ below can be interpreted as being fully equivalent to $P(A = a|do(B = b), C = c, D = d)$.

Each of the rules depends on one or more manipulated graphs, meant to indicate the implications of intervening upon one or more nodes in the network. The manipulated version of graph $G$ corresponding to experimental intervention on

[78] Throughout his work, Pearl occasionally uses a $see(\cdot)$ operator to expressly indicate an observed value. It is omitted in this chapter.
[79] Or the researcher's best guess about it, at any rate.

node(s) $X$ – which would take the form of removing all edges pointing into $X$ – would be written as $G_{\overline{X}}$. A graph where all of the edges leading out of $X$ have been removed is written as $G_{\underline{X}}$.

Note that in each of what follows, only set $A$ needs to be non-empty: if $B$ or $D$ are empty, they simply drop out of the rule.[80] If $C$ is empty, no change to the conditional probability statement is made.

**Rule 1 (addition/removal of observation $c$):**

$$P(a|\hat{b}, c, d) = P(a|\hat{b}, d), \text{ if } (A \perp\!\!\!\perp C|B, D) \text{ in } G_{\overline{B}}$$

Rule 1 tells us that if $A$ and $C$ are *d*-separated by $B$ and $D$ in the graph $G_{\overline{B}}$, then we can safely add or remove the observed variables $C$ without affecting the probability of $a$ conditional on $\hat{b}$ and $d$.[81]

**Rule 2 (changing $c$ from action to observation, or vice versa):**

$$P(a|\hat{b}, \hat{c}, d) = P(a|\hat{b}, c, d), \text{ if } (A \perp\!\!\!\perp C|B, D) \text{ in } G_{\overline{B}\underline{C}}$$

Rule 2 states that if $A$ and $C$ are *d*-separated by $B$ and $D$ in the manipulated graph $G_{\overline{B}\underline{C}}$ (with all edges into $B$ removed, as well as all those leading out of $C$), then we can freely denote $c$ as either being a set of intervened-upon variables ($do(C = c)$), or as a set of observations ($C = c$).

**Rule 3 (addition/removal of action $c$):**

$$P(a|\hat{b}, \hat{c}, d) = P(a|\hat{b}, d), \text{ if } (A \perp\!\!\!\perp C|B, D) \text{ in } G_{\overline{B},\overline{C(D)}}$$

Where $C(D)$ is the set of nodes in $C$ that are *not* parents/ancestors of any node in $D$ in the graph $G_{\overline{B}}$.

Rule 3 states that if $A$ and $C$ are *d*-separated by $B$ and $D$ in *both* of the manipulated graphs $G_{\overline{B}}$ and $G_{\overline{C(D)}}$, then the intervened-upon variables $C$ can be removed from the probability expression without consequence.

Returning to our original query about the difference in the probability of $Y$ if we were to intervene so that $X = 5$ and then $X = 2$: we'll need to use one or more of the rules from the *do*-calculus to turn our current query, which contains a *do*-operator (or, equivalently, a hatted $\hat{x}$), into one that does not contain any *do*-operators (or hats). In other words, we have to find a formula that allows us to express our interventional query ($P(Y = y|do(X = x))$) using only conditional probabilities (e.g.

$$P(Y = y|X = x)$$

).

To accomplish this, we want to use Rule #2 to change $X$ from an action ($\hat{x}$ or $do(X = x)$) to an observation ($x$ or $X = x$), but we can't do so right away: Rule #2 demands that the variable on the left-hand side of the 'conditional' sign is independent of the action/observation in $G_{\overline{B}\underline{C}}$.[82] Since, in this case, $C = \{x\}$ and $B$ is an empty set, $G_{\overline{B}\underline{C}}$ takes on the configuration depicted in Figure 18.

Since $X$ and $Y$ are joined by the open path $X \leftarrow W \rightarrow U \rightarrow Y$, they are not *d*-separated and – thus – not independent ($X \not\perp\!\!\!\perp Y$). To get the ball rolling, we need to add a variable to set $B$, which we can do by adjusting for an observed variable in the graph. Adjusting for either $U$ or $W$ would do the trick, but we can't adjust for $U$ because it is unobserved; we'll use $W$ by necessity. Adjusting on $W$ involves stratifying our data on every level of $w \in W$, which we can write as follows:

$$P(y|\hat{x}) = \sum_{w} P(y|\hat{x}, w)P(w|\hat{x})$$

[80] This implies that manipulated graphs involving the empty set are equivalent to their un-manipulated counterparts: ($G_{\overline{X}} = G$ if $X = \{\}$)

[81] The removal of all edges into $B$ is licensed by the fact that we are treating it as though it were an intervened-upon variable. This is true for each of the *do*-calculus rules, but doesn't necessarily imply that $B$ was actually controlled in experimental settings. Following the *do*-calculus allows us to treat certain variables as if they had been experimentally controlled.

[82] It's worth noting that if we were interested in deriving a formula for $P(z|\hat{x})$, we'd be able to use Rule #2 right away: the path $X \leftarrow W \rightarrow U \rightarrow Y \leftarrow Z$ is already blocked by the collider on $Y$, which means we don't need to adjust for anything and can swap $\hat{x}$ for $x$ with no additional steps.

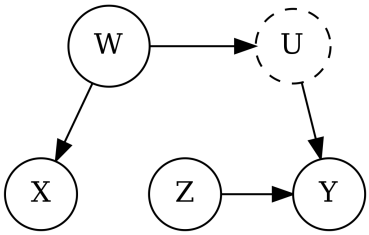


Figure 18: $G_{\overline{B}\underline{C}}$, a manipulated causal DAG with all edges emanating from $X$ removed.

Now the right-hand side of the equation has two $do(\cdot)$ operators. This might seem counterproductive at first, but it does represent progress! Having made this move, we can easily dispatch with the $\hat{x}$ in $P(w|\hat{x})$ using Rule #3. To check this, we'll rewrite Rule #3 substituting $w$ for $a$ and $x$ for $c$ (and dropping the empty sets $b$ and $d$):

$$P(w|\hat{x}) = P(w), \text{ if } (W \perp\!\!\!\perp X) \text{ in } G_{\overline{X}}$$

We'll examine $G_{\overline{X}}$ – depicted in Figure – to see if our use of Rule #3 is warranted. We can see from this graph that there is only one path joining $X$ and $W$: $X \rightarrow Z \rightarrow Y \leftarrow W$. Since there is a collider on $Y$, the path is closed by default, and $W$ and $X$ are independent in $G_{\overline{X}}$.

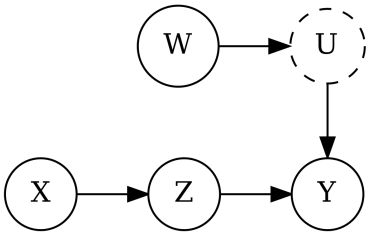


Figure 19: $G_{\overline{X}}$, a manipulated causal DAG with all edges running into $X$ removed.

Here's what we're left with:

$$P(y|\hat{x}) = \sum_{w} P(y|\hat{x}, w)P(w)$$

Only one $do(\cdot)$ remains, and we are ready to apply Rule #2 to it (again, dropping the empty set $B$ from the original rule):

$$P(y|\hat{x}, w) = P(y|x, w) \text{ if } (Y \perp\!\!\!\perp X|W) \text{ in } G_{\underline{X}}$$

To check if our use of Rule #2 is warranted, we must create a manipulated version of our original graph, this time with all of the edges emanating from $X$ removed. The result can be seen in Figure 20.

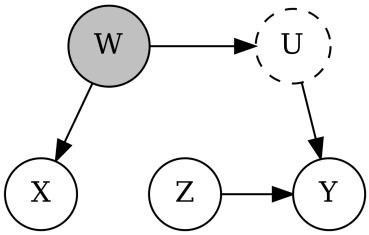


Figure 20: $G_{\underline{X}}$, a manipulated causal DAG with all edges emanating from $X$ removed and $W$ adjusted for.

Finally, using $G_{\underline{X}}$, we can test whether or not $Y$ is independent of $X$ conditional on $W$ using $d$-separation. The 'front-door' path $X \rightarrow Z \rightarrow Y$ in $G$ has been severed in $G_{\underline{X}}$, so it is not of any concern. The path $X \leftarrow W \rightarrow U \rightarrow Y$ is intact, but does not represent a threat to identification: since we are adjusting for $W$, the path is blocked. As such, we can conclude that $Y$ is independent of $X$ conditional on $W$ in $G_{\underline{X}}$, and thus the use of Rule #2 is justified. The result is a mathematical statement capable of estimating the effect of an experimental intervention on $X$ using only observational data:

$$P(y|\widehat{x}) = \sum_{w} P(y|x,w)P(w)$$

The answer, then, to our original query[83] is:

$$\left[\sum_{w} P(Y=y|X=5,W=w)P(W=w)\right] - \left[\sum_{w} P(Y=y|X=2,W=w)P(W=w)\right]$$

Pearl's *do*-calculus can be used to derive one or more adjustment formulae for any viable causal query asked of a causal graph (Shpitser and Pearl 2006; Huang and Valtorta 2012).[84]

**Identifiability**

Pearl's contributions to the causality literature are primarily motivated by the desire to determine which interventional queries are 'identifiable.' The effect of an intervention is said to be 'identifiable' relative to a graph $G$ if there exists some set of operations that permits the recovery of an unbiased estimate of that effect (Pearl 2009b, 77). Together, the three rules of Pearl's *do*-calculus are 'complete,' which is to say: any interventional statement $P(A = a|do(B = b))$ made in relation to any causal graph $G$[85] is identifiable *if and only if* said statement can be successfully solved for using the *do*-calculus. Multiple teams have independently proven that this is so (Shpitser and Pearl 2006; Huang and Valtorta 2012).

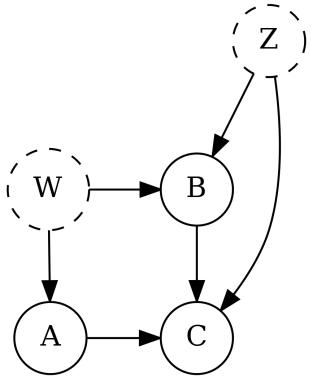


Figure 21: A Non-Identifiable SCM

Not all interventional statements are identifiable. A single node might, for instance, serve as a pipe for a non-causal path *and* a collider, as is the case with node $W$ in Figure 21. If we are interested in estimating the causal effect of $A$ on $C$, we would first have to account for the spurious path $A \leftarrow W \rightarrow B \rightarrow C$. Since $A$ and $C$ are our variables of interest, and $W$ is unobserved, our only choice here is to adjust for $B$ to close the path. Doing so, however, merely serves to create a different problem: adjusting for $B$ has opened the collider $W \rightarrow B \leftarrow Z$, which opens the non-causal path $A \leftarrow W \rightarrow B \leftarrow Z \rightarrow C$, which can't then be closed owing to the fact that $Z$ is unobserved.[86]

[83] For this final equation, I've elected to switch back to Pearl's extended notation; doing so helps clarify which variable is taking on the values of 5 and 2, respectively.

[84] This also implies that if the *do*-calculus cannot produce an adjustment formula, then the causal query cannot be 'identified.' See: section on 'identifiability' below.

[85] Where $A$ and $B$ are arbitrary sets of nodes in $G$

[86] Lattimore and Rhode (2019a, 2019b) argue that Bayesian causal inference can produce estimates from non-identifiable SCMs, but these estimates are very sensitive to priors. Non-identifiable SCMs may not yield reliable inferences, but they do gesture towards steps that might be taken to solve the issue, including gathering data on unobserved variables, seeking and including an instrumental variable (Pearl, Glymour, and Jewell 2016), etc.

Reassuring though it may be to know that the *do*-calculus can tell us which causal queries are identifiable, deriving adjustment formulae from graphs of even modest complexity can become dauntingly complex. Fortunately for those of us lacking graduate-level training in mathematics, the possibilities engendered by the *do*-calculus in the context of most causal graphs can be distilled into two "shortcuts."

### Back-Door Criterion

The first – and most widely-applicable – shortcut describes the conditions under which confounding paths can be closed off using observed data. It is called the 'Back-Door Criterion' because it prevents paths with arrows pointing into $X$ (via a back-door route of sorts) *and* $Y$ from biasing the estimate of causal effect. It does so by adjusting for a minimally-sufficient set of variables such that all of the back-door paths become blocked.

Formally, Pearl, Glymour, and Jewell (2016, 61) express the back-door criterion as follows:

> Given an ordered pair of variables $(X, Y)$ in a directed acyclic graph $G$, a set of variables $Z$ satisfies the backdoor criterion relative to $(X, Y)$ if no node in $Z$ is a descendant of $X$, and $Z$ blocks every path between $X$ and $Y$ that contains an edge into $X$.

If the back-door criterion is satisfied, it yields the following formula:

$$P(Y = y|do(X = x)) = \sum_{z} P(Y = y|X = x, Z = z)P(Z = z)$$

If the above formula looks familiar, that's because it's nearly identical to the adjustment formula we arrived at in the worked example above.

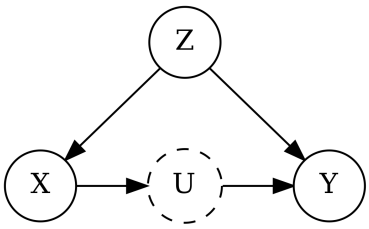


Figure 22: SCM demonstrating the utility of the back-door criterion

The formula implies that if a set of variables $Z$ is sufficient to close off all 'back-door' (a.k.a. spurious) paths between $X$ and $Y$, then the $do(\cdot)$ operator enclosing $X$ can be safely removed by stratifying on each combination of values the variables in $Z$ can take. Consider Figure 22: since node $Z$ is observed, it constitutes a minimally-sufficient adjustment set by itself. The fact that $U$ is unobserved, in this case, poses no threat to identifying the effect of $X$ on $Y$.

### Front-Door Criterion

Despite the back-door criterion's obvious utility, many models will contain back-door paths that cannot be closed off by adjustment. The causal graph depicted in Figure 23 is one such model – attempting to apply the back-door criterion fails to produce an identifiable effect, as $U$ is unobserved and thus cannot be adjusted for. Fortunately, an alternative 'front-door' approach exists: it takes advantage of variable sets that fully capture the effect of the treatment on the outcome.

Formally, Pearl, Glymour, and Jewell (2016, 82) outline the following conditions for the front-door criterion to hold:

> A set of variables $Z$ is said to satisfy the front-door criterion relative to an ordered pair of variables $(X, Y)$ if:
>
> 1. $Z$ intercepts all directed paths from $X$ to $Y$.
> 2. There is no unblocked back-door path from $X$ to $Z$.
> 3. All back-door paths from $Z$ to $Y$ are blocked by $X$.

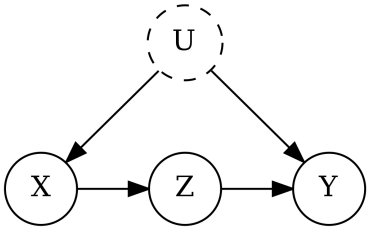


Figure 23: SCM demonstrating the utility of the front-door criterion

If the front-door criterion is satisfied, it yields the following formula:

$$P(Y = y|do(X = x)) = \sum_{z} P(Z = z|X = x) \sum_{x'} P(Y = y|X = x', Z = z)P(X = x')$$

Where $x'$ is the index of all values $X$ can take. The front-door formula tells us that we can replace $do(x)$ by first determining the effect of $X$ on $Z$, and then propagating that effect through $Z$'s effect on $Y$. The intuition here is that if $Z$ captures all of the effect $X$ has on $Y$ (condition 1), $X$ and $Z$ are not themselves confounded (condition 2) – and nor are $Z$ and $Y$ (condition 3) – then the effect of $X$ on $Y$ is identifiable.

#### Good and Bad Controls

The preceding discussion has focused on considering minimally-sufficient sets of variables who, once adjusted for, are capable of identifying a causal query. Sociologists may nevertheless find themselves with a large set of variables that prove unnecessary for identifying the causal effect of interest. Received statistical wisdom holds that variables should be adjusted for[87] provided they are 'pre-treatment' variables. While this is generally true, *do*-calculus can be used to illustrate cases where the inclusion of a pre-treatment variable can – for example – induce collider bias or harm precision.

Using a causal graphical model permits us to speak meaningfully of "good" and "bad" controls (Cinelli, Forney, and Pearl 2021). Good controls are part of a minimally-sufficient adjustment set for identifying a causal effect,[88] whereas "bad" controls induce bias. Even if a control is neither 'good' nor 'bad,' the decision to include it is not trivial: some variables (typically those that cause the outcome variable, but not the treatment) can permit more precise estimates, whereas others (typically those that cause the treatment variable, but not the outcome) accomplish the inverse. Those who are keen to peer into the pool of accumulated wisdom about how various controls impact inference would do well to familiarise themselves with "A Crash Course in Good and Bad Controls" (Cinelli, Forney, and Pearl 2021).

## Transparency and Generality

How do we know that a causal model is 'correct'? The short answer is: "we don't." As far as I am aware, there is no way to confirm the veracity of a causal model, and I highly doubt that any method for doing so will be discovered in the foreseeable future. This, however, should not be cause for alarm. Throughout, I have spoken about causal models in general – and SCMs in particular – as being languages for describing the sets of assumptions[89] that motivate and enable statistical estimation of an empirical effect. Since one cannot draw statistical inferences without first making assumptions (McElreath 2020),[90] it's reasonable to conclude that sociologists already habitually make strong assumptions about causal relationships, functional forms, conditional independencies, and so on. Making strong assumptions is a perfectly acceptable way to conduct empirical research, provided those assumptions are communicated exhaustively and transparently. As such, sociologists should not feel that they must develop a perfect causal model before they can estimate empirical effects. Rather, they should endeavour to use domain knowledge, theory, and extant literature to create the best, most complete causal model they can, and then test the implications and estimate the identifiable effects that can be

[87] In the context of inferential quantitative sociology, this is typically accomplished by adding said variables to a regression model.

[88] A given causal query can, in the context of a given graph $G$, have several overlapping minimally-sufficient adjustment sets.

[89] Most of which are causal, but some of which – depending on the language – are not.

[90] I'm certain this statement has been made elsewhere, in nearly as many words, but I'm specifically drawing it from McElreath (2020), who – on page 553 – states: "There is no inference without assumption, but do not choose your assumptions for the sake of inference."

drawn therefrom. The key here is transparency: causal assumptions must be communicated as clearly as possible so as to facilitate critique, dissent, or additions from interlocutors in the field.[91]

It is my belief that, out of the frameworks currently available, Pearl's approach to causal inference is the most transparent and generally-applicable: as such, it is the most appropriate for the field of sociology.[92] Pearl's theory of inferred causation offers causal models that are more transparent[93] and inculcate a more powerful set of falsifiable implications than equivalent models created using Rubin's Potential Outcomes framework (Pearl 2009b, 283).[94] Pearl's framework also addresses the shortcomings inherent to 20th-century Path Analysis and SEM by offering generalized non-parameteric causal models whose implied effects can be estimated by almost any statistical technique capable of computing conditional probability[95] – including non-parametric SEM (Bollen and Pearl 2013).[96]

My endorsement of Pearl's causal inference framework is primarily motivated by my desire to see causal graphs become a *lingua franca* of sociological theory and model building. To that end, I believe Pearl's framework represents the most transparent and general approach currently available, and is thus most likely to see widespread use in sociology. In no way do I wish to imply that other approaches to causality are somehow fundamentally deficient: the Potential Outcomes framework and traditional Path Analysis/SEM are perfectly viable, conditional on somewhat more restrictive assumptions (in the case of linear SEM) and/or somewhat less transparent models with fewer testable implications (in the case of Potential Outcomes). Nevertheless: provided sociologists are willing to clearly, exhaustively, and rigorously state their casual assumptions and demonstrate that their approach to estimation is consistent thereto, they should feel at liberty to employ any of the available causal languages.

I also do not wish to imply that this dissertation unreservedly endorses Pearl's philosophical view of causality. While I find his writing on the topic compelling, I believe the matter has not yet been satisfactorily settled. As of this dissertation's publication, the metaphysics of causality continues to sustain a lively academic debate. I nevertheless believe that causal graphs – and Pearl's causal framework in particular – are widely applicable and can be productively employed by practicing sociologists regardless of the particular causal metaphysics they espouse. Adherents of Nancy Cartwright (2004, 1999a, 1999b), for example, might be compelled to create far more detailed and complete causal graphs than those Pearl would produce, but their differences are largely a matter of scale and specificity – Pearl's language of variables, dependencies,

[91] Perhaps Richard McElreath says it best: "There is no method for making causal models other than science. There is no method to science other than honest anarchy" (McElreath 2021b).

[92] Subsequent contributions have built upon Pearl's theory of inferred causation, adding new terminology, definitions, theorems, notation, and graphical components (see: Malinsky, Shpitser, and Richardson 2019). Any widely-accepted permutation of Pearl's causal inference framework should be equally suitable.

[93] I identify Pearl's framework as offering a transparency advantage over Rubin's primarily due to the former's emphasis on separating and visualizing causal assumptions before assessing the logical consequences of those assumptions. While Rubin's 'ignorability' and Pearl's 'identifiability' are – in theoretical cases with perfect information – mathematically equivalent, the two are behaviourally distinct in applied settings. Pearl writes: "As stated several times in this book, the opacity of 'ignorability' is the Achilles' heel of the potential-outcome approach – no mortal can apply this condition to judge whether it holds even in simple problems, with all causal relationships correctly specified, let alone in partially specified problems that involve dozens of variables" (Pearl 2009b, 350). Because the identifiability of an effect is an emergent property of the causal assumptions made in an SCM, disagreements over the validity of an SCM can focus on individual assumptions, rather than the validity of the effect itself. Ignorability, conversely, is often assumed directly and is not composed of component assumptions (Pearl 2009b, 347). This has led to the observation that ignorability is evoked – without justification – on the basis of convenience rather than principled causal assumptions. Joffe et al write: "Most attempts at causal inference in observational studies are based on asumptions that treatment assignment is ignorable (Rosenbaum and Rubin 1983); ignorability involves conditional independence of treatment and potential outcomes. Such assumptions are usually made *casually,* largely because they justify the use of available statistical methods and not because they are truly believed" (Joffe, Yang, and Feldman 2010, 1 emphasis added to highlight that the authors use the word CASually and not CAUsally).

[94] Despite recognizing the importance of Pearl's contributions to causality, Rubin and his many adherents maintain that they see little value in replacing the Potential Outcomes framework with Pearl's approach to causal inference. Despite the fact that both Pearl's and Rubin's respective approaches to causality have been proven to be mathematically identical (Pearl 2009b, 244–45), disagreements between the two camps continue to flare up to this day, mainly in the context of blog posts (and replies thereto) soaked through with thinly-veiled acrimony (see: Gelman 2019). As far as I have been able to determine, the primary disagreement between the two seems to boil down to whether or not one subscribes to the mantra of 'no causation without manipulation,' which –in this context – is taken to mean 'a value that a conscious human being could plausibly influence in an experimental setting.' Those in the Potential Outcomes camp generally adhere to this mantra, whereas Pearl – alongside almost everyone who follows the 'graphical' paradigm – believe that non-manipulable variables can still be productively spoken about and modelled as causes. The practical effect of this disagreement is that Rubin's camp balks at the inclusion of unobserved confounders in graphical models, as doing so ascribes causality to them (and this is incompatible with the 'no causation without manipulation' credo). Consequently, Potential Outcomes purists generally oppose the inclusion of causal graphs in their work (Rubin 2004a; Pearl 2009b; Pearl and Mackenzie 2018).

[95] This is a significant point: divorcing causal models from functional or parametric assumptions permits the use of a much wider array of estimation techniques. In light of these developments, some have called for a greater emphasis on using machine learning and/or neural networks in place of – or as an augment to – more traditional general linear models (Schölkopf 2022).

[96] Pearl is credited with developing the first general approach to non-parametric SEM, which emerges as a logical consequence of the axioms powering his theory of inferred causation (Bollen and Pearl 2013).

and atomic interventions is perfectly compatible with Cartwright's approach.[97] I argue that the same holds for Salmon (1998) and Dowe's (1992) 'Process Theory,' which views causality as a chain of physical causes-and-effects, drawing heavily upon conserved quantity theory: the language of graphs is well-suited to tracing said chains.[98] By way of a final example: 'Agency Theory' (see: Von Wright 1968; Menzies and Price 1993) interfaces well with Pearl's use of counterfactuals in making sense of causal claims, though the two camps would disagree over the degree of anthropomorphism demanded by said counterfactuals. In sum, I believe that the language of graphs can accommodate a wide array of causal commitments, and can serve as a useful common lexicon for specifying and debating inter-paradigmatic causal problems.

## Conclusion

In this chapter, I argued that every statistical estimate of empirical effect destined for publication in a sociological journal should be accompanied by a transparent causal model. I provided a brief account of the development of the two major causal inference frameworks in the social sciences – graphical approaches and the potential outcomes framework – and proposed Judea Pearl's causal inference framework as the best candidate for widespread adoption in Sociology, and thus the framework best poised to assist sociologists in meeting the requirements laid out in this chapter.

In the subsequent chapter, I describe the results of my search through the works published in top sociological journals' 2022 volumes for evidence that sociologists consistently express their causal assumptions using a viable causal language. It should come as no surprise that the results therefrom are dismaying.

[97] Cartwright disagrees (2004), but I find the response to her objections in (Pearl 2009b) to be more than sufficient to demonstrate the equivalence between the two positions.

[98] Though the utility of doing so in an exacting manner remains, in my view, dubious.

# Chapter 3 – A Review of Causal Methods in Contemporary Sociology

Pierson Browne

# Introduction

In the previous chapter, I argued for the indispensability of causal inference in inferential quantitative sociology, provided an account of the development of two major schools of causal inference, and introduced Judea Pearl's structural causal models, which synthesize elements of the graphical and potential outcome approaches. This chapter builds directly on that work by seeking to determine the extent to which transparent causal models (regardless of the specific approach) are used in published inferential quantitative work. The impetus for this chapter emerged from a fruitless search for any kind of systematic field survey or review of causal methods in sociology: to the best of my knowledge, no such survey/review yet exists. This chapter contributes to sociological knowledge by carrying out a review of the causal methods employed be the articles published in 'top' sociological journals in the year 2022.[99]

The core motivation for this review can be found in the following passage from Judea Pearl's *Causality* (2009b, 334):

> Any causal premise that is cast in standard probability expressions, void of graphs, counterfactual subscripts, or $do(\cdot)$ operators, can be safely discarded as inadequate. Consequently, any article describing an empirical investigation that does not commence with expressions involving graphs, counterfactual subscripts, or $do(\cdot)$ can be safely proclaimed as inadequately written.

This chapter has three related objectives. The first is to determine the extent to which research articles published in top sociological journals include *explicitly causal* statistical estimations of empirical effect. This query is based upon Pearl's claim that all empirical investigations must be based on one or more causal premises (2009b, 334).[100]

The second stems from Pearl's contention that causal claims and assumptions must be supported by purpose-built causal notation, consisting of counterfactual notation, graphs, or the $do(\cdot)$ operator (Pearl 2009b, 334). To what extent is this the case? are causal assumptions transparently communicated in articles published in 'top' journals in the field?

The third query seeks to determine the health of Structural Equation Modelling as an approach to causal inference. As described in the previous chapter's account of the development of the 'graphical' paradigm, SEM was developed by sociologists as an extension to Wright's (1921) original method of path analysis – starting in the late 1980s, a bevy of critiques (see: Karlin 1987; Freedman 1987; Cliff 1987; Muthen 1987) caused SEM practitioners to largely abandon two of the method's distinctive features: diagrammatic portrayal of structural equations (Morgan and Winship 2015, 87), and an inborn assumption of causal relevance (Pearl 2009b, 135). In light of Pearl's development of non-parametric SEM and rehabilitation of the path diagram, has sociological SEM reclaimed its original purposes?[101]

The chapter proceeds by:

- Outlining the rationale supporting the chosen scope of the review.
- Describing the procedure used to select 'top' sociological journals.
- Detailing the 6 steps of the review process.
- Using the results from the review to answer the three queries detailed above.
- Discussing the review's limitations.

The chapter concludes by ruminating on the potential benefits of sociology's emergence as a causally rigorous field. It also describes recent efforts to lower barriers to the adoption of causal inference using purpose-built software packages for R and Python.

# Review Scope

In determining the scope of the review, I sought to strike an effective balance between breadth and depth. The breadth of the review had to be sufficiently wide to permit claims about trends in 'top' sociological journals. A smaller corpus of

[99] Throughout this chapter, the term " 'top' sociological journal" is frequently evoked. The scare-quotes around 'top' are consistent and intentional, in recognition of the fact there's no rigorous way of identifying which journals are – in fact – the 'top' journals. I discuss my solution to the problem in the 'journal selection' section below.

[100] The basis for this query can be found in the block quotation contained in this chapter's introduction (2009b, 334), but the order is different than that used here. The original Pearl quote makes three arguments in the following order: 1. all causal premises must use appropriate notation; 2. empirical estimates of effect must be based on one or more causal premises; 3. therefore, any article containing an empirical estimate of effect must use appropriate notation. The first query I describe here is based on the second of Pearl's three claims, whereas the second query is based on the lattermost of Pearl's three claims.

[101] In this case, I will consider both Structural Equation Models *and* Simultaneous Equation Models.

articles – say, the contents of one or two journals – would not have been sufficient to make the claims I wished to make. Using a smaller collection of journals also ran the risk of including too many journals whose contents were not sufficiently pertinent to this review's stated objectives.[102]

This review was not wholly automatable: in order to permit the kinds of analysis I wished to conduct, I needed to be able to spend time performing in-depth readings of the relevant sections in each article. Some aspects of the review – such as searching for authors' names – could be conducted at scale. The core of the review, however, depended on being able to exactly determine the nature of the causal claims made by the authors, their causal assumptions (where transparently communicated), and the techniques used to justify said causal claims (where applicable). In this regard, there was no replacement for my ability to spend time reading each article included in the review.

To satisfy both competing demands described above, I elected for a corpus of 14 journals maximum: up to 10 drawn from of the 'top' sociological journals overall, and two each from lists of the top British and Canadian sociological journals. While I approached this review task systematically, I do not claim that what follows constitutes a 'systemic review' of the field in a formal sense.[103] My contribution is necessarily limited to a small subset of the available journals and only one year's worth of publications therefrom.

## Journal Selection

In selecting journals to review, I used Jerry A. Jacobs' "Journal Rankings in Sociology: Using the H-Index with Google scholar" (Jacobs 2016) as a starting point.[104] To account for ranking changes since the publication of his article, I combined Jacob's list of 'top' sociological journals by H-index with Google Scholar's 2022 H-index rankings. The top 5 journals were taken from each of the two lists.[105] A journal was 'skipped' if its 2022 volume contained over 150 articles, in which case it was replaced by the subsequent journal on the respective list.[106] Two journals were skipped in this way:

- *Social Indicators Research*, which was 5th on Jacobs' list; replaced by *American Journal of Sociology*, which was 6th.
- *Journal of Ethnic and Migration Studies*, which was 2nd on Google Scholar's 2023 list; replaced by *Sociology*, which was 6th.

The resulting journal lists were as follows:

| Jacobs (2016) | Google H-index 2023 |
|---|---|
| American Sociological Review | American Sociological Review |
| Criminology | Journal of Marriage and Family |
| Demography | Demography |
| Journal of Marriage and the Family | Annual Review of Sociology |
| American Journal of Sociology | Sociology |

To ensure that results were not exclusively representative of American sociological practice, the top two British and Canadian journals (from Jacobs' list[107]) not otherwise included in the list were selected:[108]

[102] Although this only became clear in retrospect, some of the journals included – such as the journal titled *Sociology* – were overwhelmingly qualitative. The *Annual Review of Sociology*, by way of another example, contained many articles that were *about* quantitative sociology, but contained no articles with quantitative estimates of effect – as such, none of the articles therefrom were included in the review.

[103] i.e. following explicit knowledge synthesis methodologies commonly practiced in, say, the health sciences

[104] Google calls their journal ranking index the 'h5-index,' which indicates that it covers the 5 most recent years. To avoid confusion, I'll employ Jacobs' notation ('H-Index') throughout. Despite the H-index's well-publicized shortcomings (Waltman and Van Eck 2012), it was nevertheless suitable for Jacobs' purposes: so shall it be for mine.

[105] This was done with no regard for duplication between the two: If a journal appeared on both lists, it was included without extending the list beyond the top 5.

[106] This was done for practical purposes. For example: *Social Indicators Research* alone published 312 articles in the year 2022, and I simply did not have the time to review each of them. I also think I'd rather take the hemlock.

[107] Jacobs identifies *Innovation* as the top Canadian journal. Given that the journal is no longer headquartered in Canada, nor does it prioritize Canadian content, it was skipped.

[108] The journal *Sociology*, which appears in the 6th position of Google Scholar's 2023 rankings, is headquartered in the United Kingdom. Since it appears in the original slate of journals selected for inclusion here, I did not count it against the additional UK journals added here.

| UK Inclusions | Canada Inclusions |
|---|---|
| Information, Communication and Society | Canadian Journal of Sociology |
| European Sociological Review | Canadian Review of Sociology and Anthropology |

The Canadian Review of Sociology was on hiatus during the year 2022. Instead of removing it from the review, I elected to use the articles it published in the year 2021 instead. The final list of journals,[109] as well as the count of 2022 articles published by each each, is as follows:

| Journal | Number of 2022 Articles |
|---|---|
| Information, Communication and Society | 132 |
| Demography | 87 |
| Journal of Marriage and Family | 68 |
| Sociology | 64 |
| European Sociological Review | 55 |
| Canadian Review of Sociology and Anthropology | 40 |
| American Sociological Review | 36 |
| American Journal of Sociology | 35 |
| Criminology | 24 |
| Annual Review of Sociology | 24 |
| Canadian Journal of Sociology (2021) | 9 |

Altogether, the selected journals contained 574 articles.

## Review Process

My approach to this review was primarily informed by two sociological publications that contain reviews of sociological literature: Lynch and Bartlett (2019), and Lundberg et. al. (Lundberg, Johnson, and Stewart 2021). Lynch and Bartlett's review (2019) assessed the prevalence of Bayesian statistics in sociology by conducting a Web of Science search for the term 'Bayesian' in articles published in 'top' sociological journals[110] between the years 2008 and 2017, inclusive. They found that less than 1% of the articles included in their search utilized Bayesian statistics.

Lundberg et. al. (Lundberg, Johnson, and Stewart 2021) conducted an in-depth analysis of the 32 quantitative articles published by *American Sociological Review* in 2018. Each of the 32 articles was read by two of the three researchers who would discuss their separate findings and come to a consensus about each article on a case-by-case basis. They found that they could be certain of an article's theoretical estimand[111] and target population in exactly 0 of the articles so reviewed. They also observe that while roughly half of the articles report causal claims, none were sufficiently detailed for Lundberg et. al. to be "entirely confident of the intended intervention" (Lundberg, Johnson, and Stewart 2021, 552).

My approach might best be described as the median between Lynch and Bartlett's multi-year field survey and Lundberg et. al.'s in-depth single-journal review. My review covers the articles published in 11 sociological periodicals in a single year (a larger scope than Lundberg, Johnson, and Stewart (2021), but much smaller than Lynch and Bartlett (2019)) and conducts a review of each article that contains at least one statistical estimate of empirical effect size (which is a more in-depth than Lynch and Bartlett (2019)'s keyword-only approach, but not as in-depth as the detailed multi-reviewer approach used by Lundberg, Johnson, and Stewart (2021)).

Each article in the corpus was reviewed using the following steps:

[109] Note the lack of citations. For reasons I discuss below, the vast majority of the articles I consulted in the course of this review will not be cited anywhere in this dissertation. Since I have mentioned the periodicals consulted – as well as the dates of the volumes so consulted – the APA style guide insists that it is improper to include a formal citation or a bibliography entry for them (Lee 2012).

[110] The journals included in their search are: American Sociological Review, Sociological Methodology, Social Psychology Quarterly, Sociological Theory, Journal of Health and Social Behaviour, Journal of Education, Demography, Social Methods and Research, American Journal of Sociology, and Social Forces (Lynch and Bartlett 2019).

[111] Consisting of a unit-specific quantity of interest and a target population (Lundberg, Johnson, and Stewart 2021).

**Step 1:**

The article's abstract, introduction, and each of its figures and tables were consulted to determine whether it contained one or more statistical effect estimations based on empirical data.

- If no such effect estimations were found, the article was excluded from the analysis.[112]
- If one or more effect estimations was found, the article was labelled as either 'Frequentist' or 'Bayesian', depending on the statistical paradigm employed.

**Step 2:**

The paragraph surrounding any occurrence of the "causal," "causality", "causative", or "causation" was read to determine what causal assumptions the authors were making, as well as how these causal assumptions informed the remainder of the analysis. On the basis of the causal claims and assumptions found therein, each article was classified into one of four categories:

- **Explicitly Non-Causal**:
  - Articles that contained one or more explicit disclaimers that reported effect sizes could not be interpreted as causal.
  - Articles that called for further research in the area to determine whether or not the reported effect is causal.
- **Ambiguous**:
  - Articles where no discussion of causality was present.
  - Articles where discussion of causality was limited to the literature review section, discussing other authors' causal efforts/claims.
  - Articles whose discussions of causality were few, and limited to highlighting causal ambiguity in the reported effect (usually to profess that the direction of the causal effect was unknown).[113]
- **Limited**:
  - Articles that highlighted barriers to inferring causality from the reported correlations **AND** did not take steps to ameliorate said barriers **AND** refrained from explicitly ruling out causal interpretations.
- **Explicitly Causal**:
  - Articles that explicitly estimated one or more causal effects.

**Step 3:**

The full text of the article was searched for instances of terms that could indicate the presence of a viable causal strategy.[114] If three or more such instances were found in a single article, each instance in that article was examined to determine if the strategy informed the article's methodology.[115]

- If two or more such sets of terms were found in a given article, the article was categorized according to the author's characterization of their study's causal framework.
- If two or more potentially causal strategies were identified, then the most general term was used (causal frameworks, such as Potential Outcomes, took precedence over study design/experimental design/bias reduction techniques).

**Step 4:**

The full text of the article was searched for instances of the names 'Rubin' and 'Pearl' (not case-sensitive). Any matches were manually examined to see if the article cited any of the causality-related works published by either Donald B. Rubin or Judea Pearl.

[112] Most of the articles excluded in this way might best be classified as 'qualitative.' Many, however, presented descriptive statistics, focused on methods, or were reviews.

[113] Articles in this category may discuss coefficients using causally suggestive language without explicitly denoting them as such. I have elected not to draw a distinction between articles that present correlations/coefficients as 'effects' and those that merely present them as correlations/associations.

[114] These terms included reasonable permutations of each of the following: experiment, instrumental variable, potential outcomes, propensity score, discontinuity, structural equation, ignorability, front-door criterion, back-door criterion, counterfactual, matching regression, switching regression, panel data, and difference-in-difference.

[115] Frequently, they did not: in many cases, especially among articles with lower term counts, the terms either appeared in the article's references, or as part of the literature review describing what other authors had done.

**Step 5:**

The article's tables, figures, and equations were examined to determine whether the article unambiguously employed Potential Outcomes subscript notation, the $do(\cdot)$ operator, or a structural causal model.

**Step 6:**

Finally, on the weight of the evidence collected during the steps above, I categorized each article as either being 'causally adequate' or 'causally inadequate'. Articles deemed 'causally adequate' shared the following qualities:

- Clearly-stated, explicitly causal assumptions rendered using a Structural Causal Model, counterfactual Potential Outcomes notation, or the *do*-operator.
- Explicit recognition of the threats to identification implied by said causal assumptions.

Throughout this review, I take no position on the viability or the appropriateness of the causal assumptions made in any article, as to do so would require domain-specific knowledge that I lack.

Causality remains a contentious and hotly-debated topic. In light of this lack of consensus, it is my position that authors should not shoulder the blame for gaps in their understanding of causal inference. This, in turn, suggests that authors should not be 'singled out' for failing to employ causal practices that they were – in all likelihood – neither sufficiently aware of nor trained in the use of.[116] As such, I have elected to avoid any direct citation or mention of articles whose causal assumptions I could not fully identify.[117] I will instead include direct citations to individual articles whose methods are exemplary[118] – information about all other articles will be aggregated.[119]

# Results

Of the full corpus of 574 articles gathered, 283 included some statistical estimation of effect[120] – the remainder were not reviewed.

## Query #1: Explicit Causality

To answer Query #1, we must simply examine the results to see if explicit causal claims are made in all – or the vast majority of – the inferential quantitative sociological articles published in 2022.[121]

Among the articles reviewed, the most prevalent approach to causality was to simply neglect to mention it, or to include one or two sentences casting doubt on the causal order of the variables. I identified 139 articles as adopting this approach – nearly one-half of the 283 articles reviewed.

The second-most common approach to causality – found in 74 of the articles – was to include some kind of explicit disclaimer that the study's results should not be interpreted causally, and should instead be interpreted as raw

[116] My backing on this point is sparse. I nevertheless cite Bartram (2021) – and specifically the note on page 726 – that follows from Bartram pointing out the inadequacy of specific articles using a 'standard set' of control variables without reference to a fully-specified causal model. He writes: "It is perhaps unfair to single out these authors in this context; others adopt the same approach but use different terms to describe it (I myself adopted it in earlier work: Bartram 2011). This mode of analysis is far from uncommon" (Bartram 2021). Here, I take the position that it *is* unfair to single out authors, especially given how widespread the problem is.

[117] My decision to do so runs counter to the approach Lundberg et. al. (2021) take in their article. Lundberg et. al. provide in-depth accounts of the methodological shortcomings present in the articles they reviewed. On page 552, for example, they include a detailed discussion of how specific articles they reviewed elected to condition their effect estimates on post-treatment variables, which severely undermines the viability of their estimate (Lundberg, Johnson, and Stewart 2021, 552) – throughout the section, the authors of said articles are mentioned by name.

[118] Or, more specifically: exemplary in light of Pearl's aforementioned criteria (Pearl 2009b).

[119] Obviously, the purpose of doing so is not to ensure that all of the authors whose works I reviewed remain *anonymous* per se, as records of their contributions are publicly available – anyone with sufficient motivation could easily reverse-engineer the entire list of authors whose works are reviewed in this chapter. My purpose in approaching the issue this way is to ensure that no author(s) are 'singled out' for systemic issues in academic publication and review. Since I do not have room to mention each author by name in the context of this chapter, selecting a subset thereof would unfairly direct negative attention towards their work, whilst arbitrarily sparing others the same fate. The same applies on a macroscopic scale: even if I had the room to mention – by name – each author whose work I reviewed, doing so would constitute an unfair singling-out of the authors who published their work in one of the 11 journals I selected in the year 2022; those who published in other journals or in other years would be semi-arbitrarily spared.

[120] 279 were Frequentist, 4 were Bayesian – this suggests that the field of Sociology hasn't improved much since Lynch and Bartlett (2019) published their findings on the topic in 2018.

[121] Detailed results pertaining to the causal strategies employed and journal-by-journal tabulations can be found in Appendix A.

correlations or associations. Although slight variations in the language used to explain these disclaimers made it unfeasible to rigorously categorize and tabulate the reasons given, I can confidently claim that most such disclaimers were made in reference to the study's lack of access to longitudinal data, experimental data, or both. These justifications are deeply concerning, as both are based on myth. While it might be accurate to state that causal claims *can,* in some circumstances, be more convincing when made using longitudinal data, there is no fundamental temporal requirement (Wunsch, Russo, and Mouchart 2010).[122] The same is true of experimental data: although experiments are more likely to make the treatment assignment 'ignorable' – in Rubin's (1974) words – they do not guarantee an unconfounded effect. Conversely: observational data can still achieve conditional ignorability.[123] In other words: experimental data is neither a necessary nor a sufficient condition for causal inference – the same is true of longitudinal data.

I identified 27 articles as containing 'limited' causal claims. The remaining 43 articles explicitly claimed that their estimates could be interpreted as causal.

A generous interpretation of the data suggests that only 70 of the 283 articles reviewed – about one quarter – contained one or more causal claims. Since 27 of these articles avoided making explicit causal claims, a more strict interpretation of the data reduces the proportion of explicitly causal articles to about one seventh of the total. Regardless of which figure one uses, one can safely conclude that top sociological journals do not yet insist that authors include explicitly causal interpretations of statistical estimations of empirical effects as a precondition to publication.[124]

## Query #2: Transparent Causal Models

Query #2 concerns a subset of the 43 explicitly causal articles identified above, assessing them in light of Pearl's contention that "any article describing an empirical investigation that does not commence with expressions involving graphs, counterfactual subscripts, or $do(\cdot)$ can be safely proclaimed as inadequately written" (2009b, 334). Of all the articles included in this review, only five were deemed 'causally adequate'.[125] In what follows, I briefly describe each:

Lanfear's "Collective efficacy and the built environment" (2022), published in *Criminology*, seeks to examine "associations 1) between past collective efficacy and present criminogenic features of the built environment, as well as 2) between those built environmental features and crime, net of present collective efficacy" (Lanfear 2022, 370). To accomplish this, Lanfear depicts the article's causal assumptions using graphical causal models (see example in Figure 24), considers *d*-separation and conditional/sequential ignorability in relation to the causal models, and uses SEM to estimate the effects of interest.

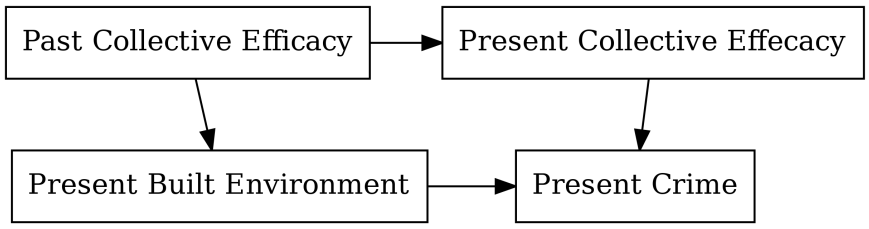


Figure 24: Causal model adapted from Lanfear (2022)

Using a natural experimental design, Aksoy and Gambetta (2022) seek to provide a causal estimate of the theorized effect of religiously-motivated 'sacrifice' (in the form of Ramadan fasting) on religiously-motivated voting patterns. Their article posits a causal model (see Figure 25) in the form of a Structural Equation Model, considers threats to identification, and concludes that all such threats are accounted for. In their causal model, the 'Fasting Length' variable is determined by exogenous disturbances from the interaction between the Ramadan period and day length (which is caused by latitude);

[122] Wunsch et al write: "the argument according to which longitudinal studies are the only way to make causal inferences because they take temporal order of the cause-effect relation into account is ill-founded" (2010, 15).

[123] I have elected to use Rubin's criteria for unconfounded estimation here, as his ideas are more prevalent than Pearl's in the articles I reviewed.

[124] This conclusion should not be interpreted as a claim that **no** journal editors/reviewers in sociology insist on explicit causality. There may be individual editors/reviewers who do insist on explicit causality – it is possible that they are involved with journals that were not included in this review's corpus, or that some of them were nominated as reviewers for articles that this review identified as being explicitly causal. As such, the claim I make here should only be interpreted as a claim about the lack of widespread norms demanding explicit causality in inferential quantitative sociology.

[125] Throughout the entire corpus, only 20 articles cited Donald B. Rubin directly. Only a single article cited Judea Pearl. The sole article that mentioned Pearl cited his and Dana Mackenzie's 'Book of Why' (2018). The article that did so is Kratz, Pettinger, and Grätz (2022) – which I am willing to identify here because theirs is one of the articles that I judged to be 'causally adequate' in view of Pearl's criterion. See below.

this, they claim, permits the use of 'Fasting Length' as an instrumental variable for estimating the influence of 'Religiousity' on 'Islamic Vote.'[126]

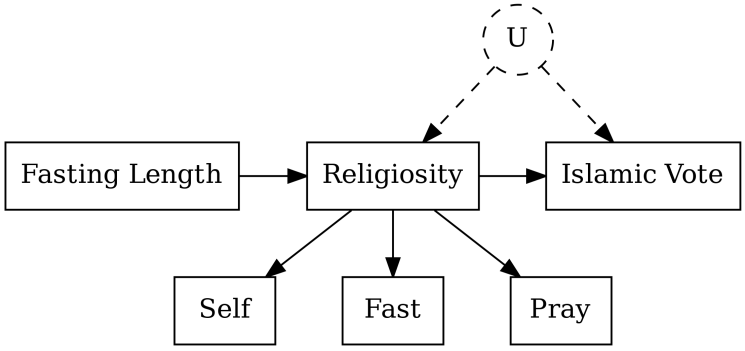


Figure 25: Causal model adapted from Aksoy and Gambetta (2022)

Kratz et. al.'s (2022) article seeks to estimate the effect of social origin on social destination: one model uses direct effects, the other uses indirect effects (via educational attainment). They do this in light of a structural causal model (see Figure 26), and a series of counterfactual statements, including explicit specification of Average Treatment Effects, Controlled Direct Effects, Natural Direct Effects, and Natural Indirect Effects. Their article cites both Rubin and Pearl by name, and liberally employs elements drawn from both the Potential Outcomes framework *and* the Graphical paradigm.

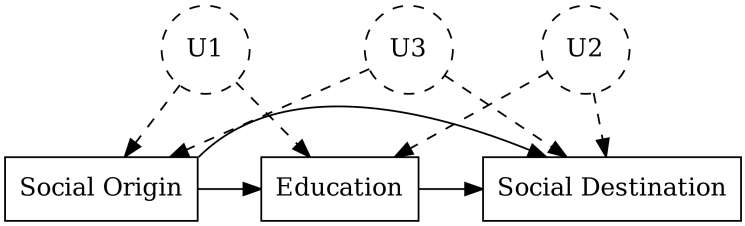


Figure 26: Causal model adapted from Kratz et. al. (2022)

Wodtke et. al. (2022) estimate the effect that growing up in a disadvantaged neighbourhood has on children's vocabulary skills – a causal effect they argue is moderates by exposure to lead contamination. Among the five articles selected for consideration here, Wodtke et. al.'s is noteworthy for containing – by far – the most detailed and explicit consideration of mathematical causality, as well as being the only article in the entire corpus of 574 articles to directly cite any of Judea Pearl's works (Wodtke, Ramaj, and Schachner 2022).[127] They provide a structural causal model (see Figure 27), select de-confounding covariates in light of the causal model's implied conditional independencies, and include detailed consideration of Average Treatment Effects, Average Joint Effects, and Controlled Directed Effects (the lattermost of which isolates direct causes, separate from mediating variables). They express the validity of their causal claim using Rubin's language of ignorability.

The article by Legewie and Cricco (2022) is unique among the articles in this section in that they do not include any graphical depiction of their causal assumptions. They explicitly recognize that their estimate relies on the sequential ignorability assumption, consider threats to identification, and postulate that their use of 'marginal structural models' – an explicitly causal method – is sufficient to remove confounding effects on their estimate.[128] While their focal equation on page 1747 does not appear to explicitly espouse Potential Outcomes notational standards, it uses subscripts to denote counterfactual cases. This, combined with the explicit causal content of their article, suggests that they have selected a causally-appropriate estimator. They use this estimator to assess the effect of neighbourhood policing on graduation rates, with a focus on the different environments typically experienced by racialized high school students.

[126] They write: "the SEM framework presents a clear diagram of the causal pathways. Religiosity fully moderates the effect of fasting daylength on votes for Islamist parties" (Aksoy and Gambetta 2022, 576)

[127] The work in question is Pearl and MacKenzie's *The Book of Why* (2018).

[128] They write: "This procedure properly adjusts for biases arising from post-treatment confounding without overcontrolling for the effect of the treatment or inducing collider stratification bias" (Legewie and Cricco 2022, 1747).

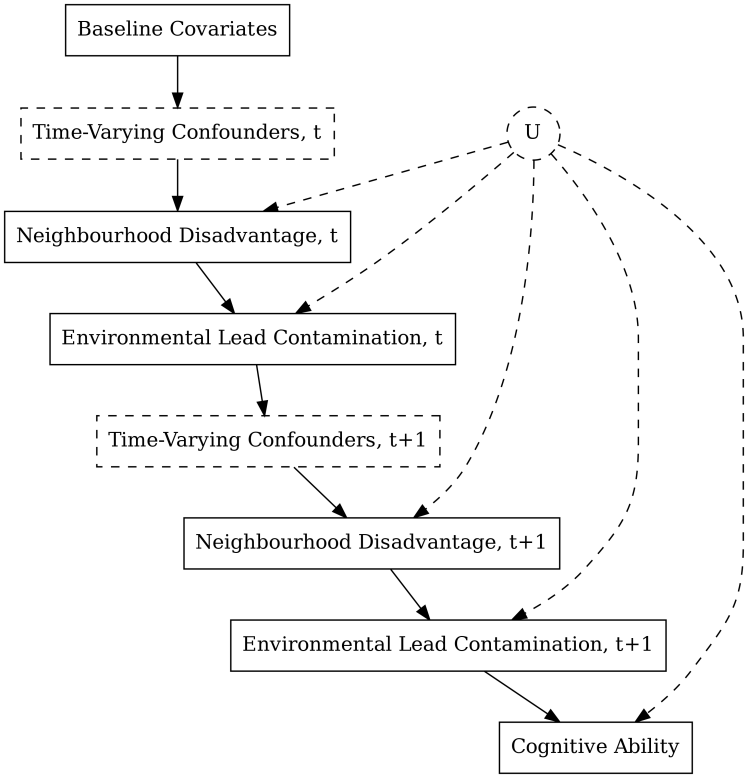


Figure 27: Causal model adapted from Wodtke et. al. (2022)

Despite some promising signs from a handful of authors publishing in sociological journals, the evidence provides a fairly obvious answer to query #2. Do 'top' sociological journals insist that causal assumptions be transparently communicated as a precondition to publication? No, they do not.[129]

## Query #3: Causal, Transparent SEM

Query #3 contains two components. The first pertains to the following quote from Morgan and Winship (2015):

> We see two related outcomes of the rise and then demise of traditional linear path models conceived and estimated in this fashion in the social sciences. First, when it became clear that there was no agreed upon way to represent within path diagrams the interactions that could be specified in regression equations to capture variability and context, path diagrams came to seem much less useful. Researchers interested in such variability and context may have continued to draw path diagrams on yellow pads in their offices, but rarely did their drawings turn up in published articles. Estimation and reporting became a word and number affair, often with too much of each. Second, and far more troubling, many scholars apparently chose to retain a linear additive orientation, even while no longer using path diagrams. (Morgan and Winship 2015, 86-7)

The above quote begs the question: since the publication of Morgan and Winship's book in 2015, have path diagrams re-emerged as a common feature in Sociological uses of SEM?

The second component of this query stems from the claims Pearl (2009b) makes in the following passage:

> Structural equation models are used by many, but their causal interpretation is generally questioned or avoided, even by their leading practitioners. [...] The current dominating philosophy treats SEM as just a convenient way to encode density functions (in economics) or covariance information (in social science). (Pearl 2009, 135)

[129] This claim is subject to similar caveats as the concluding finding in the subsection on query #1: for similar reasons as before, I am not claiming that there aren't **any** editors/reviewers who insist on causal transparency, only that doing so is not currently a standard practice.

Pearl leads one to ask: in the decade since the second edition of his book was published, do practitioners of Sociological SEM generally view their work as having a causal interpretation?

Overall, I identified 21 articles that employed SEM as an estimation technique in at least some portion of the article.[130] Of these 21 articles, 14 included visual depictions of their models, including path coefficients. 4 included diagrams that were strictly conceptual. The remaining 3 included no graphical representation of their model – conceptual or otherwise. Although the majority of the SEM analyses reviewed here fall short of including a diagram that Pearl would identify as constituting a Structural Causal Model, it no longer seems accurate to portray sociological SEM as being solely a "word and number affair" (Morgan and Winship 2015, 87). The results from 2022 would suggest that at least some Sociological SEM practitioners are comfortable employing path diagrams, and that they frequently do so.

Only 3 of the 21 SEM articles explicitly identified their estimates as causal. The majority – 12 in all – contained some discussion of causality, but did so with the inclusion of a caveat precluding their findings from being interpreted as causal. The reasons given include:

- Conditioning on a collider (1)
- Selection bias (1)
- Theory causal, findings not causal (1)
- Results ambiguous and tentative (1)
- Nonexperimental data (1)
- Possibility of reverse causation (2)
- Lack of longitudinal data (5)

The remaining 6 contained no mention of causality whatsoever. Here again, we see a relatively prominent group (7 in total) who view experimental and/or longitudinal data as *sine qua non* for causality. For the same reasons given above (in the subsection on query #1), this is a disturbing trend, as it is neither accurate nor practical (Rubin 1974; Wunsch, Russo, and Mouchart 2010).

Taken as a whole, the results from 2022 would suggest that SEM has not yet recovered from its data-driven hangover. Practitioners of Sociological SEM appear to care about causality, but most appear to have been deluded into thinking that causal claims can only be made with the benefit of longitudinal data, experimental conditions, or some other combination of factors they don't typically enjoy access to.

## Limitations

This review has several limitations that are important to consider. First, and most obviously, the review only considers a small subset of the contemporary sociological journals available, and does so for only one year's worth of publications. It is entirely possible that the results would have led to a broadly different impression had I considered journals from topic- or method-specific subfields of sociology, student journals, or publications from a different year.

This review was also limited by the fact that I conducted the review alone. Unlike the in-depth review described in Lundberg et. al. (Lundberg, Johnson, and Stewart 2021) – where each article was read by two readers – I am able to offer no similar guarantee of internal validity. While I contend that any other scholar with similar stated objectives would draw similar conclusions from a review of the same body of material, there would surely be some (likely minor) differences.

Finally, this review was limited by my lack of domain knowledge in the various subfields to which each article contributes; this may have hampered my ability to recognize causal assumptions if they were rendered using domain-specific notation or references. As such, there may be cases where I erroneously judged an article to be lacking an explicit causal model, when in fact a causal model was described textually or using non-standard notation. The article by Legewie and Cricco (Legewie and Cricco 2022) is an example of one such article that lacked a graphical causal model and employed nonstandard notation, but which I nevertheless assessed as 'casually adequate.' The corpus likely contains several similar articles where field-specific terminology, assumptions, references, and shorthand combined to communicate an adequate causal model to readers who are well-versed in said field-specific markers/methods, but which I – owing to my relative unfamiliarity – was unable to appreciate. This limitation does not, however, damage this chapter's position on the indispensability of *transparent* structural causal models. All works of published inferential quantitative sociology should

[130] Note: due to the tiered categorization strategy described in the 'Review Process' subsection above, the number of articles using SEM (21) differs from the number of articles classified as using SEM as their primary causal strategy (17).

include a causal model whose assumptions are transparent and therefore available for scrutiny by any sociologist – regardless of their familiarity with the field.

I do not believe that the limitations of this review undermine the conclusions I draw from it. As stated above, the objective of this study was to determine the extent to which articles published in 'top' sociological journals explicitly reference causality in their statistical estimates of empirical effects (query #1), and/or are accompanied by a transparent causal model (query #2). The limitations of the study do not prevent me from confidently concluding that – as of the year 2022 – neither causality nor transparent causal models are yet considered an essential component of inferential quantitative sociology.

# Conclusion

In the previous chapter, I argued that all estimates of effect size in sociology require the use of a causal model, and that said causal models should be as transparent as possible. The results of the review conducted in this chapter show that, as of the year 2022, sociological journals do not yet insist that the inferential quantitative articles published therein include explicit, transparent causal models. This is a dire finding: the inferences drawn by a causally-inadequate article should be viewed with suspicion, which in turn casts doubt on the majority of the inferences published in 'top' sociological journals in the year 2022.

It is entirely plausible that reading through this dissertation's two chapters on causal inference was – for some – a distressing experience. It is difficult to swallow the notion that the vast majority of the inferential quantitative sociological articles published in 2022 were 'inadequate' (in the words of Pearl (Pearl 2009b, 334)). I wish to stress, however, that most of the articles I reviewed were not that far distant from achieving causal adequacy: many authors clearly devoted careful thought and consideration to theory, literature, data collection, and analysis – incorporating causal inference into this well-established chain of knowledge production will incur few costs to those who do so. The inculcation of principled causality will also reap dividends far greater than the effort required for most sociologists to adopt it as standard practice: by making the process of operationalizing theory and domain knowledge to derive a measurable quantity explicit, causal inference formalizes and makes transparent the process of synthesizing knowledge and transferring it from extant findings to new research projects. Aside from the obvious benefits therefrom, I'd also like to emphasize that the potential perils of positing an 'incorrect' causal model – by, perhaps, producing a SCM whose testable assumptions are later shown to be violated by data – are comparatively minor: a cycle of model postulation and subsequent refinement and/or refutation is a healthy reification of the scientific method. For the most part, the kinds of thought and theorizing needed for building and refining causal models are similar to that which sociologists already habitually employ in their work: causal inference merely represents a codification of this process. It is for this reason that I urge readers to think about the inadequacy of causal methodology in sociology not as an indictment of the field, but as an opportunity.

Distressed readers should also take heart in the knowledge that graduate-level mathematical proficiency is not a requirement to fully implement a structural causal model in one's research workflow. Aside from the conceptual shortcuts contained in Pearl's work, enterprising individuals have begun building a constellation of software packages designed to facilitate the specification and exploration of causal models. The excellent `dagitty` suite, for example, has been developed into both a package for the `R` statistical programming language, as well as a standalone browser-based graphical user interface (Textor et al. 2016). Using `dagitty`, researchers can build structural causal models using a point-and-click system which allows for smooth parsing and simple revisions of the causal graph. Once the structural causal model has been specified, `dagitty` automatically determines a suite of minimally-sufficient adjustment sets compatible with most standard approaches to statistical estimation.[131]

Another such tool is the `dowhy` Python package (Sharma and Kiciman 2020), which is capable of automating causal estimation and applying a series of refutation tests to the estimate. While `dowhy` sacrifices some of the ease-of-use and the user-friendly interface of suites like `dagitty`, it encodes a powerful and mature software-driven approach to ensuring causal rigour.

As tools similar to `dagitty` and `dowhy` emerge – and extant packages continue to mature and accrue greater functionality – barriers to the adoption of causal inference will monotonically crumble.

[131] Including, if justified, linear regression.

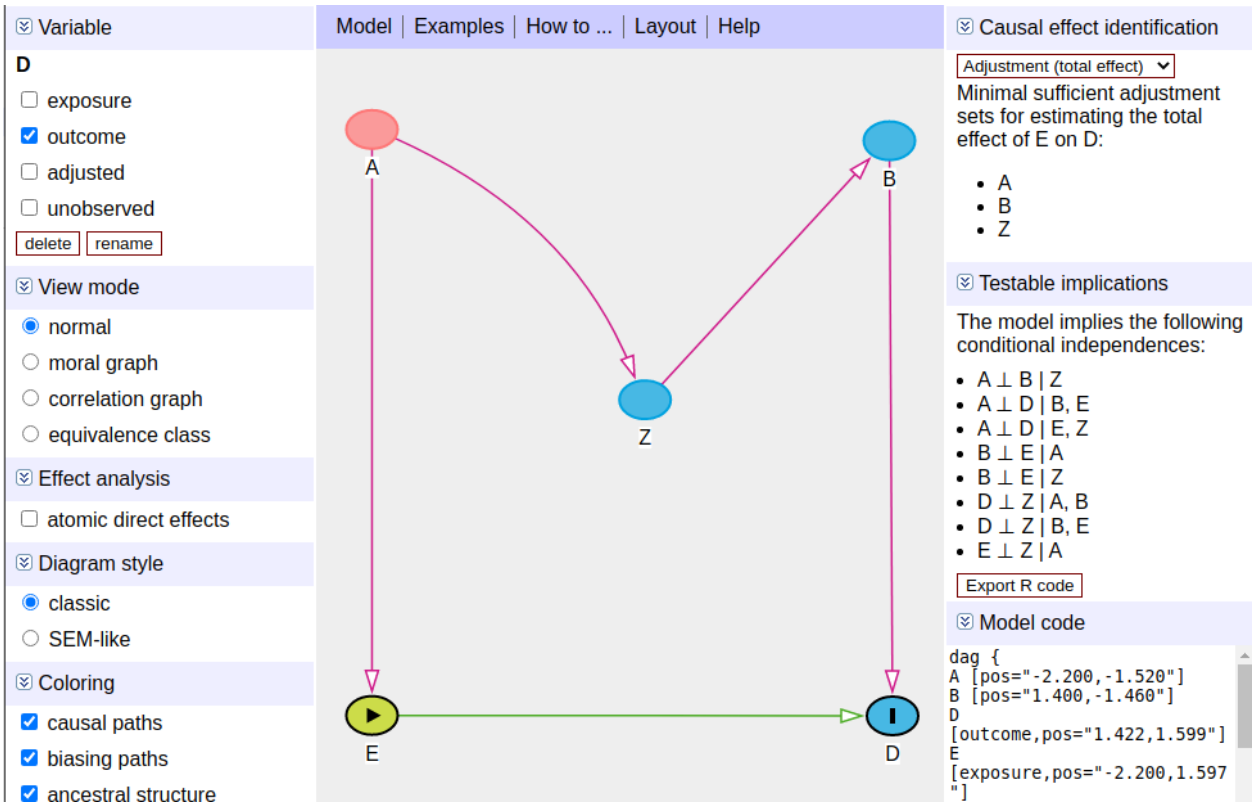


Figure 28: A screenshot of the Dagitty (Textor et al. 2016) interface, showing the minimal sufficient adjustment sets and the implied conditional independencies on the right-hand side.

# Chapter 4 – Reproducibility and Principled Data Processing

John McLevey, Pierson Browne and Tyler Crick

# Introduction

Scientific research continues to be plagued by a replication crisis, wherein results from published academic literature are found to be impossible to replicate, even by the teams responsible for arriving at said findings (H. M. Collins 1992; King 1995; Freese 2007; Baker 2016; Freese and Peterson 2017; Pridemore, Makel, and Plucker 2018; Christensen, Freese, and Miguel 2019). In response to this crisis, many highly ranked academic journals have become signatory to initiatives designed to enforce high standards for replication (Gleditsch et al. 2003; Hartzell 2015). Although there are many such initiatives – the specifics of which vary – most are concerned with ensuring that all published material is accompanied by an exhaustive, accessible copy of the data as well as the code used to arrive at the conclusions presented in the finished paper.

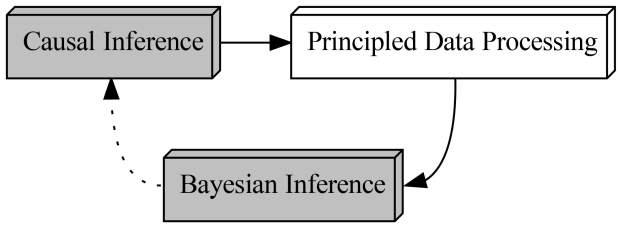


Figure 29: The foundational cycle, highlighting Principled Data Processing.

In this chapter, we consider the much-discussed replication crisis to be only one dimension of the more general project of doing research – and computational social science specifically – in reproducible and transparent ways. Later, we discuss the importance of reproducibility and transparency in computational social science, followed by a discussion of how these ideals are complicated by the realities of doing rigorous collaborative research in our field. We then present an overview of a framework called 'principled data processing' (PDP) which offers a core set of principles and practices that demand a commitment to reproducibility and transparency, on the one hand, and improving the quality and efficiency of collaborative research on the other (Ball 2016a, 2016b). Finally, we show one powerful way of organizing and managing computational social science research projects within this PDP framework using `pdpp` a simple and lightweight command line tool (Browne et al. 2021).

Our primary motivation with this chapter is to contribute to collective efforts to facilitate greater reproducibility and transparency in our field. As such, we use a number of vignettes and other hypothetical examples to connect high-level discussions of principles and ideals with concrete research practices. We also present some minimal examples of code that are sufficient for using `pdpp` in computational social science research projects, even though this breaks with the conventional emphasis on high-level literature reviews in handbook chapters. Finally, the pragmatically oriented portions of this chapter assume that researchers have already implemented version control (using tools such as `git`) into their workflows and view it as a necessary – but by no means sufficient – cornerstone of doing reproducible and transparent research.

# Reproducibility and Transparency

At risk of stating a truism, the volume of data available to computational social scientists has grown rapidly and is likely to continue to do so for the foreseeable future. This unprecedented wealth of data has emerged in lockstep with increasingly inexpensive access to more computing power than ever. To collect, manage, and process data at the scale that is currently available, researchers need to work in teams with diverse expertise, adopt new programming techniques and computational infrastructure, and apply new methods and models that are suitable for the types of data that are increasingly available. In contexts such as these, doing research that meets a high standard of reproducibility and transparency can be as challenging as it is scientifically and ethically imperative.

Until relatively recently, the opacity of data analysis was widely accepted: since it was usually impossible to examine the code and data one's colleagues were using to derive their findings, it was assumed that most researchers could reproduce their own work if necessary. There continue to be few professional incentives to do replication studies in science: they are generally devalued, require striving for conformity rather than novelty, can be perceived as hostile or threatening, and are extremely challenging to execute properly (R. Collins 1998; H. M. Collins 1998; Christensen, Freese, and Miguel 2019).

Furthermore, some types of research design, including mixed-methods designs that include qualitative components, do not lend themselves well to a narrow focus on replication. With the potential payoffs stacked against replication studies, very few scientists have made systematic attempts to reaffirm the reliability of published findings. Ongoing concerns over the replicability of scientific results have prompted certain academic journals to take action. For example, in 2012, the *American Journal of Political Science* began implementing a reproducibility standard for article acceptance. They propose, "[by] requiring scholars to provide access to their data and conducting our own replications on those data, we confirm the rigour of, and promote public confidence in, the studies we publish" (Jacoby, Lafferty-Hess, and Christian 2017).

Although there is a great deal of discussion about the 'replication crisis' in the academy and even in popular media, we see replication as just one important part of a larger and more pressing challenge: making reproducibility, transparency, and openness into cornerstones of our scientific cultures (H. M. Collins 1998; Christensen, Freese, and Miguel 2019; McLevey 2019, 2021). Doing so would help make replication easier and more common, as one of the main barriers to successful replication is that published articles and methodological appendices contain only a fraction of the information and *craft knowledge* necessary to execute even an imperfect replication (H. M. Collins 1974, 2016; Freese and Peterson 2017). An emphasis on reproducibility and transparency makes replication easier and less threatening, allows researchers to learn more quickly from one another's work, can accelerate discovery and innovation, and can be adapted when faced with the different sets of constraints imposed when working with sensitive data or qualitative methods.

Beyond scientific benefits, a lack of reproducibility and transparency in computational social science is, in many cases, *unethical*. O'Neil (2016) provides a powerful argument along these lines in *Weapons of Math Destruction*, which critiques computational models that are opaque, unregulated, and incontestable, yet are widely used in contexts that have tangible effects on people's lives and well-being. Through a series of case studies on the use of computational models in domains such as policing and education, she shows how algorithms that were considered by some to be more impartial or 'fair' ways of making decisions actually replicated deeply ingrained biases and, due to their opacity, were beyond reproach by those affected. In many situations, the problem lies with algorithmic feedback loops; a tangible example of this can be seen in policing, where predictive models exacerbate inequalities based on race and class (O'Neil 2016; Brayne 2017, 2020).

In the context of computational social science, reproducibility and transparency are the twin cornerstones of scientific validity – they are necessary but not sufficient components of doing research with high ethical standards. Opaque models and important research decisions made behind closed lab doors can exacerbate social inequalities and other injustices, and if estimates from a model impact life and well-being, the model should *at the very least* be open to interrogation and rectification. As computational social scientists, we should default to openness, reproducibility, and transparency as much as possible, but especially when using complex computational models or when we have any expectation that our work may have an impact beyond the knowledge we build in our immediate scientific communities. In other words, we must be mindful of the ethical problems posed by doing work that is not open, transparent, and reproducible and ensure we do not worsen social inequalities by fuelling harmful algorithmic feedback loops of the kind described in detail by O'Neil (2016).

Computational social science projects (and, indeed, all scientific projects) should also be auditable: reviewers unfamiliar with the project should be able to rapidly and easily determine how the various pieces of code in the project work together, at a high level, to produce the final output. Reviewers, moreover, should be able to dig into said code and identify problem areas based on discrepancies or anomalies observed in the output. Although traditional documentation (in the form of in-code comments and external written manuals) – if properly maintained – can help the project be auditable, the shifting scope and requirements of any computational social science project will often render such documentation obsolete. Wherever possible, teams should rely on self-documenting code: a paradigm in which directories, scripts, variables, functions, and data files are given succinct, descriptive names (Gentzkow and Shapiro 2014). If correctly implemented, measures such as these can help facilitate rapid and effective external reviews of the data and code used to develop and articulate findings.[132]

The practical challenges of reproducibility and transparency, then, are ensuring that data processing workflows are automated and auditable. Automated workflows eliminate the need for reviewers or other interested parties to become familiar with any kind of documentation before results are forthcoming. As long as the automation is foregrounded, reviewers can determine for themselves whether the project produces the results as presented. Gentzkow and Shapiro

[132] It should be noted that transparent code and data will not facilitate reproducibility tests that implicate the data-generating process itself; for example, the push for transparent data and code will not help in cases where researchers employ flawed or intentionally mendacious techniques for gathering or generating data.

(2014), for instance, recommend writing a shell script that runs a series of other scripts in the correct order and produces a finalized PDF, DOCX, or HTML output.

Many researchers and organizations have begun to innovate in response to the challenges presented by the need for collaboration and transparency in computational social science. The *American Journal of Political Science* (*AJPS*), for example, makes use of Harvard's Dataverse project, which is an open data platform for academic research data and has seen nearly five million data set downloads at the time of writing. To reproduce the analysis, the *AJPS* requires that the commands used to analyze the data be included on the Dataverse. These guidelines, however, are chiefly aimed at the syntax files used in statistical software such as Stata and R, with the main suggestion being that researchers intuitively name their syntax file(s) to match up with specific analyses in the submitted article. Computational social science workflows are not always structured in such a one-to-one translatable fashion. As such, we propose the adoption of an intuitive framework for compartmentalizing the pieces of computational social science projects and have developed a simple but powerful command line tool to help researchers more easily build their projects to fit such a process.

By focusing on reproducibility and transparency generally, rather than replication specifically, we emphasize practices and principles that can be used to greatly improve the quality of any research project, regardless of whether it is ever replicated, and to support more ethical research practices. In a sense, it is more important that all projects are auditable and reproducible rather than replicable, which is also more realistic, and can include research ethics as part of the equation. However, before we describe this framework and the accompanying Python package, we will take a closer look at how collaboration – essential though it is – can introduce many challenges to doing reproducible and transparent work.

## Adhocracy, Black Boxes, and the Collaboration Problem

It is likely that most computational social scientists reading this chapter have experienced the feeling of dread that manifests when the prospect of making changes to old code is raised. Perhaps new data have been gathered, or the research team wishes to make use of a previously overlooked variable, or the time has come to scale up the existing project to account for a broader range of cases and applications. Despite previously having everything working and being confident that the project accurately and responsibly processes the data sets your team is using, you are far less confident in your team's ability to extend or make changes to your existing code base without breaking everything.

Perhaps you think you're overreacting and take a peek at code that – despite being regularly relied upon – hasn't been thoroughly examined since it was originally written and tested months or even years ago. Instead of a well-organized, easily understandable procession of discrete steps in a sensible data processing pipeline, you're greeted by an incomprehensible primordial soup of data and code – dozens of files languidly suspended in the same directory with no organizing principles in sight.

The aforementioned experience – or experiences akin to it – are endemic in our trade. Gentzkow and Shapiro (2014) describe how writing and debugging code has become an indispensable and routine practice in computational social science. They also recognize that most social scientists are self-taught programmers and lack deep knowledge of relevant computer science and engineering practices. This, they claim, leads researchers and research assistants to adopt a 'seat-of-your-pants' approach to generating and organizing code, data, and results. For example, Figure 1 is a graphical representation of a data processing pipeline our team of six researchers created over time, using such a 'seat-of-your-pants' approach. Sorting out the horrible mess we created when developing the first version of this data processing pipeline prompted us to (a) start from the beginning again and (b) use the principled data processing framework described later in this chapter. The experience of working within this framework has been so positive that it led us to develop the `pdpp` tool to make it easier for our team, and others, to use this approach.

Our team developed Figure 1 by hand in an effort to clarify which scripts (developed by which researchers) were responsible for producing which dependencies. While the specifics are irrelevant, the meandering, disorganized structure should be familiar to anyone who has attempted to develop a collaborative data processing project. Of particular note here:

- *Too complicated and vague*: Even though each component of this project was developed carefully and tested rigorously on its own, the overall data processing pipeline is nearly impossible to understand as a result of its convoluted structure and equally convoluted conventions (or lack thereof) used to name scripts and their outputs.
- *Expired references*: Several scripts in the data processing pipeline either create or read in files (including some

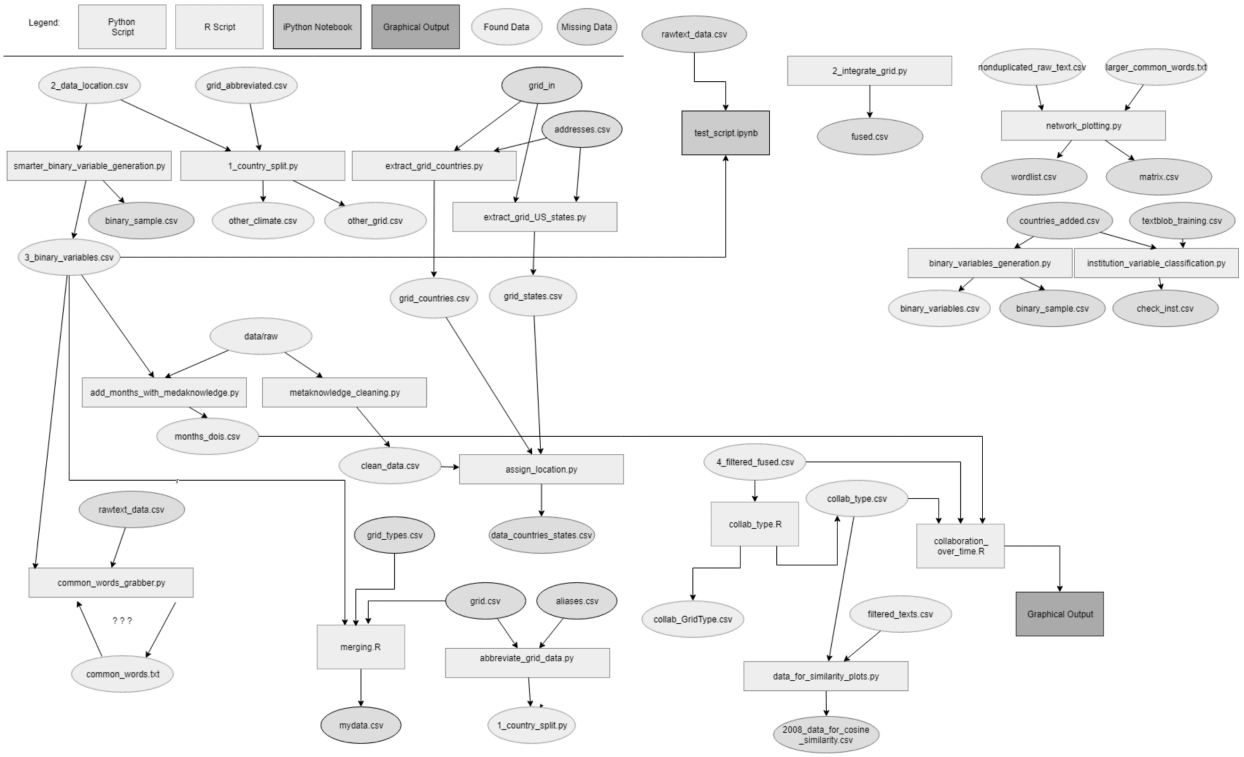


Figure 30: Project-level chaos resulting from a lot of reasonable context-specific decisions by experienced developers and researchers.

intermediate data sets) that are no longer formally part of the data processing pipeline. For example, a script might create a file to be read in by another script, but one of those scripts changes and the project directory becomes messier and more complex over time, and the processing pipeline becomes harder to understand because the actual files used in the pipeline are mixed in with other files that are not actually used.

- *Islanded tasks*: The diagram's upper-right corner contains several tasks that – for reasons we were never able to fully understand – make no contribution to the data processing pipeline or the overall project. They simply exist inside the project, doing nothing.
- *Circular dependencies*: In the lower left corner, a task produces its own input file; the file's only purpose in the project is to be fed back into the task that created it. While we can determine who created these files and when, all knowledge of their original purpose or previous connections to the larger data processing pipeline has been lost.

At this point, readers might be under the impression that the primordial soup of code and data that is revealed when 'pulling back the curtain' is the result of researchers' carelessness, neglect, or incompetence. Not so. We contend that – more frequently than not – these Gordian knots of data and code are the result of a series of well-intentioned decisions made by intelligent researchers who are well equipped to complete the tasks they are charged with. In the earlier example, the researchers' decisions were sensible at the time and included many 'best practices' from engineering and data science, such as unit testing. Moreover, the project itself worked: provided that the handcrafted list of step-by-step instructions for executing the pipeline was followed exactingly, the project depicted in the diagram was – astoundingly – capable of producing the desired outputs from the data set gathered for the project. When we set out to adapt the pipeline to a different data set, however, our team was intractably stymied by the convoluted, incomprehensible morass we had built.

The pipelines that computational social scientists use are often developed following an 'adhocratic' (or 'seat-of-the-pants,' in Gentzkow and Shapiro's words) logic. The term 'adhocracy' describes multidisciplinary teams of experts that have dynamic membership from project to project, non-hierarchical structures, and a relative lack of 'rules' (Mintzberg and McHugh 1985). The field of organizational management has recently had cause to re-examine adhocracy; despite its archaic origins, the structure it describes maps well onto the agile development processes frequently employed by contemporary technology firms (Birkinshaw and Ridderstråle 2010). As far as organizing principles go, adhocracy can

enable teams of individuals with diverse skills and expert knowledge to meaningfully contribute to a project and produce results in a relatively short time frame. If the only objective is to finish the data analysis and move on to something else, the kludged-together, on-the-fly style of problem solving that adhocracy engenders is an ideal means of completing the project's goals.

Adhocracy is also a pretty good model of what happens on many computational social science research teams: one or more principal investigators develop a research project that may include, for example, a series of related papers as output. First, the project itself, the papers, and the outputs that make it up are planned; then, some division of labour is agreed upon for the faculty investigators as well as student members of the research team. Although all members of the team generally contribute at least some code, it is fairly normal for student members of the team to take on responsibility for developing and testing a significant portion of the project's codebase. In a typical academic effort, research assistants will need to write code capable of collecting, screening, cleaning, summarizing, organizing, analyzing, and visualizing the project's data (Gentzkow and Shapiro 2014). Even for small teams, say of two to three people, collaboration brings together researchers with different backgrounds, preferences, strengths, weaknesses, and idiosyncrasies, which can introduce just as many problems as it solves.

University-based research teams carry a high risk of rapid collaborator turnover; as time passes, any given contributor's schedule can shift, their availability can change, and students graduate. Thus, academic projects need to be capable of unexpectedly losing collaborators and bringing new collaborators onto the team, all without paying the prohibitive cost of getting every new collaborator up to speed with every piece of code written for the project to that point. Unlike software development, where agile teams are only producing code and have dedicated staff groups devoted to documentation and oversight, academic research teams are producing code as only a portion of their work. It is unusual to have a student doing only documentation or code oversight; even if this were the case, those students would still need to be working towards graduation.

By allowing anyone to contribute to a project as best they can without forcing each contributor to learn in detail how every piece of the project's code works, adhocratic projects often fall victim to rampant 'black boxing.' In a heavily 'black-boxed' project, most team members know what any given piece of the project's code does and are aware that it works reliably, but they do not know how the code works or how the small pieces of code fit together to form an integrated multi-step data analysis workflow. What's more, black-boxed code often defies understanding and cannot be altered or repaired: everyone except the authors (and, in some cases, the authors themselves) might find it difficult to comprehend what a piece of code does. Gentzkow and Shapiro describe how their attempts to understand their research assistants' coding work were often abortive: in many cases, they found it more efficient to painstakingly rewrite the code themselves.

Aside from incomprehensibility, excessively black-boxed code can deleteriously impact the scientific rigour of a computational social science research project. It is often impossible, not to mention undesirable, for principal investigators to micromanage the work their collaborators – student or faculty – are doing. As such, contributors will inevitably make mistakes which, due to the many impenetrable layers of code written by many different contributors, will be difficult to detect, let alone locate or rectify. Gentzkow and Shapiro describe how coding errors led to their regression models producing incomplete, inaccurate results, or situations where discrepancies in definitions of samples between various pieces of code weren't detected until after the results had been submitted to a scholarly journal. Gentzkow and Shapiro's accounts mirror our own experiences: we were forced, at one point, to redesign and implement a data processing pipeline that was functional but which we could no longer understand or generalize to other parts of our research project.

In sum, adhocracy is often dropped into an academic environment but does not thrive therein – it needs some modification or structure that is, itself, reproducible. Without structure, the highly competent and well-intentioned members of research teams can make many individual decisions that are reasonable at the time – and for some small portion of a larger project – but in the big picture this often results in incomprehensible data processing pipelines and an excessive amount of black boxing. In other words, reasonable decisions made by experts about some specific component of a project can still result in a code base that is incomprehensible even to members of the team and far from the ideals of reproducible and transparent science. The core struggle of adhocracy generally, and collaborative computational science specifically, is to convince all members of the team to voluntarily adopt a set of principles designed to structure their combined work without hampering the teams' capacity for flexibility and adaptability.

Researchers are aware of the issues we have discussed in the two preceding sections. In the section that follows, we will introduce a framework developed by Patrick Ball (2016a, 2016b) to ameliorate the struggles inherent to collaboratively

producing auditable, transparent data processing pipelines.

## Principled Data Processing

This principled data processing (PDP) framework was initially proposed by sociologist and statistician Patrick Ball of the Human Rights Data Analysis Group (Ball 2016a, 2016b) in the context of their statistical research on mass killings and other acts of political violence. Ball outlines a set of core principles that, as we have argued to this point, should be explicitly adopted in research workflows:

1. *Transparency*: Every portion of the project is visible, and the role it plays can be easily derived from its name (in broad terms, at least).
2. *Auditability*: The project (or components thereof) can be successfully executed on different platforms and in different software environments by analysts who were not necessarily involved in its creation.
3. *Reproducibility*: The project's expected results can be generated by anyone with the same data, code, and software dependencies.
4. *Scalability*: The project's constituent elements are capable of being used for a larger number and wider variety of inputs, outputs, and purposes than originally intended.

Ball describes the system as one designed to ensure that the structure and functionality of a project and its constituent parts are easily understood: not only by other members of a team, but also by the researchers who wrote the code in the first place. If followed by all members of a team, a data processing framework that is anchored in these four principles can buttress against the collaboration and transparency problems facing computational social scientists.

The first – and arguably most fundamental – portion of the PDP framework calls for projects to be split up into a series of discrete tasks, each of which is stored in its own separate, dedicated directory. Each task's directory should have a succinct name that allows newcomers to rapidly ascertain the task's role in the project's overall structure.

Second, each task directory should follow the same organizational scheme. Rather than allowing data, source code, and results to freely mingle, Ball recommends organizing the contents of each task directory into three subdirectories: (a) the 'input' directory, which contains all of the data required by the task; (b) the 'output' directory, which is where all of the task's results are stored; and (c) the 'src' directory, which holds all of the source code necessary to properly and reliably transform the inputs into new outputs.[133]

The small tasks that make up a larger project are interconnected. Aside from the first and final tasks in any given project, every task retrieves its input data from the output of another task, and the output it produces is the input of some other task. The relationships between tasks are formalized by a series of symbolic links (or 'symlinks'), which are lightweight placeholders in a computer's filesystem that, when accessed, point the accessor to a different file or location elsewhere on disk. Each task's input directory contains one or more symlinks pointing to files in other tasks' output directories. This serves three purposes: first, the use of symlinks allows for the separated directory structure to be maintained without necessitating the use of duplicate files; second, the symlinks allow users to determine the project's overall dependency structure (i.e., which tasks are executed in what order); third, the use of symlinks ensures that any changes made to any task's output file will be propagated to the tasks for whom that file is a dependency, ensuring that all tasks stay up to date without any need for manual oversight.

The final critical aspect of Ball's PDP framework is the use of GNU Make via a Makefile. A Makefile is a general-purpose tool used to intelligently execute a series of shell commands – it is commonly employed to compile complex software. GNU Make allows users to split the commands they wish their Makefile to execute into a series of discrete steps, each with their own set of 'dependencies,' 'targets,' and 'actions.' has many more features and is capable of much more than what is detailed here. For the purposes of focus and brevity, our description of GNU Make only covers those aspects pertinent to Ball's PDP framework.] 'Dependencies' are what a task needs before it can run, 'targets' are what a task produces, and 'actions' are the specific steps a task takes (usually in the form of shell commands) to turn the 'dependencies'

[133]While 'input,' 'src,' and 'output' should be the only project-relevant directories found inside any given task, Ball allows that other directories may be included. One such directory Ball mentions by name is the 'hand' directory, which is used to develop new source code in an interactive environment such as a Jupyter Notebook. Ball recognizes that Jupyter Notebooks can be remarkably useful for developing new code, but they are not suitable for use with the overall project and as such should not be placed in the 'src' directory.

into the 'targets.' Once properly established, a project-level Makefile allows users to run the entire project using a single command – make. Once invoked, GNU Make will determine how the various tasks should be executed – and in what order – based on the chain of dependencies defined therein. GNU Make also determines whether a task needs to be rerun; tasks are only executed if their dependencies have changed since the last time they were run (if the dependencies haven't changed, then the targets should be identical and do not need to be generated again).

In our view, Ball's principled data processing is an excellent protocol that directly addresses the problems outlined in our previous sections on transparency and collaboration. Using a series of automated tasks as the quanta of an analytical workflow is immensely helpful in ameliorating many of the pitfalls inherent to data processing, including onboarding new collaborators, documentation, writing transparent code, and ensuring reproducible outputs.

### Potential Drawbacks

As useful as Ball's principled data processing is, some might encounter difficulties implementing the protocol. Strictly following Ball's recommendations would require every member of a team to familiarize themselves with GNU Make, Linux/Unix, and YAML. While these requirements are sensible in contexts where everyone can be reasonably expected to have some background in some or all of the aforementioned areas, these same requirements can present an onerous or even insurmountable barrier in others.

On a more practical level, the very act of creating and maintaining Makefiles can become prohibitively costly. In projects featuring a comparatively large number of files spread across a large number of tasks, the amount of menial labour (in the form of typing or copying file and directory names with exacting precision) required to simply create a Makefile can significantly impact the time it takes to automate a project. Even for those familiar with GNU Make, the spectre of updating a Makefile after a relatively minor change in a large project's dependency structure can be daunting.

When we first adopted the PDP framework, we used the same tools in the same configuration that Ball (Ball 2016a) proposes using. However, we found that the use of highly complex tools like Make actually made aspects of our collaborative projects *more* opaque to some members of the team, especially those who have expertise in statistical and computational models but know little about open source computing culture and tools. This caused an unofficial divide on the team where some collaborators were more capable of making contributions within the PDP framework than others were. To reduce the amount of necessary technical training, we developed a command line tool that dramatically simplifies working within the PDP framework.

## Introducing `pdpp`

`pdpp` is a command-line tool for Python3.6+ that simplifies principled data processing for computational social science projects (Browne et al. 2021). Although it is written in Python, it can be used to manage projects written in any programming language whose scripts can be run from the command line (e.g. R). `pdpp` views analytical projects as being composed of many small, granular, interconnected tasks: each task takes an input, runs the input through one or more pieces of source code, and produces one or more output files. The outputs from most tasks serve as the input for other tasks. In aggregate, the tasks form a chain that takes the project-level input (e.g. a .csv file with data) and produces one or more forms of project-level output (which can take the form of papers, visualizations, statistical model results, other data sets, and so on).

`pdpp` offers a number of advantages over a strict interpretation of Ball's original principled data processing framework.

### Ease of Use

Our primary motivation for developing `pdpp` was to give a broader range of researchers easier access to the benefits of principled data processing. As such, while developing the package, ease-of-use was one of our chief priorities; we wanted to ensure that even new and inexperienced programmers would find our package simple to learn and apply. After all, the principles are more important than the tools. We also designed `pdpp` to reduce the amount of menial labour necessary to implement Ball's recommendations for principled data processing. When projects get very large and involve shuffling around vast numbers of individual data files, the simple act of copying and pasting filenames into your automation suite of choice can become prohibitively time consuming. The `pdpp` package handles all of this automatically and allows you to rapidly assemble and reconfigure complicated projects.

### Suitable for Non-Expert Programmers

When working with pdpp, a basic understanding of Python can be helpful in some edge cases, but it is far from necessary: all of pdpp's functionality can be applied without the need for any direct code alteration or scripting. Knowledge of the command line from your operating system of choice is all that is needed. This stands in contrast to Ball's principled data processing, which requires users to be familiar with syntax for GNU Make, YAML, and Unix/Linux.

### Cross-Platform, Cross-Language

Another benefit of using pdpp is its cross-platform, cross-language compatibility: pdpp can be used by most platforms, with most languages, on any of the major operating systems, and for most projects involving the preparation and analysis of data sets. While pdpp is written in Python, it smoothly interfaces with other languages without requiring users to write (or even understand) Python code. As such, pdpp is an excellent choice for researchers who are familiar with, for example, R but have little to no experience with any other programming environments.

### Minimal Downstream Workflow

We wanted to ensure that the projects developed using pdpp can be shared, iterated upon, and reproduced without forcing end users (such as journal reviewers, colleagues, etc.) to familiarize themselves with pdpp. To run projects built using pdpp, all that is required is an installation of pdpp.

## How to Use pdpp

Every pdpp project consists of one *import* directory, one *export* directory, a dodo.py file, and one or more tasks (each contained in their own descriptively named directory). The *import* directory is used as the common starting point for all locally stored data used throughout the project: all of the project's input data should be stored in the *import* directory.[134] The *export* directory is used as the common terminus for all project-level end products and provides a convenient centralized location for gathering all needed output files. The dodo.py file is used by the doit package, upon which pdpp's automation suite is built.

As previously described, pdpp is designed to separate data analysis pipelines into a series of discrete tasks. Each of these tasks should be descriptively named and are usually accompanied by three pieces of functional metadata:

1. Dependencies – the files a task will need to execute properly.
2. Actions – the commands a task will execute.
3. Targets – the files a task will produce as a result of the actions it will take.

The pdpp package and Ball's original formulation of the PDP framework share similar goals, but the routes they take to achieve them differ. pdpp makes three assumptions which abstract away much of what must be explicitly stated in Ball's framework: the first assumption is that tasks will typically be small in scope, self-contained, and written in Python or R.[135] Note that this is merely pdpp's default behaviour. pdpp can be used with scripts written in any language, provided those scripts can be executed using shell commands. By default, pdpp assumes that each task's source code will be composed of a single script, as tasks that require multiple scripts are probably an indication that the task needs to be split further into multiple tasks. pdpp can still automatically handle tasks whose source code consists of multiple scripts, provided each script is written in the same language and has the same file extension. Tasks with two or more scripts written in different languages must be manually implemented. When a project is run, pdpp will execute all of the scripts (written in languages it recognizes) contained in each task's src folder in an arbitrary order.

The second assumption is that any files in one task's output directory can be considered its targets if – and only if – they are dependencies for other tasks. Each task's targets are dynamically determined at runtime based on other tasks'

[134] This requirement to store does not apply to dynamically retrieved or generated data; data pulled from a URL or queried from a database does not need to be stored in the _import_ directory. This also does not apply to data files created as part of the project's execution. The _import_ requirement exists to ensure that the dependency structure between tasks is preserved for all project-level input data files.

[135] Although pdpp only has automated support for Python and R at the time of this chapter's publication, more languages will be added in post-release updates.

dependencies. This means that rather than having to explicitly list each task's targets, `pdpp` infers what each task's targets are based on what files other tasks expect to be able to get from it. From a conceptual standpoint, this assumption may seem tautological, but in practical terms, it removes a remarkable amount of menial upkeep.

The third and final assumption is that anything identified as neither a dependency nor source code is unimportant to the project and can be safely ignored.

In typical use cases, the assumptions previously enumerated permit `pdpp` to rapidly build robust, reliable dependency structures based only on users' indications of which tasks depend on which files elsewhere in the project.

`pdpp` is built around five main shell commands, each of which handles one aspect of the package's functionality. Note that the following commands must be entered in a project-level directory (either the main project directory or one of the subproject directories). `pdpp` commands will fail to resolve if the console's current working directory is a non-project location. In alphabetical order, the commands are:

### `pdpp graph`

The `pdpp` graph command can be used to visualize your project's dependency structure and, in line with Ball's original vision, document the project's structure dynamically without requiring keeping written documentation up to date. It has four modes:

1. sparse – graphs only the dependencies between tasks.
2. source – graphs task dependencies alongside representations of each task's source code.
3. file – graphs the dependencies between tasks and individual files.
4. all – includes all elements from both source and file modes.[136]

In all cases, `pdpp` graph will produce a graph of your project's dependency structure in either the .pdf, .png, or .jpg format.

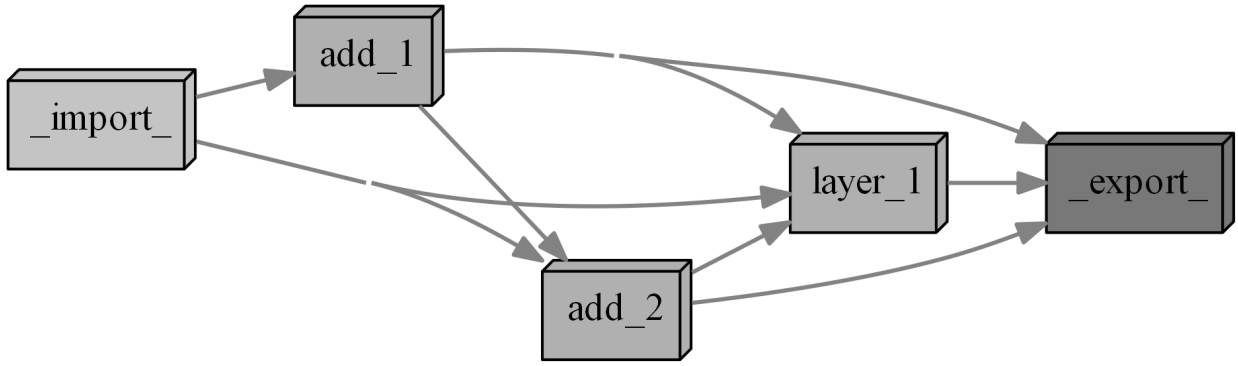


Figure 31: Sparse Mode

### `pdpp init`

The first step in creating a `pdpp`-compliant project involves creating an empty directory, navigating to it, and executing the `pdpp` init command from within. This command populates the directory (which we will refer to from now on as the 'project directory') with all the infrastructure necessary to support `pdpp`. The `pdpp` init command does the following:

• Creates an *import* directory and an *export* directory, used for storing the entire project's input data and outputs, respectively.

• Creates a generic .gitignore file used by version control to selectively exclude certain files.

• Creates a dodo.py file used by the doit package, which `pdpp` uses to handle automation.

[136] Be warned: the 'all' graphs can become unreasonably messy, even for modestly sized projects.

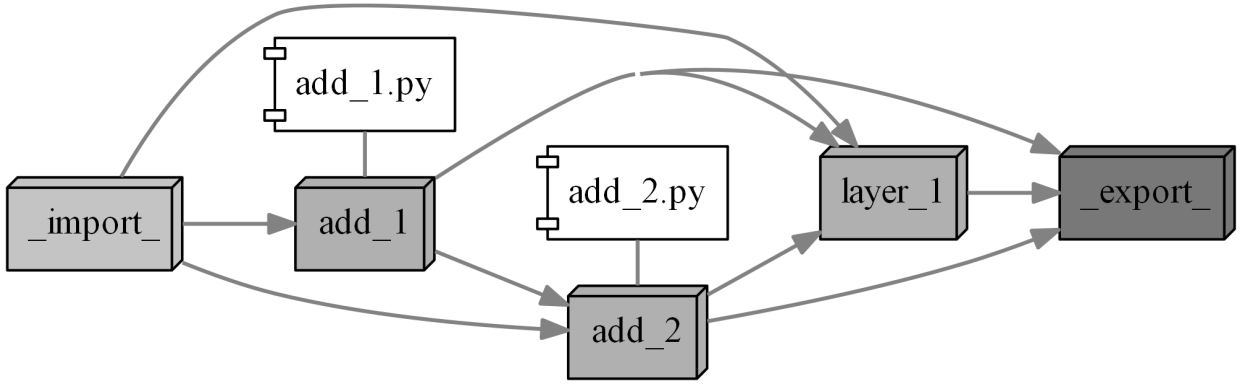


Figure 32: Source Mode

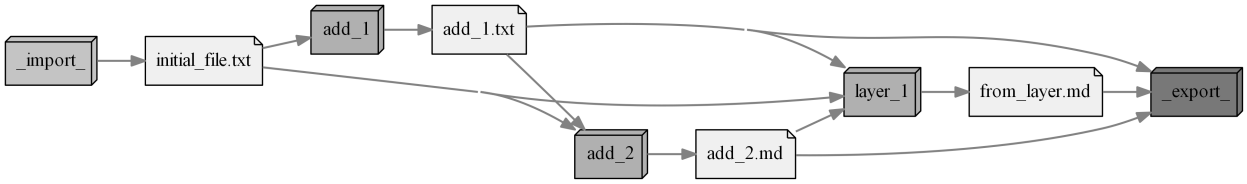


Figure 33: File Mode

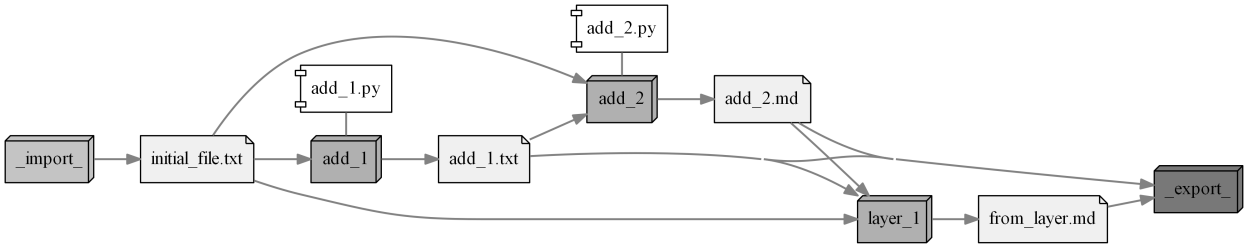


Figure 34: All Mode

### `pdpp new`

The `pdpp` new command is used to create a new directory structure corresponding to one task. Upon executing the command, the user will be asked to supply the task's name, which should follow the following conventions: unique within the project, no upper case characters, no spaces, and no non-underscore special characters (underscores should be used in place of spaces).[137] For example, clean_tweets is a valid name for a task, whereas CleanTweets and Clean tweets are not.

The command creates a task directory, which itself contains input, output, and src directories, as well as a task-specific `pdpp_task.yaml` file, which `pdpp` uses to store execution metadata about the task. By default, a blank Python script will be added to the `src` directory, and `.gitkeep` files will be included in each of the created directories to ensure their preservation via version control software. Users should feel free to begin writing code in the provided blank Python script or replace it with a script in a language of their choosing.[138]

After the new task has been named and its directory structure populated, `pdpp` will prompt users to indicate which files contained elsewhere in the project are dependencies for the new task. This is done in a two-step process: first, `pdpp` asks users to indicate which of the project's other task directories contain dependencies for the new task; second, `pdpp` presents users with the outputs[139] from the selected tasks (alongside all of the files in the *import* directory), and prompts users to choose which of the individual files are needed by the new task. Recall that all the pre-existing data files used by – but not produced by – the project's tasks should be stored in the project's *import* directory.

### `pdpp rig`

The `pdpp` rig command is identical to the `pdpp` new command, except that it is intended for use with existing tasks. The rig command allows users to reconfigure the dependencies of any task: after entering `pdpp` rig, users will be prompted to select which task they wish to 'rig,' at which point users will be first prompted to indicate which other tasks (including *import*) contain dependencies for the task being rigged, and then asked to identify which of the other tasks' files are dependencies. `pdpp` rig can be used to configure the *export* task, which is necessary to ensure that the project's overall output files (visualizations, statistical models, etc.) are properly identified as targets of their respective tasks.

### `pdpp run`

The `pdpp` run command is used to execute each of the tasks in the user's project from start to finish. It is the equivalent to running Make in Ball's original implementation.

## Sample Workflow

In this section, we present a sample workflow designed to familiarize users with the `pdpp` package's basic functionality and to make the preceding discussion more concrete. For this example, we'll use a modified version of Karan Bhanot's "Dataset Creation and Cleaning: Web Scraping Using Python" project (Bhanot 2017). Bhanot's project is designed to serve as an example of how one might use information scraped from a web page to create a data set for analysis. It begins by gathering information from Wikipedia, converting the retrieved HTML data into the pandas dataframe format, and then saving the resulting dataframe to disk as a CSV (comma separated values) file. The second step in the process involves loading the saved CSV, cleaning the data set, altering covariates, and saving the results to disk in another CSV, ready for further analysis.

Readers are encouraged to replicate the workflow described later: all that is required is an installation of Python (version 3.6 or greater), pandas, `pdpp`, and access to the example scripts in the public `pdpp_`example repository at https://github.com/pbrowne88/`pdpp_`example.

[137] These conventions are necessary to avoid the complications that arise from how various languages and operating systems interpret special characters and capitalization.

[138] Note that at the time of writing, `pdpp` does not support tasks that have multiple scripts written in different languages. This is because `pdpp` directly runs project scripts from the shell and needs to know which shell command is appropriate to use on which files in an `src` directory. Advanced users will be able to create custom workarounds, but most users should avoid creating tasks that rely on multiple pieces of source code from multiple languages. If your task has two pieces of source code, it is almost always more appropriate to split it into two separate tasks.

[139] Defined as files in the task's output directory.

The best way to start a new project using `pdpp` is to create an empty directory, navigate to it, and execute the `pdpp` init command. In the following example, the project directory is named example_project, but the name is unimportant. It should now contain the *import* and _export directories.

Many projects will begin with one or more pre-existing dataset(s) saved to disk; in such cases, all relevant data should be stored in the newly created *import* directory, which can be found inside the project directory configured by `pdpp` init. This example, however, does not rely on extant data: the scrape.py script is responsible for collecting and storing data.

This example project consists of two tasks – 'scrape' and 'clean' – which will gather and process our data set, respectively. Now that our project directory is configured, we can proceed by creating a directory for our first task – 'scrape.' Execute the `pdpp` new command and type 'scrape' when prompted to name the new task. `pdpp` will then inquire about which of the project's other tasks contain dependencies for the newly created task. Since this is the project's first task and the project doesn't require us to store any data in the *'import'* folder, the 'scrape' task has no dependencies; all of the other tasks should be left deselected.

When creating a new task, `pdpp` automatically creates a Python script with the same name as the task directory and places it in the task's src directory. At this point, that placeholder script should be replaced with the scrape.py script from the example repository. Once the script is in place, navigate to the src directory in the scrape task's directory and run the script with your Python interpreter from the command line.

You can confirm that the script ran correctly by listing the contents of the scrape/output directory: it should contain a file called dataset.csv – this file should contain one row for each United Nations member state and columns for 'Country,' 'Total Area,' 'Percentage Water,' 'Total Nominal GDP,' and 'Per Capita GDP.' While examining dataset.csv, you might notice that much of the data collected and stored by the scrape.py script is riddled with non-numerical symbols and that these symbols are prevalent enough to seriously complicate computational analysis. Fortunately, the clean step is designed to ameliorate this issue.

After navigating back to the project directory, execute the `pdpp` new command – this time, use 'clean' as the name of the new task. The new clean task is designed to take in the dataset.csv file produced by scrape and prepare it for use in an analytical workflow: as such, dataset.csv is a dependency of the clean task. When `pdpp` prompts us to indicate which other tasks contain dependencies for the clean task, we can select the scrape task (and only this task) by highlighting it with the spacebar.

At this point, `pdpp` will present us with another menu of all the eligible output files contained in all of the tasks we indicated earlier. Since we only indicated one task (scrape) and that task's output directory contains only one file (dataset.csv), we are given only one option. Select it and proceed!

Once this is done, `pdpp` will proceed by creating a hard link to dataset.csv in the scrape/output directory and placing the link in the clean/input directory. This means that the dataset.csv file can now be accessed by the clean task, and any changes made to scrape/output/dataset.csv will be automatically reflected in clean/input/dataset.csv.

As previously, replace the placeholder clean.py script with the complete version from the repository, navigate to the clean/src directory, and run the script with your interpreter. If everything worked as intended, there should be a file titled final_dataset.csv in the clean task's output directory. The entries in each of the columns should be much cleaner now, though not necessarily correct: the Australian landmass, for example, is almost certainly not 76% water! If you're feeling bold, try digging into this example project's code and the dataset.csv file to see if you can isolate where the issue is coming from. In a small example project such as this one, it should be a relatively easy task; in larger projects with dozens of tasks, structural clarity will be indispensable when tracking down and ameliorating error sources.

Now that we have implemented both of the project's tasks, we can finalize the project. First, to ensure that all of the project's end products are stored in the same location, ready to be easily accessed by other scripts or projects, use `pdpp` rig on the *export* task and list clean/final_dataset.csv as a dependency.

Finally, we can use `pdpp` graph to produce a graphical representation of our project's structure, shown in 6.

Now that the project's structure is in place, you can use `pdpp` run to rerun 'stale' tasks. This will be useful in cases where changes have been made to input data or a piece of source code.

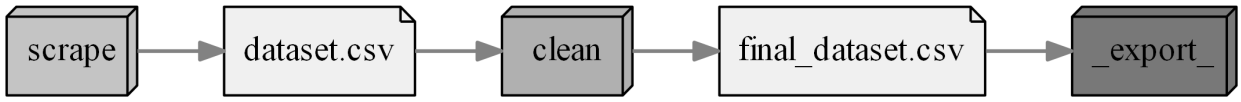


Figure 35: The project's final structure as visualized by `pdpp graph`

## Conclusion

In this chapter, we highlighted the ongoing struggle to embed adherence to reproducibility and transparency into the practice of computational social science, which we see as a superset of – and requiring much bigger changes than what is called for in – the ongoing discussion of the role of replication in the sciences. A strong emphasis on reproducibility and transparency in computational social science would make replications easier and likely more successful while also being compatible with mixed-methods research that does not lend itself well to the replication model. In addition to its many intellectual benefits for our community and other individual investigators and teams, reproducible and transparent research practices also enable us to do more ethical computational analyses, especially when our methods and models may have tangible impacts on individuals' lives and well-being. We introduced a framework – principled data processing – that can help accomplish this and demonstrated one way of adopting this framework that relies on a new command line tool called `pdpp`.

# Chapter 5 – Ameliorative Arguments for Bayesian Inference in Sociology

Pierson Browne

# Introduction

For nearly a decade, the quantitative social sciences have been adrift in a 'replication crisis' of their own making (Gelman 2016; Baker 2016; Wiggins and Christopherson 2019; Clayton 2021). For the latter half of the 20th century, the high standards of statistical significance placed on most published quantitative work had been interpreted as a near-guarantee that a result was generalizable and could be relied upon to replicate – thus obviating any need to confirm results via replication studies (Gigerenzer 2018). In 2015, the Open Science Collaboration published the results of their good-faith attempts to replicate 100 published psychological studies; only 36% of the results so tested were successfully replicated (Collaboration 2015) – similarly dismal results from most other fields soon followed (Clayton 2021).

Among the root causes of the replication crisis is the widespread use of 'Classical' statistics (Clayton 2021), and – in particular – Null Hypothesis Significance Testing (Gigerenzer 2004, 2018; Szucs and Ioannidis 2017; Gelman 2018; Colling and Szűcs 2021; Anderson 2020; Bird 2020). Scholars and commentators beyond counting have penned missives calling for urgent action; their proposals have ranged from modest changes in the language of statistical significance (Dushoff, Kain, and Bolker 2019), to overhauls in the use of the null hypothesis (Szucs and Ioannidis 2017), to the wholesale abandonment of statistical significance (McShane et al. 2019). To date, none have garnered sufficient succour to make any lasting impact. The social sciences remain in desperate need of a solution.

In this chapter, I will argue that social scientists are aware of the need for a better approach to quantitative research, but have not yet mounted a concerted effort capable of addressing the root causes of the replication crisis. I start from the now well-established position that the standard 'Classical' approach employed in the quantitative social sciences (henceforth referred to as the 'Frequentist' paradigm) is incapable of overcoming its intrinsic flaws. The Bayesian statistical paradigm, meanwhile, represents a well-developed alternative that intrinsically avoids the pitfalls that precipitated the replication crisis (Clayton 2021) and, thus, should be regarded as a foundational pillar of all quantitative sociology.

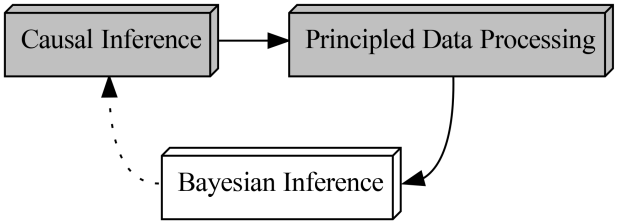


Figure 36: The foundational cycle, highlighting Bayesian Inference.

Despite the presence of this viable successor, the Frequentist paradigm continues to predominate the social sciences (Lynch and Bartlett 2019). In response to the foregoing, this chapter proposes and substantiates a 'ameliorative' mode of argumentation in favour of Bayesian Inference. Ameliorative arguments are intended to break free of Gelman's (2008) typology wherein most arguments for Bayesian inference can be reduced to one of three unhelpful modes. Ameliorative arguments, by way of contrast, take seriously the concerns Frequentist social statisticians may have about Bayesian inference and attempt to show how Bayesians could rectify said concerns. In the penultimate section, I introduce `pyKrusch`, a Python package I developed to advance one key portion of the Bayesian ameliorative project: visualizing model structure and prior specification (Browne 2021).

# Ritual and Illusion

According to Gigerenzer (Gigerenzer 2004), the practice of statistics in the social sciences has become a mindlessly routine affair. Social scientists are trapped in what he calls "The Null Ritual" (Gigerenzer 2018), wherein quantitative research is near-ubiquitously performed using the following unstated guidelines:

1. Statistical research should be conducted in the absence of substantive predictive hypotheses.
2. Statistical research should always assume an arbitrary null hypothesis of "no difference" or "no correlation".
3. Statistical research should reject null hypotheses using a threshold of $p < 0.05$ and interpret this rejection as justification for interpreting a model's estimates as the 'alternative' hypothesis.

Tragically, the null ritual – while masquerading as the vanguard of the Frequentist statistical paradigm – is an unprincipled amalgam that borrows heavily from incompatible statistical techniques.

Null hypothesis testing stems from Ronald Fisher's work (Ronald A. Fisher 1956; R. Fisher 1955), which involves testing a 'null' hypothesis – it neither requires nor allows for consideration of a pre-specified alternative hypothesis. Fisher envisioned null hypothesis testing as a technique that could prove helpful in cases where researchers were unaware of viable alternative hypotheses. Using Fisher's approach, researchers were expected to judiciously select and interpret a test statistic on its own merits. Fisher primarily intended his doctrine to employed in service of determining which avenues of research might be of interest, as his tests identified results that had low probabilities of arising due to random chance – in this sense, Fisher emphasized significance testing as an inductive, exploratory tool (Gigerenzer 2018; Perezgonzalez 2015).

The Neyman-Pearson school (Neyman and Pearson 1933; Pearson 1962) saw little value in Fisher's null hypothesis testing; they posited that the 'null' hypothesis needed to be tested against a pre-specified *alternative hypothesis* to be of any use. For Neyman-Pearson, significance was imbued with meaning via a researcher's careful calibration of sensitivity and power levels (known as $\alpha$ and $\beta$, respectively), designed to provide predictable rates of Type I and Type II error (false positives and false negatives, respectively). While the Neyman-Pearson approach also made use of the null[140], it required the explicit specification of a pre-determined alternative hypothesis against which the null was to be tested – placing it firmly in the deductive camp (Gigerenzer 2018; Perezgonzalez 2015).

Fisher's and Neyman-Pearson's respective approaches to statistical inference are incompatible and were, in fact, developed in cognisant opposition to one another (Gigerenzer 1998). Despite their immiscibility, they do share one crucial commonality: both approaches enshrine the experience and discretion of the individual researcher as *sine qua non* – the two techniques' results are meaningless in the absence of researcher judgement (Gigerenzer 2018). There is a bitter irony, then, that the names Fisher, Neyman, and Pearson have been used to justify the utility of Null Hypothesis Significance Testing – a statistical ritual that grafts together components from Fisher and Neyman-Pearson with an eye towards eliminating the researcher from the research process (Perezgonzalez 2015).

The origins of the now-ubiquitous NHST ritual can be traced back to a series of textbooks released in the mid-1900s that endeavoured to create a 'purely objective' form of automatic inference: to do so, they borrowed elements from both camps that eliminated the need for researcher judgement wherever possible. The result: Null Hypothesis Significance Testing. Thanks to its pernicious claim of offering objective, one-size-fits-all statistical inference, this unprincipled hybrid spread like wildfire through textbooks, classrooms, and professional organizations (Perezgonzalez 2015; Gigerenzer 2018). In the contemporary social sciences, NHST is the unchallenged norm (Lynch and Bartlett 2019).

[140] Neyman and Pearson evoked the 'null' hypothesis in a different sense than Fisher, and so the concept does not flawlessly port from one school to the other; nevertheless, both schools emphasize explicitly testing against a single 'less-interesting' hypothesis, and both referred to it as the 'null' (Perezgonzalez 2015).

Far from being a harmless fudge employed in practical settings, results derived from NHST are only weakly meaningful at best; in most social scientific settings, they actively mislead. Gigerenzer (2018) identifies three NHST 'illusions' that most quantitative social scientists believe:

- **The Replication Illusion**, wherein the value $1 - p$ is interpreted as the probability of any future replication study producing a significantly similar result, which means that replication studies are never needed where significant results have been produced.
  - *In reality, a $p$-value contains no valuable information about the probability of replication. A low $p$-value cannot legitimately justify the omission of replication studies.*
- **The Certainty Illusion**, wherein strongly significant NHST results are taken as 'proof' of an effect, or – equivalently – high values of $p$ are interpreted as proof of the null hypothesis.
  - *The above is untrue: statistical methods are incapable of producing proof.*
- **Bayesian Wishful Thinking**, wherein the $p$-value is interpreted as the probability of the null hypothesis being true, or $1 - p$ is interpreted as the probability of an alternative hypothesis being true.
  - *While such probabilistic interpretations are valid for **ranges** of hypotheses under the Bayesian paradigm (more on this later), the Frequentist school strictly forbids the application of probability to hypotheses. What's more: $p$-values contain no meaningful information about the probability of a substantive hypothesis being true or false.*

In light of these widely-believed illusions, the replication crisis should come as no surprise: for nearly a century, the social sciences have been publishing under the assumption that NHST gives reliable information about the probability, certainty, and/or replicability of experimental results when it has – in fact – done nothing of the sort.[141]

### Mending the Looking-Glass

When a student of the quantitative social sciences is first confronted with evidence of NHST's inadequacy, it would be reasonable to expect them to react – and I speak from personal experience here – with disbelief, shortly followed by a desire to repair this cornerstone of contemporary statistical inference. One need not look far to uncover a bacchanalian trove of proposals, suggestions, and tweaks published in the hope of shoring up NHST's shortcomings. To list but a few, authors have proposed that we:

- Insist on pre-registered hypotheses (Nosek et al. 2018).
- Abandon the use of the term 'Significance' in favour of 'Clarity' (Dushoff, Kain, and Bolker 2019).
- Abandon the use of p-values (McShane et al. 2019).
- Abandon null hypotheses; insist on competing substantive hypotheses (Gigerenzer 2018).
- Implement better education about the meaning and interpretation of common statistical tests (Perezgonzalez 2015).
- Set lower thresholds of $p$ for publication (Ioannidis 2018).[142]
- Insist on replication before publication (Schooler 2014; Key 2016).

If it were in the power of any individual (or group thereof) so described, the social sciences would likely enshrine one or several of the rectifying changes suggested by statistical and domain experts. Unfortunately, the unhappy fate of each of these well-intentioned endeavours is sealed. This is because the entire foundation for the Frequentist statistical paradigm is balanced atop a now thoroughly discredited fallacy, which I'll explore in the subsequent section.

## Sleepwalking to Significance

To grasp how the Frequentist paradigm has backed itself into an intractable corner, we must start by articulating fundamental statistical desiderata. In our quest to learn more about the world, we are primarily interested in assessing the plausibility of some hypothesis ($H$) by using data ($D$) to draw inferences about said hypothesis. To that end: the 'inferential probability' of a proposition is what all social statisticians wish to know.[143] We can describe inferential probability as "the probability of a hypothesis conditional on some observed data", or using standard statistical notation:

[141] As have many other disciplines, but discussing statistical practice outside the social sciences, and sociology in particular, is beyond the scope of this dissertation.

[142] I am aware of Ioannidis's recent activities; my citation of his work here should not be interpreted as an endorsement of his positions on… Anything, really.

[143] Unless otherwise noted, the terminology employed in this section is drawn from Clayton (2021) or from the standard statistical lexicon.

$$P[H|D]$$

Our quest for inferential probability is hampered by a significant hurdle: inferential probability cannot be observed, nor can it be measured. There is no way to empirically assess the objective truth value of a hypothesis.[144] As such, we've been forced to construct statistical machinery capable of accounting for observed evidence and converting the result into an inferential probability. The operative component our statistical machine almost always focuses on "the probability of the observed data conditional on a hypothesis", and can be written as:

$$P[D|H]$$

The most commonly-used term for the above quantity is 'likelihood,' but from here on out, I'll refer to it as the 'sampling probability.' Fortunately for us, sampling probability **is observable** and **can be measured**.

One hurdle remains to us: inferential probability ($P[H|D]$) and sampling probability ($P[D|H]$) are not generally equal to one another. In order to use the latter to gain insight about the former, we require some framework for linking the two. Unfortunately, there is no universally-accepted method of doing so. The approaches currently in vogue utilize differing definitions of probability; in the subsections that follow, I'll briefly outline three of the more influential views on the subject.

## Probability as Axiom

Venerable definitions of probability typically make reference to the mathematical properties outlined in Kolmogorov's axioms (Kolmogorov and Bharucha-Reid 2018). The axioms (presented here in simplified form) are:

1. The probability of an event $A$ occurring is a non-negative real number.

$$P[A] \geq 0$$

2. The probability of any event in a set of such events $S$ occurring is 1.

$$P[S] = 1 \qquad \text{(where } \{A...B\} \in S\text{)}$$

$$0 \geq P[A] \geq 1 \quad \text{(as a result of the above)}$$

3. The probabilities of mutually exclusive events add together.

$$\text{if } P[A \text{ and } B] = 0, \text{ then } P[A \text{ or } B] = P[A] + P[B]$$

Kolmogorov's axioms are widely-respected, and no serious definition of probability leaves home without them. Conspicuously absent from the foregoing, however, is any indication of how the concept of probability might interface with the real world. As such, the axiomatic view of probability is indispensable inasmuch as it imbues probability with calculable properties, but doesn't help us figure out what probability *is*, nor does it help us bridge the gap between sampling probability and inferential probability (Clayton 2021) .

## Probability as Logic

The Bayesian paradigm views the probability of a proposition as a *logically-informed degree of belief*. There are two senses in which Bayesian inference owes its moniker to 18th-century Presbyterian minister Thomas Bayes: The first sense concerns the now-ubiquitous Bayes' theorem[145], which governs the use of 'inverse probability' (Bayes 1763; Laplace 1820) and is often rendered as:

$$P[A|B] = \frac{P[B|A]\,P[A]}{P[B]}$$

[144] At least not in the sense evoked by Aristotelian formal logic (Jaynes 2003).
[145] Frequently referred to as a 'law' or 'rule."

In plain English, the above can be interpreted as "the probability of proposition A conditional on proposition B ($P[A|B]$) is equal to the probability of B given A ($P[B|A]$) times the unconditional probability of A ($P[A]$), divided by the unconditional probability of B ($P[B]$)".

Bayes' theorem was, at the time of its articulation, groundbreaking and revelatory (McGrayne 2011), but it was also a natural outgrowth of probability (as later articulated by Kolmogorov). As such, it is implicitly accepted by, utilized with, and part of, every serious treatment of probability – including the Frequentist perspective. In light of the forgoing, it may seem odd that something so fundamental would share a name with a comparatively 'niche' statistical paradigm. This brings us to the second sense in which the Bayesian paradigm is named after the good reverend. Consider the following:

$$\text{Posterior} = \frac{\text{Likelihood} \cdot \text{Prior}}{\text{Marginal}}$$

Employed in an inferential mode, Bayes' Theorem can be used to update a **prior** degree of belief (expressed as a probability) by multiplying it by the **likelihood** of some observed data and dividing the result by the **marginal** probability of the data. This produces a **posterior** probability, which expresses our updated belief in the hypothesis after considering the observed data. Written using the now-familiar structure of Bayes' Theorem, this becomes:

$$P[H|D] = \frac{P[D|H]\,P[H]}{P[D]}$$

All we've done here is swap the symbols around ($A \rightarrow H$, $B \rightarrow D$). In light of the aforementioned meanings of $D$ and $H$ ('Empirically Observed Data' and 'A Hypothesis', respectively), this version of Bayes' theorem tells us that the probability of a hypothesis conditional upon observed data ($P[H|D]$, the 'posterior' or 'inferential probability') can be found by multiplying the probability of the data given the hypothesis ($P[D|H]$, the 'likelihood' or 'sampling probability') with the unconditional probability of the hypothesis ($P[H]$, the 'prior'), and dividing the product by the unconditional probability of the data ($P[D]$, the 'marginal probability of the data'). In essence, Bayes' theorem solves the problem posed at the beginning of this section: it permits us to rigorously and precisely convert sampling probability ($P[D|H]$) into inferential probability ($P[H|D]$).

Though elegant, using Bayes' theorem in its inferential mode necessitates careful consideration of a few hurdles. While the sampling probability ($P[D|H]$) can be known precisely, neither the prior probability ($P[H]$) nor the marginal probability of the data ($P[D]$) are so easily determined. Computational methods have largely obviated the need to know $P[D]$, but $P[H]$ remains a sticking point for many: given that the prior represents an individual's logically-informed state of belief in a proposition before observing data, some have justifiably criticized the Bayesian for introducing 'subjective' or 'arbitrary' information into what 'should' be a purely objective domain (Efron 1986; Gelman 2008).

Bayesian modelling choices are explicit, overt, and transparent to anyone with access to the model's code or mathematical specification (Gigerenzer 2018; McElreath 2020). While true that poorly-set priors may call the validity of a Bayesian model into question, the presence of an ill-tempered prior cannot easily be hidden from discerning audiences. What's more: a unified approach to prior setting was arguably best articulated by Edwin Jaynes in his book *Probability Theory: The Logic of Science* (2003). Therein, Jaynes synthesizes Richard Cox's (1946) view on probability assignments with his own, producing a doctrine that might best be described as "Probability as Logic" (Clayton 2021). The key of the Jaynes-Cox system is that it demands that the probability assigned to a proposition (wherein probabilities act as a convenient scale for degree of belief) is constrained so as to follow a set of logical principles. The end result is that probability is *entirely dependent on the information available*. The result of adhering to Jaynes-Cox is that any two individuals with the same information would arrive at identical conclusions about the prior probability of a proposition.

Despite lingering ambiguities, Bayes' theorem reliably serves as a one-size-fits-all, universally-appropriate approach to statistical inference. It can be used with any class of statistical model, and is as applicable to linear regression as it is to knockout-based neural-network learning algorithms (Clayton 2021; Geron 2019).

## Probability as Frequency

To a Frequentist, probability interfaces with reality in a straightforward way: probability and frequency are synonymous.

The Frequentist paradigm is built upon Bernoulli's (Bernoulli 1713; Bolthausen and Wuthrich 2013) early contributions to probability theory – in particular, his *Law of Large Numbers*, which associates the probability of an event with its 'long-run frequency'. In practical terms, Bernoulli defined the probability of a given event as being *asymptotically equal* to the observed frequency of that event across an arbitrarily large number of 'trials'. This is the definition of probability we most often employ whilst thinking of toy examples such as flipping a coin, drawing from a deck of playing cards, or drawing variously-coloured balls from an urn.

The dividend of Bernoulli's groundbreaking claim was that one could observe the sampling frequency of an event approach the event's already-known inferential probability (Clayton 2021).[146] Bernoulli's reasoning, however, contained a critical flaw: his logic only guaranteed that sampling probabilities would be close to inferential probabilities, not the other way around. At first blush, the importance of this distinction is difficult to grasp; so difficult, in fact, that Bernoulli – one of history's foremost mathematicians – felt comfortable assuming that since sampling probabilities would almost always converge to a known inferential probability, *an unknown inferential probability is highly likely to be close to an observed sampling probability.* Tragically, the latter-most statement in the preceding sentence is, in most cases, untrue.

While Bernoulli merely meant to show that the 'true' probability of an event or proposition could be estimated from its frequency, later statisticians such as Pearson, Neyman, and Fisher (Neyman and Pearson 1933; R. Fisher 1955) took the idea one step further by claiming that an event's probability was *strictly the same as* its frequency. In so doing, the Frequentist paradigm was born. The Frequentist interpretation spread rapidly, as it provided an intuitive interpretation of probability that appeared to work well in most settings and was easily computable (Clayton 2021; McGrayne 2011).

The Frequentist paradigm's streamlining came at a cost: if something did not have a frequency, then the Frequentists could not allow it to have a probability. Exceptions were made for rare, one-off, or past events such as elections,[147] but the paradigm was uncompromising on the subject of hypotheses and propositions: since a proposition did not have a frequency, it could not be described using the language of probability. In essence, the founding Frequentists were claiming that *there was no such thing as inferential probability* ($P[H|D]$), that all propositions ($H$) were either true or false, and that the *only* way to draw inferences about a proposition was by using the sampling probability ($P[D|H]$).

Rather than viewing the elimination of inferential probability as an unfortunate side effect of their take on probability, the Frequentists regarded Bayesian priors and posteriors with vitriolic disdain and saw their elimination as a notable innovation of the Frequentist paradigm (Clayton 2021; McGrayne 2011). To the Frequentists, Bayesian priors were subjective, and thus had no place in the pursuit of objective scientific truth. As we know from the previous subsection on probability as logic, however, the use of prior information is currently the *only* valid means by which a sampling probability can be converted into an inferential probability – Kolmogorov's axioms and the algebra of probability demand it. The Frequentist desire to eliminate priors simply doubles down on Bernoulli's existing mistake. To see why, consider the following:

In the future, a brilliant, eccentric scientist has developed a test capable of determining whether any of the 10,000 individuals aboard a spaceship is human ($n_{\text{human}} = 9{,}999$) or a murderous cyborg sent to infiltrate and destroy humanity from within ($n_{\text{cyborg}} = 1$). When the testing apparatus identifies a cyborg, it raises an alarm. The test is capable of detecting cyborg infiltrators with 100% accuracy – it never misses a cyborg. For humans, the test has a false positive rate of 1%; only 1 out of 100 subjects tested will be incorrectly identified as a cyborg infiltrator.[148] Based on this information, we can generate a list of sampling probabilities, with each taking the familiar from of $P[D|H]$, where $D$ is the test result, and $H$ is the subject's unknown status:

[146] All this assumes a sufficiently large sampling size. In the interest of forthrightness, Bernoulli himself did not use the same terminology employed in this chapter, nor did he have access to any form of notation capable of indicating conditional probability. The latter point is especially salient for two reasons: first, the concept of conditional probability had not yet been articulated; second, lacking conditional probability is a major contributing factor behind Bernoulli's blunder.

[147] Typically, this was accomplished by 'imagining' a population of said rare, one-off, past, or hypothetical events, and then sampling from among them.

[148] If the numerical conceit behind this fabricated example seems familiar to you, that's because it probably is; though of many protean forms, it appears in statistical textbooks (McElreath 2020), book chapters dedicated to the subject (Pennycook and Thompson 2016), and countless other works on probability (Clayton 2021; McGrayne 2011). Here, I merely continue the tradition of swapping out the framing narrative for something slightly different.

$$
\begin{aligned}
P[\text{ Alarm} \mid \text{Cyborg }] &= 1 && \text{(True Positive)} \\
P[\text{ All Clear} \mid \text{Cyborg }] &= 0 && \text{(False Negative)} \\
P[\text{ Alarm} \mid \text{Human }] &= 0.01 && \text{(False Positive)} \\
P[\text{ All Clear} \mid \text{Human }] &= 0.99 && \text{(True Negative)}
\end{aligned}
$$

Going by sampling probability, the test seems remarkably reliable (which is a relief, as subjects who trip an alarm are summarily thrown out the ship's airlock). According to a strict interpretation of their inferential doctrine, the Frequentist paradigm would be content with interpreting the result of the cyborg scan as a test statistic and proceed to the inferential step: Since being human is, in this case, a common and uninteresting result, it would serve as the null hypothesis. We run the test and observe an alarm, which leads us to consider the sampling probability of observing that data under the assumption that the null is true. Since $P[\text{ Alarm} \mid \text{Human }]$ is very low (0.01), we can reject the null hypothesis at the 1% significance level – a threshold sufficient to justify publication in many field-leading medical journals (Clayton 2021).

Now let's see how our conclusions might have differed had we used the Bayesian approach, instead. Since we've observed an alarm, we're interested in calculating the inferential probability $P[\text{ Cyborg} \mid \text{Alarm }]$. Using Bayes' theorem will require us to use the sampling probabilities (which we already know), the prior probability $P[\text{ Cyborg }]$ (which we can trivially calculate using what we know about the ship's population), and the marginal probability of the data $P[\text{ Alarm }]$, which is 0.0101.[149] Plugging this information into Bayes' theorem gives us:

$$
P[\text{ Cyborg} \mid \text{Alarm }] = \frac{P[\text{ Alarm} \mid \text{Cyborg }]\, P[\text{ Cyborg }]}{P[\text{ Alarm }]} = \frac{1 \cdot \left(\dfrac{1}{10000}\right)}{0.0101} \approx 0.01
$$

We can also calculate the corollary inferential probability:

$$
P[\text{ Human} \mid \text{Alarm }] = \frac{P[\text{ Alarm} \mid \text{Human }]\, P[\text{ Human }]}{P[\text{ Alarm }]} = \frac{0.01 \cdot \left(\dfrac{9999}{10000}\right)}{0.0101} \approx 0.99
$$

Taking this information and re-creating the table above, only this time with inferential probabilities of the form $P[H|D]$, we get:

$$
\begin{aligned}
P[\text{ Cyborg} \mid \text{Alarm }] &\approx 0.01 && \text{(True Positive)} \\
P[\text{ Cyborg} \mid \text{All Clear }] &= 0 && \text{(False Negative)} \\
P[\text{ Human} \mid \text{Alarm }] &\approx 0.99 && \text{(False Positive)} \\
P[\text{ Human} \mid \text{All Clear }] &= 1 && \text{(True Negative)}
\end{aligned}
$$

The latter set of statements paints a dramatically different picture from the former: where our set of sampling probabilities led us to conclude that an alarm result was a reliable indication of a cyborg, the inferential probabilities show that an alarm result means that the subject is *almost certain to be human.* This apparent contradiction arises from the fact that the crew of the spaceship being tested is overwhelming human ($n_{\text{human}} = 9,999$), containing but a single cyborg ($n_{\text{cyborg}} = 1$). This means that even with a very low false positive rate, the vast majority of the alarms will be false alarms ($\approx 99\%$). Based on this example, we can conclude that Bernoulli's claim (that $P[D|H] \approx P[H|D]$) is not reliably true; this poses a constitutive threat to the Frequentist paradigm's foundational beliefs.

Even though the above example is contrived, it is modelled after countless similar real-world examples of Frequentist inference producing disastrous results, collectively known as "Base Rate Neglect" (Clayton 2021; Pennycook and

[149] I've skipped over the details of calculating the marginal in the body text. Marginal probabilities are often difficult to calculate, but in discrete problems such as the one posed here, we can use the following: $P[D] = \sum_H P[D|H]P[H]$. The forgoing only works when all possible hypotheses are known and accounted for, which isn't often the case. Nevertheless, it leads to the observation that Bayes' theorem requires all alternative hypotheses to be mathematically considered and accounted for; an elegant facet of the theorem that emerges naturally from first principles.

Thompson 2016).[150] The related and well-known "Prosecutor's Fallacy", for example, stems in part from a criminal case where an almost-certainly innocent woman was convicted of murdering her two infant children because the 'null' hypothesis (namely, that both her infants succumbed to SIDS) was exceedingly unlikely and could be rejected: the statistical argument was deeply flawed in part because it failed to consider other alternative hypotheses (i.e. that the infants perished of undiagnosed bacterial infections), and it neglected to account for the fact that the one alternative hypothesis it did consider – double infanticide – was also exceedingly unlikely ($P[\text{ Double Infanticide }]$ was very low) and would have been rejected had it been chosen as the null (Clayton 2021).

The only settings in which sampling probabilities can be relied upon as valid estimators of inferential probabilities are those with **complete sets of alternative hypotheses** and **no important prior information**. This leads us to the truly pernicious aspect of the Frequentist delusion: using sampling probability as a substitute for inferential probability works perfectly well in almost all of the toy examples (i.e. drawing cards, rolling dice, removing balls from an urn, etc.) we humans habitually use to validate our understanding of probability. *These intuitions fail when applied to real-world problems.* The bottom line is this: it is impossible to draw valid inferences from data in settings influenced by important prior information *unless one takes that prior information into account*.[151] By arbitrarily and inconsistently limiting the range of prior information allowed, Frequentism needlessly renders itself unable to engage with most inferential domains in a principled manner.

We'd be better served by adopting a better paradigm. Fortunately, such a paradigm exists: Bayesian inference is capable of simultaneously and directly comparing multiple hypotheses without any need for a reference to a null. It is also capable of incorporating prior information about hypotheses and producing full descriptions of the resulting inferential probabilities.[152] This is not to say that Bayesian inference cannot be manipulated or exploited; this *is* to say that Bayesian inference makes the act of statistical mendacity – intentional or otherwise – much more vulnerable to scrutiny. Researchers' assumptions and priors are transparent, and their implications are testable by anyone with access to the same data (Gigerenzer 2018; Szucs and Ioannidis 2017).

There is much, much more that could and should be written here on the topic of statistical inference. Unfortunately, a full treatment is beyond the scope of this chapter.[153] For the time being, suffice it to say that the Bayesian paradigm has come to be recognized as the most viable, tractable, and logically consistent approach to statistical inference currently available (Clayton 2021). In light of this, one might reasonably conclude that Bayesian inference has come to dominate in the quantitative practices of most disciplines across academia – this is not, unfortunately, the case. In the next section, I'll briefly summarize the scant literature on the subject to show that Bayesian inference is represented in but a miniscule fraction of contemporary quantitative social science publications.

[150] Of course, Frequentism is aware of base rate neglect, and has developed techniques capable of adjusting for it in inferential settings. Their approach is effective in limited cases with well-established base rates, but wholly unsatisfactory as a general-purpose solution to the problem. Since Frequentist adjustments for base rate neglect are only permitted in settings where the base rate is known to a high degree of accuracy, how do we account for cases where the prior information is known to be important, but is not known to a high degree of accuracy? If we know that the number of cyborgs aboard ship is low, but we're not entirely certain *how* low, would the Frequentist paradigm permit us to use that information? What's more, how do we adjust for base rate neglect in cases where there isn't a referent population, such as one-off or rare events? Adjusting for Base-Rate Neglect represents a clumsy band-aid solution to a fundamental flaw in Frequentism that cannot be adequately addressed without abandoning the foundational myth that probability is synonymous with frequency. The general solution to Base-Rate Neglect (and all of its innumerable variations) involves explicitly considering prior information, but the very act of doing so in all but the most circumscribed of settings constitutes a serious violation of the tenets of the Frequentist paradigm and forces users into a Bayesian mode. In sum: the only way to fix Frequentism is to be Bayesian.

[151] Unconvinced readers may be tempted to retort by claiming that we can still use Frequentism in research settings because it's usually safe to assume that one knows nothing about the problem one is studying, and can thus proceed as if there is no important prior information. This is almost always an unrealistic assumption: the very fact that one is studying something implies the presence of prior information sufficiently strong to motivate the act of research – said prior information will almost always play a significant role in inferential settings, whether the researcher accounts for it or not. Researchers ignore prior information at their own peril.

[152] Bayesian inferential probability – or, equivalently, posterior probability – is always a probability distribution. As such, Bayes provides a much more complete answer to an inferential question than summary-bound counterparts (McElreath 2020).

[153] A panoply of outstanding Bayesian textbooks have emerged of late, and intrigued readers are highly encouraged to consult them for a more complete treatment of the subject. For an intuitive, conceptual approach to the problems with Frequentism and the viability of the Bayesian alternative, readers should consult Clayton (2021). Kruschke (2014), Davidson-Pilon (2015), Downey (2021), and McLevey (2021) all contain practical introductions to various facets of Bayesian data analysis. At the time of this dissertation's publication, I view Gelman et. al. (2020) and McElreath (2020) as the leading works on advanced Bayesian inference: the former requires a working knowledge of mathematical probability, whereas the latter is intended for readers with statistical programming experience.

## The State of Bayes in the Social Sciences

Bayesian inference is a good idea, but we can't count on good ideas to spread by themselves – especially not in light of institutional inertia and barriers to adoption (Rogers 1976). If, as of the publication of this dissertation, there was evidence that the pace at which the social sciences are adopting the Bayesian paradigm would position it at parity with Frequentism in the foreseeable future, I would be content to allow the diffusion of this good idea to proceed without intervention. If, on the other hand, no such evidence is forthcoming, I would interpret that as a call to action. The importance of any such disparity between the social sciences and other leading fields cannot be overstated, as it would suggest that – in the absence of meaningful efforts for a dramatic overhaul – the social sciences are at serious risk of missing out on the benefits of methodological advances from other fields, as well as being stymied in the conditions that made the replication crisis a near inevitability (Clayton 2021).

Unfortunately, the social sciences have a lot of catching up to do – and appear to be going about this task with risible sluggishness (Lynch and Bartlett 2019). Across most social scientific disciplines, the story is the same: researchers continue to overwhelmingly employ Frequentist Null Hypothesis Significance Testing as their default approach to statistical inference. With minor variations, this is true of fields such as Archaeology (Otárola-Castillo and Torquato 2018), Anthropology (Konigsberg and Frankenberg 2013), Game Studies (Westera 2021), Organizational Science (Kruschke, Aguinis, and Joo 2012), Political Science,[154] Psychology (Wagenmakers et al. 2018),[155] and Sociology (Lynch and Bartlett 2019). In each case, Bayesian inference was present in less than 1% of the fields' quantitative publications (as of the year 2017).

Konigsberg and Frankenberg contend that Biological Anthropologists think, postulate, and pose questions using decidedly Bayesian logics and interpretations, but almost exclusively rely on Frequentist NHST to draw inferences from their data. They attribute the ongoing prevalence of Frequentist methods in "past training and stasis in the field" (Konigsberg and Frankenberg 2013, 168). Otarola-Castillo and Torquato contend that quantitative Archaeologists often find the underlying logic of NHST "arbitrary and confusing" (Otárola-Castillo and Torquato 2018, 436), which leads to frequent misuse and misinterpretation of Frequentist tools. Wagenmakers et al. acknowledges that while many Psychologists are aware of Bayesian inference and have a desire to be competent practitioners thereof, they remain daunted by the prospect of learning a new set of statistical techniques (Wagenmakers et al. 2018). Taken together, the literature paints a picture of social science researchers who think, theorize, design experiments, and conduct research using a quasi-Bayesian understanding of probability and statistics. These same scientists, however, cannot see their way clear to extending their use of Bayes through their models and the interpretation thereof.

One lone bright spot for the promise of a Bayesian renaissance in the social sciences: the discipline of Economics and Econometrics (Lynch and Bartlett 2019). Although the rate of Bayes-focused publications still lags far behind that of Frequentist-focused publications, Economics alone managed to break the 1% barrier in the aughts: in 2008, about 15 in every 1000 Economics papers featured some quantitative Bayesian analysis (Lynch and Bartlett 2019, 56).

The conclusion to be drawn from this review is, in short, that the social sciences are not self-correcting their use of Frequentist inference at anywhere near a sufficiently rapid pace. Without meaningful and widespread intervention, the social sciences will continue on as statistical somnambulists – they'll wander towards "significance" by going through the motions of quantitative inference without the benefits of memory, reflexivity, or foresight. In the upcoming section, I'll detail a framework that may help explain why the reliance on NHST in the social sciences continues unabated – a framework I expand upon in order to clear a path out of the inferential quagmire.

## The Carrot and the Stick

To recap the chapter thus far: the basis of most statistical inference in the social sciences is broken; Frequentism can't be fixed, only replaced; Bayesian inference represents a viable alternative. With the "Great Debate" (Clayton 2021)

[154] In the case of political science, the picture is slightly muddier – while Bayesian inference appears in less than 0.2% of all published quantitative Political Science works (Lynch and Bartlett 2019, 56), Political Scientists Gill and Heuberger have nevertheless concluded that "Bayesian methods and Bayesian inference are now ubiquitous in the social sciences, including political science and international relations" (Gill and Heuberger 2019, 961). On the face of it, Gill and Heuberger's claims appear to contradict those of Lynch and Bartlett – by considering the two together, we can perhaps conclude that Bayesian inference is now well-known in Political Science, but the studies that employ it represent a much smaller proportion of the field than those using the still-dominant Frequentist paradigm.

[155] Which, thus far, has borne the overwhelming brunt of the heat stemming from the ongoing replication crisis (Clayton 2021).

between Frequentist and Bayesian largely decided in the latter's favour, one would expect social scientists to be flooding to Bayesian inference en masse. They aren't (Lynch and Bartlett 2019). Why haven't Bayesians' arguments against Frequentism precipitated the desired sea change?

Evidence specifically regarding the social sciences is, at present, scant-to-nonexistent, but knowledge of a proxy may be instructive here: recent surveys of medical researchers have identified a lack of knowledge as the single largest barrier to their use of Bayes, closely followed by a dearth of practical tools, worked examples, and regulatory guidelines (Natanegara et al. 2014; Faya et al. 2021; Clark et al. 2022). Notably absent from the survey responses are concerns about the Bayesian paradigm's fundamental suitability: medical researchers generally want to move towards Bayes, but feel hampered by a lack of both knowledge and practical means to acquire said knowledge.

In this section, I argue that existing modes of argumentation have focused exclusively on tipping the scales in favour of Bayes whilst ignoring the knowledge barriers preventing practicing social scientists from adopting Bayesian inference.

## Gelman's Typology

In "Objections to Bayesian Statistics," Andrew Gelman (2008) summarizes Bayesian contributions to their ongoing debate with the Frequentist camp as broadly breaking down into one of three modes.[156] The typology consists of 'defense', 'affirmation', and 'attack'. In the subsections that follow, I explore each in slightly more detail.

## Affirmation

When an argument seeks to establish how Bayesian inference can outperform Frequentism, it belongs to the 'affirmation' category. Owing to the fact that the Bayesian advantages over the Frequentist paradigm are myriad, affirmation arguments are plentiful.

By way of an example, consider the supposedly iterative process of science. Whereas Frequentist inference facilitates the growth of knowledge through literature review and meta-analysis (which only functions under exacting stipulations), Bayesian inference automatically facilitates the progressive refinement of statistical estimation on a given effect (McElreath 2020). Since the posterior from one model may be used as a prior for another, Bayesian inference serves as a formal, principled system for accumulating knowledge and producing successively more confident predictions based on observed data (Lynch and Bartlett 2019). Bayesian models are even capable of doing so in cases where the experimental setup or set of controls differs from model to model (Clayton 2021). Bayesians have also argued that their paradigm offers advantages with respect to almost every aspect of the inferential workflow (Lynch and Bartlett 2019), including:

- Model selection (Peixoto 2021)
- Parameter estimation (Wagenmakers et al. 2018)
- Missing data (McElreath 2020)
- Hierarchical modelling (Britten et al. 2021)
- Ease of implementation (Clayton 2021)

Each of these points add to the growing body of evidence that the Bayesian paradigm not only offers the more principled approach to statistical inference, but it also offers power and flexibility that cannot be matched by the Frequentist alternative. In this way, affirmation arguments incentivize switching by highlighting the benefits of utilizing Bayesian inference. Affirmation arguments do not, however, address the hurdles faced by those looking to adopt Bayesian inference, and thus have not catalyzed the necessary change.

## Defense

If affirmation arguments seek to highlight how Bayesian inference can outperform Frequentist inference in specific ways, 'Defense' arguments justify the use of Bayes using the opposite logic: since Bayesian inferential tools can be configured so as to very closely resemble their Frequentist counterparts, Bayesian inference is a valid approach.

Here again, defensive arguments are myriad. Bayesian practitioners have proposed the creation of Bayesian substitutes for commonly-used Frequentist tests and statistics including $p$-values (Meng 1994) and confidence intervals (Severini

[156] In fairness to Gelman, I should point out that he adopts a faintly mocking tone towards each of the members of his typology; it seems to me that he, too, is not convinced of their utility.

1993). The intention behind each of these is to provide Frequentists with familiar metrics that fit within a Bayesian framework. This same ethos drives the marquee defensive argument that demonstrates how Bayesian models can, if properly configured, produce numerical results that almost perfectly align with what a Frequentist model would have produced (Lynch and Bartlett 2019). To do so involves the use of a 'flat' or 'uninformative' prior distribution on the model's parameter space, which allows the data to 'speak for itself' as it would in a comparable Frequentist model, but with the Bayesian advantage of directly interpretable posterior parameter distributions (McElreath 2020).

Where affirmation uses the benefits of Bayes as its carrot, defensive arguments rely on familiarity to lure social scientists over to the Bayesian camp. Where affirmation is merely ineffective, defensive arguments are liable to be actively counterproductive – why should a Frequentist pour effort into switching when the Bayesian paradigm offers little more than business as usual?

### Attack

According to Gelman, an 'attack' argument is one that focuses on pointing out the inadequacies of the Frequentist paradigm in the interests of convincing Frequentists to abandon their existing practices and embrace Bayesian inference. By demonstrating how Frequentism fails us, attack arguments seek to highlight the costs incurred by a Frequentist's decision to stay put. This chapter has already covered two of the most salient reasons why Frequentist inference is unequal to the challenge posed by the contemporary social sciences (both of which, I'd argue, take the form of attack arguments), so this section will focus on why such argumentation is not, by itself, sufficient to convince social scientists to make the switch.

While attack arguments are effective at convincing some Frequentist social scientists (including yours truly), they are not effective in aggregate. According to Kuhn (1970), scientists are unlikely to abandon an existing paradigm when confronted with anomalies therein. What's more: by assuming that attacking Frequentism will eventually result in a Bayesian takeover, attack argumentation is committing a similar mis-step to that of the titular counsel in 'The Prosecutor's Fallacy'.[157] Trying to support one position by pointing out flaws in the other only works in cases where there are only two positions. By attacking Frequentism, Bayesians may end up merely pushing 'Status Quo' Frequentists towards some sort of 'Reformer' camp, who seek to shore up Frequentism's shortcomings using a myriad of tactical fixes.[158]

Finally, and most saliently, attack-style arguments aren't effective at reducing the costs associated with making the transition from Frequentism to Bayes.

### It's Time To Try Something Different

The argumentative arsenals maintained by each mode in Gelman's typology are impressively large, but none has yet been sufficient to the task of inducing lasting change in the social sciences' choice of preferred statistical paradigm. Each mode only serves to convince the already-convinced, and does so whilst ignoring the costs associated with switching from Frequentism to Bayes. In the section that follows, I detail a mode of argumentation that addresses this elision.

## Ameliorative Arguments

In this section, I argue that a neglected mode of argumentation – which I call the 'Ameliorative Argument' – has the potential to transcend the unproductive deadlock between the Bayesian and Frequentist camps. Instead of merely attempting to bludgeon Frequentism into submission by repeatedly evoking the paradigm's shortcomings, Bayesians should attempt to hear and accept the truth behind concerns raised by those hesitant or unable to make the switch.

In this setting, an 'Ameliorative Argument' is one which addresses concerns raised by those skeptical about Bayesian inference or the adoption thereof. Rather than merely dismissing the concern or engaging in 'whataboutism', a properly-formulated ameliorative argument acknowledges the validity of one or more practical argument(s) against adopting Bayesian inference as a standard practice in the social sciences. It is also designed to show how the concerns raised by the practical argument could be addressed. In cases where a solution already exists, the purpose of an ameliorative argument is to demonstrate how extant solutions interface with established statistical praxis. In cases where

[157] The unhappy story of which can be found in the 'Sleepwalking to Significance' section.
[158] See the 'Ritual and Illusion' section for more on efforts in this regard.

no extant solution exists, an ameliorative argument might posit a set of steps that could be taken to arrive at a solution – or, if possible, simply take the necessary steps and publicize the result.

My primary objective in evoking the 'ameliorative argument', then, is to demonstrate that good faith arguments about practical concerns can be productive and instructive. To that end, the remainder of this section will be devoted to describing how Bayesians might use 'Ameliorative Arguments' to help convince social scientists to embrace Bayesian inference. I'll start by pointing out several Frequentist concerns that Bayesians have already taken steps to ameliorate.

## Concern: Adrift without Convention

Some have argued that Frequentist conventions aren't particularly sensible (Gigerenzer 2004; Szucs and Ioannidis 2017), but the presence of these well-established conventions nevertheless permit simple interpretation of Frequentist findings – 70 years of statistical dominance has all but ensured it. Frequentists, then, would be justified in pointing out that Bayesian inference enjoys no such accretion of interpretive convention. Thus far, the standard Bayesian response to this Frequentist misgiving has been glib dismissal. "A slavish adherence to star-gazing and $R^2$ values led us into the replication crisis," one might say. "Why would we want to build a new road back to the same unhappy terminus?"

Despite its allure, such a response is neither helpful nor productive. It does nothing, for example, to help the scores of medical researchers who have identified strong guidelines as a precondition for their adoption of Bayesian inference (Clark et al. 2022; Faya et al. 2021; Natanegara et al. 2014). An ameliorative response would involve drafting a working set of guidelines and circulating it for feedback and iteration. Fortunately, John Kruschke has taken up the mantle of this challenge:

## Assimilated by the BARG

In "Bayesian Analysis Reporting Guidelines" (BARG for short), Kruschke (2021) posits a comprehensive set of guidelines for explicating Bayesian analytical efforts. The guidelines act as a step-by-step checklist which, if followed, ensures that sufficient information about the analysis is available for outside observers to effectively grasp, critique, and extend on any model so reported. Throughout, transparency and reproducibility are foregrounded. Notably, the BARG provides a set of guidelines for social statisticians to follow, but it does not absolve practitioners of the burden of having to carefully think through each and every analysis they conduct. This is both an intended and desirable property of the BARG. Finally, the BARG is not final: it represents a work in progress. It was inspired by previous efforts to establish a set of reporting guidelines – such as ROBUST (Sung et al. 2005) –, and it is designed to inspire further refinements. This is an ameliorative argument in motion.

## Concern: Overly-Convenient Prior Distributions

The workhorse model of the Frequentist paradigm in the social sciences – ordinary least squares regression – is equivalent to Bayesian linear model with a flat 'improper' distribution over all of the priors (which is considered bad practice in most Bayesian circles). "While Frequentists are content to simply assume flat priors" a Bayesian firebrand might claim, "we Bayesians make explicit choices about the shape of our priors, taking pains to ensure that they are the most logically consistent given the information available!" I can imagine a cunning Frequentist then retorting that "you certainly do, my Bayesian friend: so curious, then, that your careful consideration of each and every model leads you inexorably back to the familiar territory of the Normal, Poisson, Beta, and Exponential distributions!"

The hypothetical Frequentist makes a good point: with an infinite array of probability distributions at our disposal, how can Bayesians justify our reliance on the same small rogue's gallery? It bears mentioning that for Frequentists, this is a solved problem – for a given set of assumptions, the Frequentist paradigm has rigorously defined what it considers to be the 'Best Unbiased Linear Estimator' (Clayton 2021). In that light, it's understandable that the majority of the respondents in one survey of medical researchers listed 'prior construction/elicitation' as their most pressing concern (Faya et al. 2021).

The Bayesian paradigm has already established a similarly rigorous framework justifying the priors it habitually calls upon, but said framework is buried in Edwin Jaynes' book under several chapters of opaque prose and advanced mathematics (Jaynes 2003; Clayton 2021). A potential convert would be justified in viewing this as a troubling barrier to their adoption of Bayesian inference. Fortunately, some pioneering authors have already begun to break Jaynes' work into a more digestible format:

## Maximum Entropy Made Easy

In the 10th chapter of Statistical Rethinking (2020), McElreath provides an intuitive explanation of how priors and likelihoods often trend towards familiar distributions not due to their convenience, but rather because they represent *maximally entropic states of knowledge*. In other words, the distributions most often called upon can be justified as standard choices because they encode as little information as possible without ignoring the information we have.

McElreath articulates a conservative approach to Bayesian inference: models make assumptions about that which we have knowledge of, but do not encode information we don't yet have. In most settings, however, we don't know much, and the frequent use of maximally-entropic distributions reflects this state of naivete. McElreath's guide is also approachable, straightforward, and doesn't require a graduate degree in math to comprehend. Those in the midst of a transition to the world of Bayesian inference are well served by the availability of this resource.

## Concern: Bayesian Models are Unintelligibly Complex

A fully-specified Bayesian model can be terrifying to behold. Even in the least taxing cases – such as a simple linear model – Bayesian inference demands that a great many proverbial knobs and levers be attended to (Blei 2014; McLevey 2021). This isn't a problem easily solved. One of the greatest strengths of Bayesian inference is that every Bayesian statistical model is essentially 'bespoke': each model is purpose-built by the researcher to assess the distribution of one or more latent variables (or, often equivalently, model parameters). The downside of this is that researchers must necessarily confront the daunting complexity of a Bayesian model for each inferential task they wish to tackle. Opportunities to select and employ a ready-made model "off the shelf" are nearly ubiquitous in Frequentist circles, but only scantly encountered in the Bayesian realm (Blei 2014). To a Frequentist struggling to overcome barriers separating the two paradigms, this is hardly a prospect worth relishing.

Here again, a belligerent Bayesian might reply that the Frequentist models are no less complex than their Bayesian counterparts. The only difference is that Frequentist modelling practices tend to bury the complexity under layers of *unstated* assumption and reference to convention (Clayton 2021). The Bayesian models, by way of contrast, require practitioners to explicitly reckon with every facet of the modelling process. While this may be a valid position in the strictest sense, it isn't a helpful one: it does nothing to address the concerns a transitioning Frequentist might have about the new world of complexity they're suddenly expected to be conversant in.

Of the problems covered in this section, the complexity of Bayesian modelling may be the most intractable. We can't 'solve' it, as to do so would obviate many of the crucial advantages motivating the adoption of Bayesian inference in the first place. As such, the best we can do is to help facilitate intuitive comprehension of Bayesian models. To that end, I present:

## Kruschke's Diagrams

For the second time in this section, John Kruschke's contributions deserve recognition. In his book *Doing Bayesian Data Analysis*, Kruschke (2014) develops and deploys a novel form of visualization designed to explicate how the components of a Bayesian model fit together. The diagrams take the form of Directed Acyclic Graphs (DAGs) that 'flow' from top (prior) to bottom (likelihood). In the case of the example below, the priors feed into the linear model's terms, which – in turn – defines the distribution of the 'location' parameter in the likelihood function at the bottom (See figure 37 on page 75).

Although the mathematical notation used therein may seem archaic, readers may be relieved to learn that this example diagram describes a relatively simple hierarchical linear model. In a nutshell, the diagram shows how a Bayesian linear model works "forwards," in a generative mode.[159] For a given set of observations of an independent variable $x_{j|i}$ grouped into a number of clusters indexed by $i$, this linear model describes how we arrive at an estimate of $y_{j|i}$, which can be read as "the distribution of the outcome variable $y$ for observation $j$ in cluster $i$."

Starting at the bottom, I'll walk through each of the diagram's layers. We've already covered what the $y_{j|i}$ at the bottom means. The wavy arrow pointing into it informs us that $y_{j|i}$ is distributed according to the distribution at the tail of the arrow: in this case, it's a normal distribution with location $\omega_{j|i}$ and scale $\lambda$.[160] The fact that $\omega$ is subscripted by $j|i$ indicates

[159] In an inferential mode – which is the one that most of us are more familiar with – the model works "backwards", using the likelihood of the outcome variable conditional on the model to update the prior distributions.

[160] It would be just as accurate to say 'mean' $\omega_{j|i}$ and 'standard deviation' $\lambda$ – the language used in the main text reflects recent efforts to move away from terminology which necessarily implies a Frequentist interpretation.

that each observation in the dataset will receive its own distribution over the outcome variable $y$, but that only the location of the distribution will differ: the lack of subscripting on $\lambda$ implies that each distribution of $y$ will share a single scale.

If we follow the wavy arrow back from $\lambda$, we arrive at a gamma distribution with parameters $K$ and $I$ – this is the model's prior distribution over the scale parameter $\lambda$.[161] If we follow the straight arrow back from $\omega_{j|i}$, we arrive at an equation describing how we can calculate the scale parameter of $y$'s distribution for a given observation. Here again, the symbols may appear esoteric, but this equation's essential form should be familiar to anyone with a background in regression analysis: it is the 'linear' portion of this model. It states that for a given observation $j$ in cluster $i$, the value of $\omega$ is equal to the intercept $\phi$ plus the product of the slope $\xi$ and the value of the independent variable $x$. In a single-level model, we'd then proceed to place prior distributions on the intercept $\phi$ and the slope $\xi$, and our model would be complete.

Since this is a hierarchical model, we require distinct pairs of priors for $\phi$ and $\xi$ – one for each cluster $i$. In the diagram, the notation differs between the priors placed on $\phi$ and those placed on $\xi$, but I'll cover them at the same time, since both are identical in every other regard. The priors for our linear model components are all normal, with locations determined by $\kappa$ and $\zeta$, respectively, and with scales determined by $\delta$ and $\gamma$, respectively. In the context of a hierarchical model, the Bayesian paradigm views each of the linear model's components' distributions' parameters as being distinct, but also as being drawn from an underlying ur-distribution – these distributions-of-distributions appear along the top row of the diagram, and are known as 'hyperpriors.' They are, in essence, the second-order priors that determine how our model's first-order priors are generated.[162] The location hyperpriors are normal distributions with locations $M$ and scale $H$, whereas the scale hyperpriors are gamma distributions with shapes $K$ and scales $I$.

All of the information in a Kruschke diagram is available in both the model's code and the model's mathematical specification. As such, veteran Bayesian statisticians might view Kruschke diagrams as superfluous. To those of us who still have to look up the difference between the Gamma and Beta distributions (present company included), one of Kruschke's diagrams can do a world of good. As a result, I argue that Kruschke has gifted us with a visual lexicon that should be adopted as a standard component of all quantitative modelling in the social sciences. In much the same way as we currently expect quantitative publications in the social sciences to include tables reporting correlations between modelling variables,[163] expecting the inclusion of Kruschke diagrams would help enshrine accessibility and transparency as core values in our work.

The Kruschke diagram also holds vast potential as a pedagogical tool for both instructors and autodidacts alike. Even when being developed in a context where it is not intended for public consumption,[164] the Kruschke diagram provides intuitive insight into the inner workings of a Bayesian model. Unfortunately, this exposes a mismatch between the Kruschke diagram's utility and its ease of application: they are generally hand-made by someone who understands their referent model inside and out. Despite the help that a Kruschke diagram can render, the hand-made solution is of little assistance to someone struggling to understand the implications of their *own* model.

To fully unlock the benefits of the Kruschke diagram, Bayesians must ensure that they are easily accessible, even to those who may not be deeply familiar with Bayesian model specification. As such, there's clearly room for more ameliorative work here: In the penultimate section of this chapter, I take this difficulty as a starting point for presenting my contribution to the Bayesian ameliorative project.

## Introducing `pyKrusch`

In this section, I detail a software package for visualizing the structure and dependencies of Bayesian models. Herein, I posit that the package – called `pyKrusch` – is an ameliorative argument reified as an easy-to-use, accessible, open-source piece of software. It is intended to help sociologists approach, comprehend, and utilize Bayesian models.

[161] To whom it may concern: one way of parameterizing the gamma distribution – and the one which the authors, I believe, are using – involves using the 'shape' ($k$) and 'scale' ($\theta$) parameters.

[162] One can go further out, too: there's nothing stopping you from putting priors on the hyperpriors, which would create hyper-hyperpriors, I suppose. In most cases, the benefits of adding additional layers in a hierarchical model are outweighed by the costs associated with increased model complexity… But not always!

[163] A practice that, in the author's opinion, is in dire need of re-evaluation.

[164] Such a situation might arise when someone is working through a Bayesian modelling textbook, learning about Bayes in a classroom, or attempting to recreate the machine learning models that power an online recommendation system, for example.

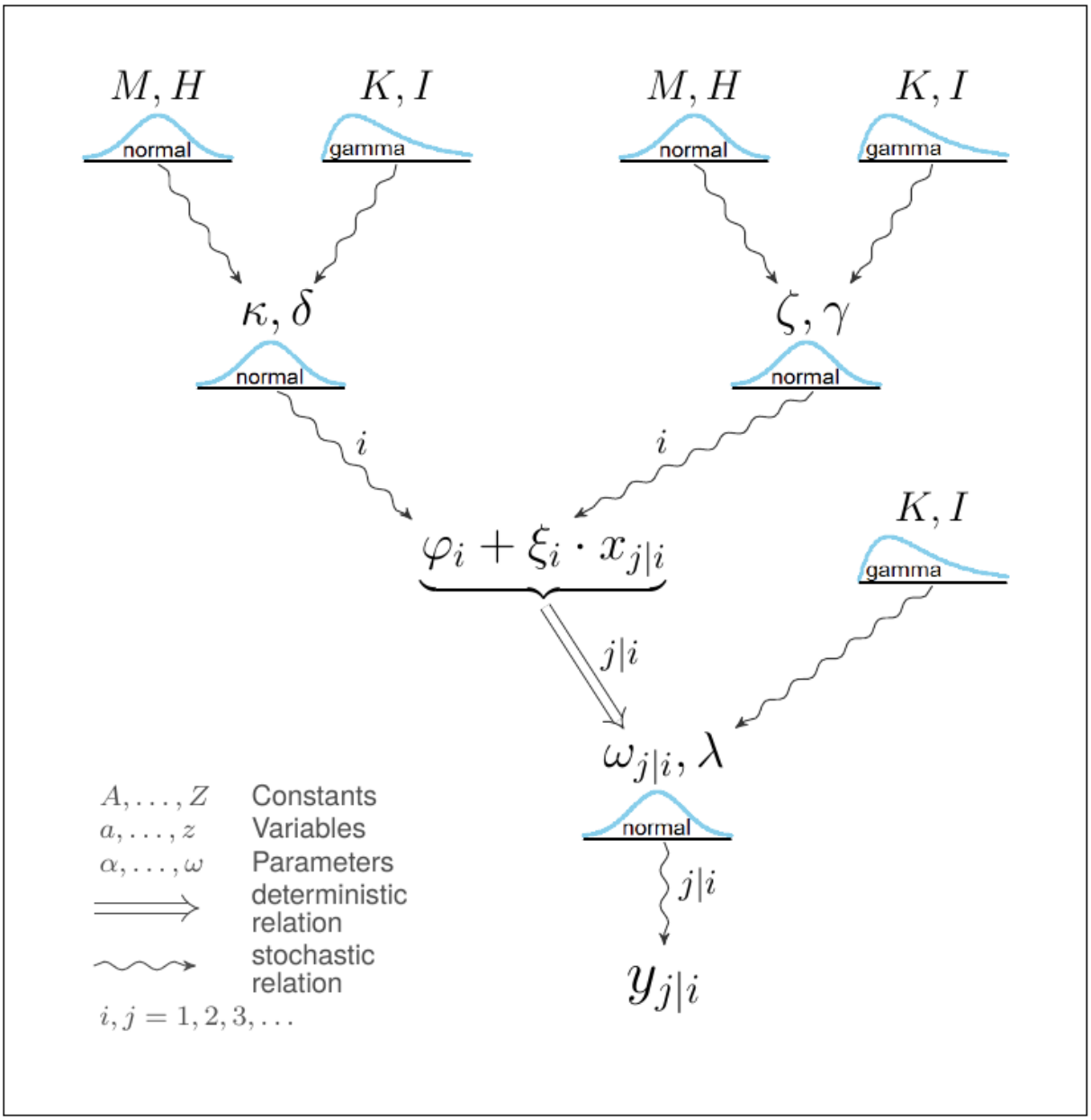


Figure 37: Example of a Kruschke Diagram, from Schneider and Adams (2013)

## pyKrusch: An Overview

The pyKrusch package is a Bayesian model visualization tool for Python 3.8+. It is, at present, designed to work with PyMC (Salvatier, Wiecki, and Fonnesbeck 2016) and arviZ (Kumar et al. 2019), two widely-used Python packages for Bayesian model specification and visualization. pyKrusch's mission statement: use the logics and aesthetics of the Kruschke diagram to enhance PyMC's built-in model visualization suite. As in most facets of science, an ounce of demonstration is worth pounds of explanatory prose (H. M. Collins 1992), so I will open with an example. Consider the following model:

$$
\begin{aligned}
\alpha &\sim \text{Normal}(178, 5) && [\alpha \text{ prior}] \\
\beta &\sim \text{Normal}(20, 2) && [\beta \text{ prior}] \\
\sigma &\sim \text{Gamma}(\alpha = 2, \beta = 3) && [\sigma \text{ prior}] \\
\mu_i &= \alpha + \beta x_i && [\text{linear model}] \\
y_i &\sim \text{Normal}(\mu, \sigma) && [\text{likelihood}]
\end{aligned}
$$

The above – rendered as what I will refer to as 'McElreath Specification' (McElreath 2020) from here on out – is a mathematical description of a Bayesian linear model. The top three lines of the model specify the prior distributions for $\alpha$ (the intercept), $\beta$ (the slope), and $\sigma$ (the standard deviation). Together with the $x_i$ (a hypothetical independent variable), $\alpha$ and $\beta$ feed into the linear model, specified on the fourth line, to produce $\mu_i$. It's worth noting that the linear model line uses an equals sign instead of a tilde – this is because the value of $\mu_i$ depends deterministically on a linear combination of $\alpha$, $\beta$, and $x_i$. The value for $\mu_i$ is then fed into the distribution on the last line (the likelihood) as its location (or mean). The $\sigma$ prior governs the likelihood's scale (or standard deviation). Considered together, the model describes the conditional probability (or likelihood) of observing the values of $y_i$, which is then used to update the prior distributions, producing posterior distributions for each. Here's the above model translated into PyMC code:

```
import pymc as pm

x = [0, 2, 4, 6] # Data Placeholder
y = [1, 2, 3, 4] # Data Placeholder

with pm.Model() as model:

    # Priors
    α = pm.Normal("α", 1, 2)
    β = pm.Normal("β", -1, 2)
    σ = pm.Exponential("σ", 3)

    # Linear Model
    μ = pm.Deterministic("μ", α + β*x)

    # Likelihood
    y_ = pm.Normal("y", μ, σ, observed=y)
```

McElreath specifications of a model (and, for that matter, their PyMC counterparts) are effective and complete, but don't always provide the most intuitive sense of how a model fits together – tracing the dependencies between priors and what they influence can be a pain. Here is how the same model appears once rendered using pyKrusch (see Figure 38). Now, at a glance, we can ascertain:

- The numerical values specifying the shapes of each prior distribution.
- The priors' and likelihood's distributions via images (called 'distrograms'), providing an at-a-glance sense of their shape.
- Helper text underneath each distribution parameter (such as 'location' under the normals' $\mu$ parameters), providing an intuitive sense of what role each parameter plays.

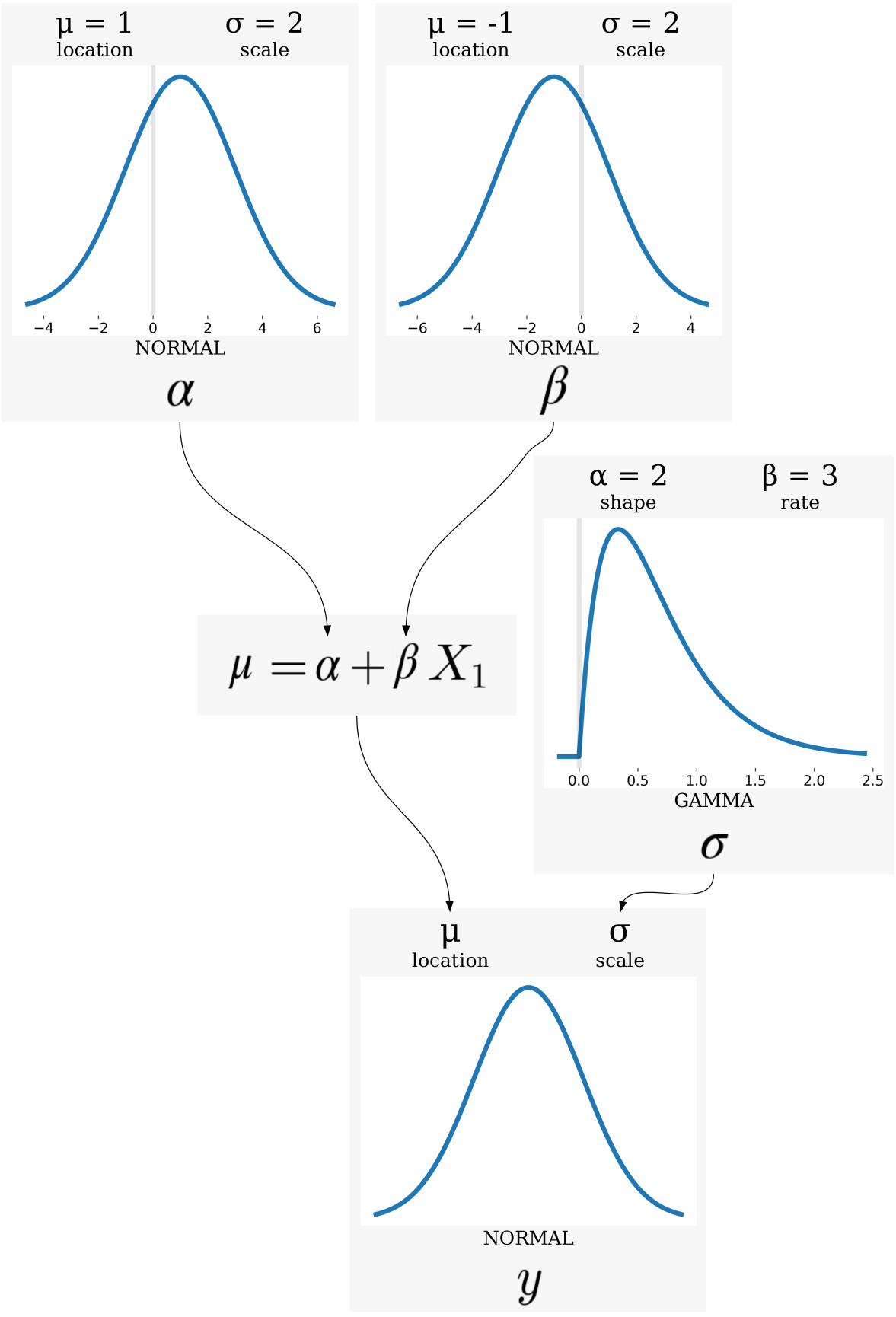


Figure 38: `pyKrusch` Visualization of a Simple Bayesian Model

- The specific relationship between one distribution and one or more other distributions' parameter(s). Arrows originating from the priors point towards a specific parameter (in this case, the likelihoods' parameters), which demystifies the process of determining the role each prior plays in the model overall.
- The specific relationship between distributions and the components of a deterministic equation (usually the 'linear model' component).

`pyKrusch` also represents an innovation on Kruschke's framework, providing information that the diagrams – in their original form – cannot. `pyKrusch` does so in two ways:

- **Dynamic Distrograms**
  - In their original form, Kruschke's diagrams make extensive use of the aforementioned 'distrograms' to provide users with a sense of each distribution's shape. These distrograms invariably take the form of an 'ideal type' distribution – they give a sense of the distribution's general shape, but provide no information about the distribution's actual scale, location, or specific shape. The `pyKrusch` package takes advantage of its automation suite to produce 'dynamic distrograms' wherever possible. Rather than merely representing a distribution's typical appearance, `pyKrusch`'s dynamic distrograms provide users with specific detailed information about most prior distributions' shape, scale, and location.[165]
- **Sampled Posterior Distributions**
  - `pyKrusch` is capable of leveraging `pymc`'s impressive sampling capabilities to include marginal posterior distributions in the resulting diagram. This represents a paradigm shift for the format: the original Kruschke diagrams were strictly for didactic or illustrative purposes. By enabling the representation of posterior distributions, `pyKrusch` permits the augmented Kruschke diagram to play a role in inference and model diagnostics: providing practitioners and readers alike with information about the results of the modelling efforts in question.[166] This functionality will be demonstrated near the end of the'`pyKrusch` Quick-Start Guide' subsection below.

In the next subsection, I walk through the (exceedingly simple) process of applying `pyKrusch` to an extant `PyMC` model, showing how `pyKrusch` effortlessly scales to larger, more complex models.

## `pyKrusch` Quick-Start Guide

Say you're a political sociologist who has recently incorporated Bayesian statistics into your inferential toolbox.[167] You are tasked with developing a model capable of capturing how campaign spending in the 2020 election for the President of the United States of America affected the popular vote differential between the two major parties (Republican and Democrat). You have access to the following information:

- The popular vote differential (Democrat minus Republican) for Republican and Democratic candidates in 371 of the federal congressional districts across 48 states in the 2020 election (dependent variable).
- The campaign spending differential (Democrat minus Republican) in each federal congressional district by the Republican and Democratic parties.
- The state within which each federal congressional district resided as of the 2020 presidential election.

Working with the information you have (and before looking at the data), you theorize that you could use a linear model to predict the effect of campaign spending differential on popular vote differential in each state. The natural choice for such a model would be a Bayesian Hierarchical Model using 'partial pooling', wherein the model coefficients (alpha and beta) would be allowed to vary from state to state, but would all be drawn from an underlying set of prior distributions (called hyperpriors). As such, you specify a hierarchical model and set your substantive priors: you reason that alpha_mu should be uninformative and centred (on the z-scale), so you distribute it as Normal$(0, 2)$. You also elect to create a relatively

[165] `pyKrusch`'s ability to do so is, at the moment, limited. While perfectly capable of creating dynamic distrograms for 'regular,' one-dimensional priors, it does not have the ability to produce dynamic distrograms for multidimensional distributions, arrays of distributions, or any of the distributions whose parameters are distributed as another distribution in the model (which is to say: `pyKrusch` cannot produce a dynamic distrogram for a prior with hyperpriors on its parameters, but can do so for those hyperpriors). In such cases, `pyKrusch` defaults to a 'canonical' representation of the distribution, which can be differentiated from the parameterized version by the former's lack of numbered hash-marks along the x-axis of the distrogram. Rectifying this shortcoming will be a focus of future development efforts on the package.

[166] `pyKrusch`'s ability to do so depends on its dynamic distrogram functionality, and is thus only currently available for variables that are eligible for receiving a dynamic distrogram. Also, it should be noted that parameters are not necessarily independent in the posterior, and thus the marginal posteriors should not be solely relied upon for inference.

[167] Code and data for the following example were drawn from the author's contributions to (McLevey 2021).

uninformative beta_mu, but you're fairly certain that congressional districts where Democrats outspend Republicans will shift the vote differential towards Democrats (and vise versa), so you elect to distribute it as Normal$(1, 2)$. This provides leaves us with the following McElreath specification:

$$
\begin{aligned}
\text{alpha\_mu} &\sim \text{Normal}(0, 2) && \text{[alpha hyperprior]} \\
\text{alpha\_sigma} &\sim \text{Exponential}(1) && \text{[alpha hyperprior]} \\
\text{beta\_mu} &\sim \text{Normal}(1, 2) && \text{[beta hyperprior]} \\
\text{beta\_sigma} &\sim \text{Exponential}(1) && \text{[beta hyperprior]} \\
\text{alpha}_k &\sim \text{Normal}(\text{alpha\_mu}, \text{alpha\_sigma}) && \text{[alpha priors, one per state]} \\
\text{beta}_k &\sim \text{Normal}(\text{beta\_mu}, \text{beta\_sigma}) && \text{[beta priors, one per state]} \\
\text{sigma} &\sim \text{Exponential}(2) && \text{[sigma prior]} \\
\text{mu}_i &= \text{alpha}_{k[i]} + \text{beta}_{k[i]} \cdot x_i && \text{[linear model]} \\
y_i &\sim \text{Normal}(\text{mu}_i, \text{sigma}) && \text{[likelihood]}
\end{aligned}
$$

The PyMC model specification, then, would look something like this:

```
import pymc as pm

with pm.Model() as model:

    # Hyperpriors
    alpha_mu = pm.Normal("alpha_mu", mu=0, sigma=2)
    beta_mu = pm.Normal("beta_mu", mu=1, sigma=2)
    alpha_sigma = pm.Exponential("alpha_sigma", 1)
    beta_sigma = pm.Exponential("beta_sigma", 1)

    # Priors
    alpha = pm.Normal("alpha", mu=alpha_mu, sigma=alpha_sigma, shape=n_states)
    beta = pm.Normal("beta", mu=beta_mu, sigma=beta_sigma, shape=n_states)
    sigma = pm.Exponential("sigma", 2)

    # Linear Model
    mu = alpha[state_idx] + (beta[state_idx]*spend_std)

    # Likelihood
    votes = pm.Normal("votes", mu=mu, sigma=sigma, observed=vote_std)
```

It bears mentioning that the hierarchical linear model used in this example is one of the simplest possible hierarchical models, and already it's getting difficult to keep track of everything. `pyKrusch` can help here; it's as simple as passing one's PyMC model object to the `krusch` function:

```
from pykrusch import krusch

krusch(model)
```

From the above line of code, `pyKrusch` handles the rest. Absent any further alterations (such as passing a filename as an argument to the `krusch` function), a file called `krusch.png` will be deposited in the current working directory (see Figure 39 below).

As mentioned above, the `pyKrusch` package is also capable of representing the posterior distribution of each top-level prior in the model (see Figure 40 below).

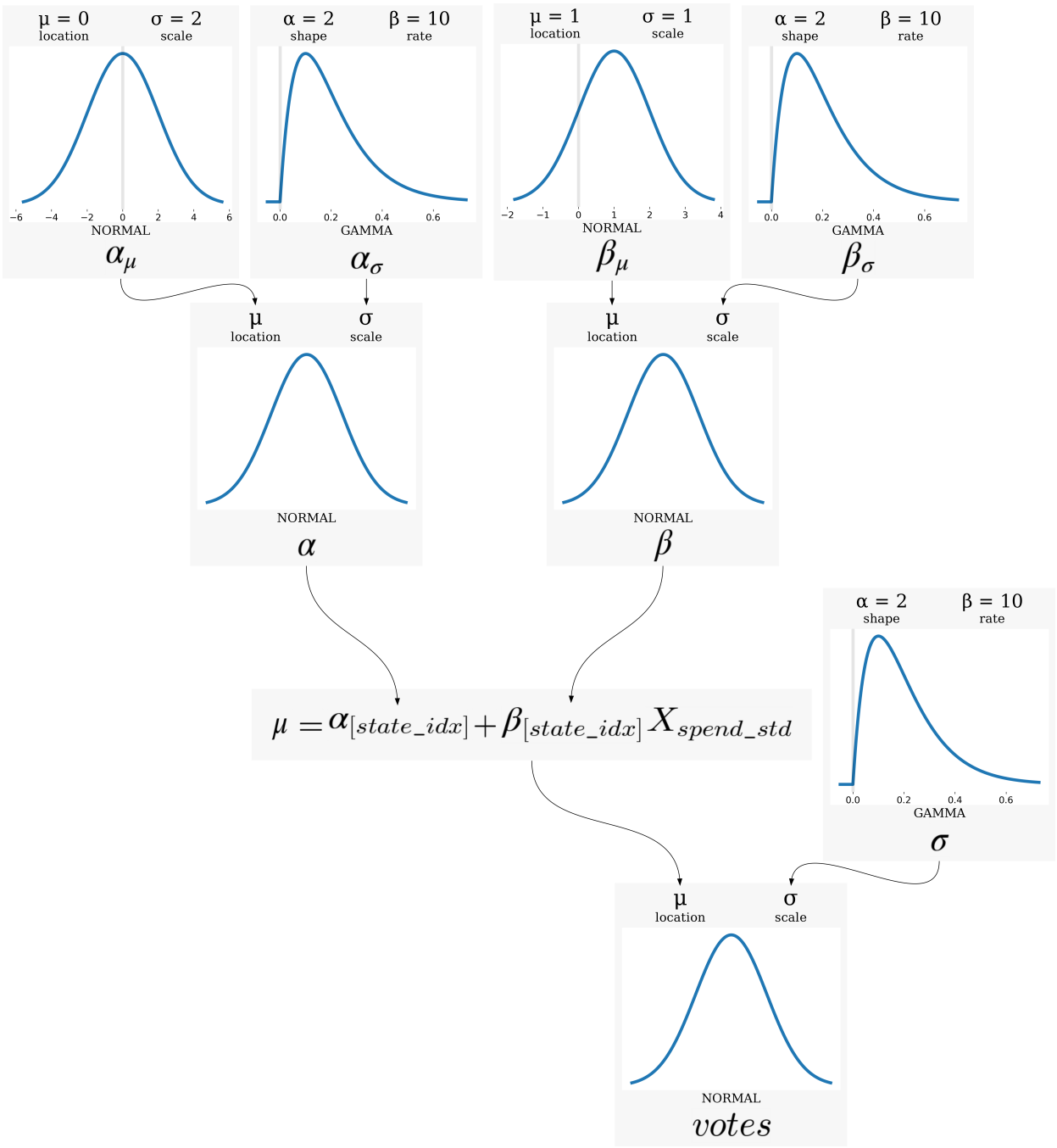


Figure 39: `pyKrusch` Visualization of a Hierarchical Bayesian Model

```
from pykrusch import krusch

with model:
    trace = pm.sample()

krusch(model, posterior_trace=trace)
```

## Conclusion

In this chapter, I argued that the ongoing replication crisis has laid bare the need for the social sciences to adopt the Bayesian paradigm as the primary mode of statistical inference. Many others have made this argument already, but most have done so using argumentative strategies that have not necessarily had the intended effect (Gelman 2008; Lynch and Bartlett 2019). In response, I posited a mode of argumentation that may help erode the barriers preventing widespread adoption of Bayesian inference. This ameliorative mode of argumentation seeks to validate practical Frequentist reservations about Bayesian inference by highlighting the work Bayesians have done – or, where no such work is extant, articulating the steps that could be taken – to assuage Frequentists' concerns.

The final section of this chapter introduced and detailed my own contribution to this effort in the form of `pyKrusch`, a Python package capable of automating the process of visualizing Bayesian model structure specified in the `PyMC` probabilistic programming suite. In my view, `pyKrusch` represents a meaningful contribution because it is not merely an automation tool: it also addresses concerns expressed by John Kruschke (whose diagrams `pyKrusch` is designed to emulate) and builds upon the Kruschke diagram's accessibility and applicability. Kruschke build the diagrams in *Doing Bayesian Data Analysis* (2014) by 'hand,' using image editing software (Kruschke 2012).[168] In a comment on his Doing Bayesian Data Analysis blog dated May 15th 2012 (2012), Kruschke recognized that there was no algorithm/code capable of producing anything resembling his diagrams and indicated that he thought such a project would be a good undertaking. `pyKrusch` also expands upon Kruschke's original intention for his diagrams (namely: as descriptive/didactic tools), permitting the augmented diagrams to assist with inference and model diagnostics. The functionality `pyKrusch` enables cannot be replicated using Kruschke's original framework.

The tragedy of the great debate between the Frequentist and Bayesian camps is that many embroiled in it have chosen to wrap their identity and sense of purpose in arguing for one side at the expense of the other. Frequentists are not to blame; the ritualistic use of inappropriate Frequentist techniques is. Bayesian inference can help here, and it is up to Bayesians to demonstrate – through actions, through software, through amelioration – that their chosen approach is accessible, interpretable, rigorous, and will substantially help with what ails the quantitative social sciences. The purpose of this chapter is not to cast aspersions on Frequentist social scientists; it is to improve our collective capacity to do good science. Any effort devoted to that end is, in my mind, intrinsically worthwhile.

[168] In his blog Kruschke describes his workflow as primarily centring on the manual assembly of diagrams in Libre-Office Draw (2012).

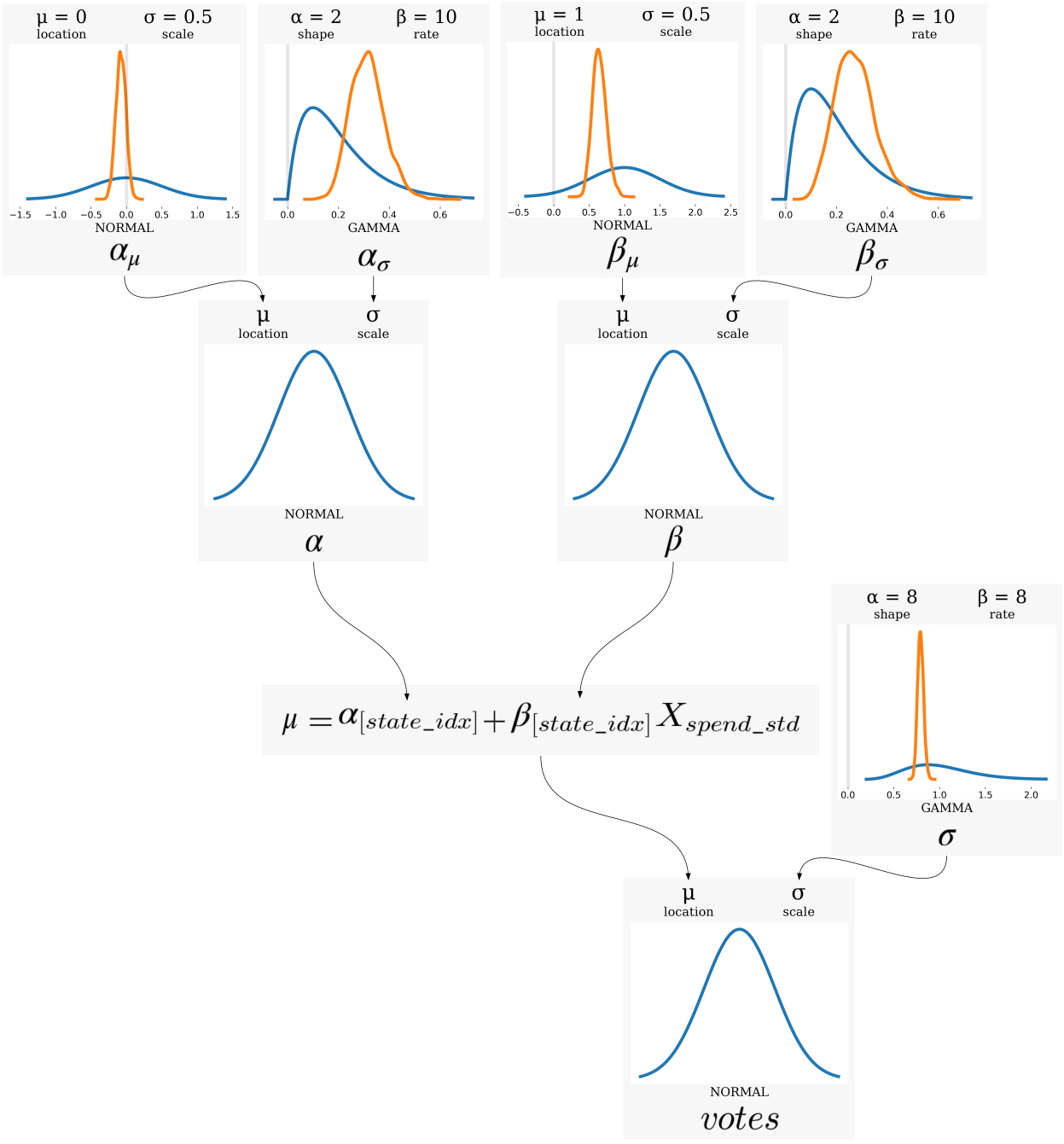


Figure 40: `pyKrusch` Visualization of a Hierarchical Bayesian Model with Posterior Distributions

# Chapter 6 – Conclusion: Bringing The Foundational Cycle Into Being

Pierson Browne

Quantitative sociology has been blinded by the opacity of the methods it most frequently employs. Owing, in large part, to a prevailing lack of methodological transparency and reproducibility, quantitative sociology has led itself into a 'replication crisis' (Burrows and Savage 2014; Baker 2016; Gelman and Loken 2016; Bird 2020), which has precipitated widespread skepticism about the reliability of our findings (Wiggins and Christopherson 2019).

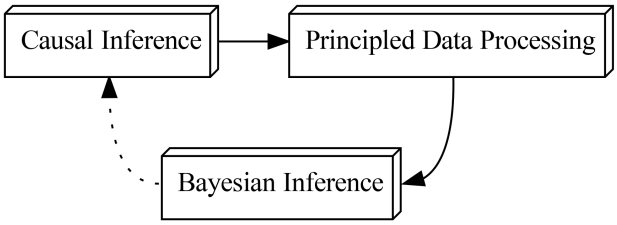


Figure 41: The foundational cycle.

This dissertation argues for the enshrinement of methodological transparency and reproducibility as core principles for quantitative sociology via a 'foundational cycle' consisting of causal inference, principled data processing, and Bayesian inference.

Chapter 2 starts from the premise that valid statistical inference about empirical effects is not possible in the absence of a causal model, and that causal inference provides a principled framework for operationalizing theory, domain knowledge, and past research to develop a causal estimation strategy (Pearl, Glymour, and Jewell 2016; McElreath 2021b). I offered a brief account of the development of two major schools of causal inference in the social sciences – graphical approaches and the potential outcomes framework – and espoused Judea Pearl's theory of inferred causation and structural causal models (2009b) as the causal framework most suitable for quantitative sociology. In the third chapter of the dissertation, I used data gathered from 'top' sociological journals in the year 2022 to show that quantitative sociology overwhelmingly does not use transparent causal models.

This dissertation conceptualizes causal inference as a powerful way of unifying preliminary activities – such as specifying research questions, variable selection, research design, statistical models, and so on – under a single banner and formalizing their collective influence on downstream research activities. In this way, Causal Inference provides researchers with the means to explicitly show how they have used theory, past research, and domain knowledge to build an approach to estimating the causal effect of interest. This approach is then used as the basis for and motivation behind the data collection and statistical inference stages of the study; as such, Causal Inference anchors the other two components of this dissertation.

In the fourth chapter, I begin with the observation that typical quantitative social science projects, in recent years, have come to demand increasingly large expenditures of time and effort on the collection, distillation, storage, and use of truly gargantuan datasets. Most such efforts are committed by skilled amateurs[169] whose good-faith efforts are often sufficient to transform raw data into a viable basis for a statistical analysis, but in so doing create a bespoke morass of code that cannot easily be understood, altered, or reverse-engineered – this goes for outside observers and the project's creators alike.

Principled Data Processing (PDP) is a set of principles and guidelines designed to ensure that data processing pipelines remain transparent, auditable, scalable, and reproducible, regardless of the researchers' collective familiarity with database management (Ball 2016b; Barrett 2022). In the context of the dissertation, PDP serves as the connective tissue between Causal Inference and Bayesian Inference; it safeguards the utility and ensures the reliability of the research project's data and the processing thereof.

In the fifth chapter, I argue that the widespread, uncritical use of Frequentist inference has led quantitative sociology into a credibility quagmire from which we cannot hope to extricate ourselves without the assistance of the Bayesian paradigm. Despite the unambiguous gains to be made by switching *en masse*, Bayesian inference is utilized in a vanishingly small proportion of all quantitative social scientific publications (Lynch and Bartlett 2019).[170] The chapter contends that this

[169] Not 'amateurs' in the research sense, but 'amateurs' with respect to formalized training in programming and database management.

[170] Although the the review I undertook – the results from which were reported in the third chapter of this dissertation – was primarily focused on Causal Methodology, I categorized each quantitative article as either 'Frequentist' or 'Bayesian.' Of the 279 quantitative articles reviewed, only 4 were deemed 'Bayesian.' Broadly speaking, this finding aligns with those in Lynch and Bartlett (2019).

sorry state of affairs stems, in part, from the Bayesian camp's lack of attention paid to the barriers of entry practicing Frequentists face when weighting whether to switch paradigms. As part of a larger mission to make Bayesian models as transparent and accessible as possible, the fifth chapter introduces `pyKrusch`, a Python package capable of producing detailed Kruschke-style diagrams of Bayesian model structure. `pyKrusch` builds on Kruschke's groundbreaking visualizations by implementing dynamic kernel density estimates of the distributions present in each Bayesian model, permitting the direct examination of dynamic parameterizations, and visualization of marginal posterior distributions.

Bayesian inference is the final step in the lifespan of any single given research project, but in the context of this dissertation and the foundational cycle, Bayes also lays the groundwork for future research. Without Bayesian inference, past research can only motivate the *narrative* of future research. By adhering to the Bayesian paradigm, researchers can publish direct estimations of their effects' posterior distributions. These distributions can then serve as the basis for a prior distribution in future research, which creates a direct mathematical link between extant findings and future research. In this sense, Bayesian Inference serves as the interface between researchers and the outside world – it rests atop Causal Inference and Principled Data Processing, making the insights contained throughout the project available in a directly useful, easily-understood format.

# Continuable Research

Doing reproducible and transparent research is a laudable goal in its own right, but striving for such research principles will have enormous downstream effects on how published results are utilized by further research. In this dissertation's introduction, I proposed the notion of 'continuable research' as a way of encapsulating the foundational cycle's revolutionary potential: by making all quantitative sociological research transparent and reproducible, we open the door to a new model of knowledge accumulation wherein past results and methods directly influence further research efforts.[171] Using the 'foundational cycle' diagram, I mean to imply that by solidifying the place of the three boxes and two solid arrows, the third dashed arrow is more likely to materialize.

Were most quantitative social scientists to adopt the foundational cycle as a core set of research practices, I strongly believe that we would reap benefits untold. Even a marginal increase in the proportion of social scientists utilizing the foundational cycle has the potential to radically realign the collective norms which govern our fields. In my view, long-term change for the better will emerge less from agitation[172] and more from action. I see two primary avenues in which this kind of "leading by example" can take place: pedagogy and praxis.

## Pedagogy

The most significant starting place for the foundational cycle to take root is, of course, in the classroom. The components of the foundational cycle should be taught as aspects of a cohesive whole – one which emphasizes competency in statistical programming, command-line interaction, and data collection techniques. Throughout, the value of "honest anarchy" (McElreath 2021b) should be of the utmost importance; rather than teaching rigorous orthodoxy, students should be encouraged to make mistakes in a transparent setting and learn from others' insights.

In this regard, promising signs are already beginning to emerge. Richard McElreath's *Statistical Rethinking* (2020) is ostensibly a textbook on Bayesian inference, but it foregrounds the value of causal inference and transparency throughout. It positions itself as an advanced textbook for PhD-level researchers, and (as evinced by its very name) emphasizes how received statistical wisdom can be reconsidered from a Bayesian perspective.

John McLevey's *Doing Computational Social Science* is the text that, in my opinion, most closely aligns with the goals of this dissertation.[173] It is designed with a steep learning curve that assumes no statistical background and starts by walking students through the Python programming language, the command-line interface, version control via Git, and approaches to data management. It then leverages these core competencies to introduce students to an array of workhorse and cutting-edge data collection techniques, machine learning algorithms, and approaches to statistical inference.

[171] In both the numerical and theoretical senses.

[172] In a similar dynamic as the one I describe in Chapter 2 of this dissertation, attack-style arguments alone will be unlikely to catalyze any meaningful shift.

[173] A fact which should hardly be surprising as it was, of course, written by my supervisor! I also contributed code and prose to several chapters, most notably the section on Bayesian inference. In this case, I wear my bias shamelessly!

Using one of the above textbooks as the cornerstone of a quantitative sociology curriculum, pedagogy can be built outwards to suit institutional requirements. Frequentist inference, for instance, should still be a standard component of all university-level quantitative training – if to no other end than to provide students with the familiarity required to assess and comprehend the Frequentist model results present in almost all extant quantitative publications. In this way, the foundational cycle can be incorporated as an overarching framework into which existing components of quantitative research can fit – the benefit of added context will outweigh the increased conceptual overhead.

## Praxis

The other *sine qua non* for any potential 'continuable research' revolution is praxis. Before the foundational cycle can hope to take root in quantitative sociology, some of us are going to have to just start doing it – torpedoes be damned. The act of switching to the continuable research paradigm is neither straightforward nor without costs. Academic 'success,' in the form of research publications, is predicated on a vast interconnected network of resources, organizations, and individuals, many of whom are motivated by incentives and desires that may be perpendicular – or even run directly counter to – the end goal of achieving widespread research continuity.

At present, quantitative sociology is locked in what might be best described as a "Researcher's Dilemma," wherein perverse incentives encourage researchers to pursue individual gains at the expense of others.[174] This is just another way of regarding the well-documented conditions (often referred to as 'publish-or-perish') that encourage the production of quick science with attention-grabbing headlines and little regard for the downstream utility or reliability of said findings (Gigerenzer 2018; McElreath 2020, 2021b). In recent years, young academics have come under increasing pressure as the paucity of tenure-track or other stable scholarly positions has intensified competition over what little remains (Kiai 2019; Furnham 2021; Van Dalen 2021). Although I cannot summon any evidence to this end, I suspect that these two phenomena are linked and mutually-reinforcing: the pressures associated with publish-or-perish encourage the production of unreliable science, which undermines public trust in academia, which emboldens governments and

[174] Here, I'm using "The Prisoner's Dilemma" – a classic game theoretical conceit – as inspiration for what I'm calling "The Researcher's Dilemma". The Prisoner's Dilemma, put simply, is a thought experiment that posits the following problem: two individuals are caught committing a crime together and imprisoned. The police have incontrovertible evidence that both prisoners are guilty of a lesser crime (say, breaking and entering), but lack the evidence to convict them of mutually committing a more serious offence (say, manslaughter). The police separate the two prisoners and forbid any communication between them, and then approach Prisoner A with a deal: "Currently, you're looking at 2 years in prison for breaking and entering. If you turn informant for us and implicate your partner for manslaughter, we'll knock your sentence down to 1 year. Be warned: we're offering the same deal to Prisoner B, and if they rat on you, we'll lock you up for a full 10-year sentence and knock *their* sentence down to 1 year. If you both implicate each other, you'll both do 8." Given that the prisoners cannot communicate or coordinate, what should Prisoner A do? At first blush, it seems obvious that staying quiet (a strategy known as "cooperation" in the literature) is the best move. If both prisoners refuse to inform, they'll each spend 2 years in jail for a total of 4 years between them. From the prisoners' perspectives, this is the best outcome overall, as a single defection would land one prisoner in jail for 10 and the other for 1, giving a much higher total of 11 years behind bars between them. A mutual defection is the worst outcome overall, as both would receive 8 years, for a total of 16. In spite of the obvious benefits of cooperation, the field of game theory has rigorously shown that defection is the best option, and here's why: assuming Prisoner A wishes to minimize their time spent in prison they should always defect, as doing so reduces their sentence *regardless of what Prisoner B decides* (Axelrod 1984). Since Prisoner A has no means of coercing, controlling, or having any advance knowledge of what Prisoner B will do, they're forced to assume the worst. Even if this assumption is mistaken and Prisoner B chooses the altruistic option, Prisoner A still gets the best outcome! The Prisoner's Dilemma – and it's $n$-player counterparts, "The Stag Hunt" and "The Tragedy of the Commons" – are used to show how systems can encourage 'players' to defect against one another, in spite of the fact that mutual cooperation produces a far better outcome for all. Robert Axelrod's groundbreaking study of the Prisoner's Dilemma – titled "The Evolution of Cooperation" (1984) – provided one of the first clear articulations of how cooperation can emerge from systems with perverse incentives. His study primarily took the form of a computer tournament, where academics and practitioners were invited to submit a simple code-based "strategy" that would compete against other such strategies. The strategies that received the best average payoff across many iterations of the prisoner's dilemma would be crowned victorious. Although a wide variety of such strategies were submitted, the consistent winner was one of the simplest. Submitted by Canadian mathematical psychologist Anatol Rapoport, the 'tit-for-tat' strategy had a disarmingly straightforward approach to the game: cooperate on the first iteration, and then copy the opponent's previous move on all subsequent iterations. The beauty of the 'tit-for-tat' strategy was that it acted as a good-faith actor that would not let past transgressions go unpunished. It was capable of reaping maximum payoffs by cooperating with other good-faith actors, would punish bad-faith actors by giving as good as it got, and – most critically – was willing to 'forgive' former defectors – provided they were willing to play nice. The key to all of this was Axelrod's realization that cooperation would not emerge in one-off systems where the two players might never play the same game together again. Only when forced into the same situation with the same players across an arbitrarily large number of iterations was cooperation possible. 'Tit-for-tat' all but proved this by emerging as the unchallenged victor of both tournaments – the second of which featured several strategies specifically designed to defeat it. Later in the book, Axelrod extends his analysis of the iterated Prisoner's Dilemma to an evolutionary setting, wherein successful strategies survive and/or replicate, and unsuccessful strategies are slowly eliminated from the environment. From these later tournaments, Axelrod produced the additional insight that even the best good-faith actors (including tit-for-tat) can come to ruin if forced to interact with a comparatively larger population of bad-faith actors. This dynamic was reversed, however, if actors were allowed to preferentially interact with others whose past interactions were the most fruitful. In other words, if good-faith strategies were allowed to form a clique of preferential interaction, their gains would eventually allow them to dominate the ecosystem and force erstwhile bad-faith actors to play nice, to the benefit of all.

funding bodies to reduce their level of support for academic research, which further limits the number of positions available, which further intensifies the publish-or-perish dynamic, and so on.

If those working to advance the common good find themselves primarily interacting with those acting in their own self-interest, the altruistic party will all-too-often come out the worse for it (Axelrod 1984). It isn't until a critical mass of common-good cooperators find one another and cluster together that they can insulate themselves from the depredation of the system as a whole. In concrete terms, this would take the form of quantitative sociology communities that publish, critique, and take up research in the same journals and conferences. Whether purpose-built or merely adopted, these forums would need to be uncompromising about the components of the foundational cycle: all research must foreground structural causal models, transparent and reproducible data processing pipelines, and fully-specified Bayesian inferential models (where appropriate).

By way of an example: consider the imposing problem of getting one's 'continuable' research published. While most social scientific journals would likely regard the addition of causal inference and principled data processing with something ranging from indifference to approval, the foundational cycle's insistence on Bayesian inference could run afoul of a journal's standard modus operandi. At present, quantitative sociology journals rely (in large part) on the Frequentist concept of 'statistical significance' to assess whether a submitted quantitative article is publishable (McShane et al. 2019).[175] Bayesian inference provides no such 'one-size-fits-all' heuristic for sorting the proverbial wheat from the proverbial chaff – from the paradigm's point of view, all statistical results are equally 'interesting.' The implication of this is that 'continuable' research forces peer reviewers and journal editors to search elsewhere for the markers that can be used to separate publishable results from those unfit for print. Anyone using the 'continuable' paradigm should be prepared to o'erleap comparatively high publication hurdles for the foreseeable future.

Unfortunately, I cannot see any way forward that doesn't involve early adopters of the 'continuable' paradigm willingly accepting the costs of making the switch. Said costs will stay high until such a time as a critical mass has been achieved, whereupon the disadvantages of being largely excluded from the usual publishing circuit will all but disappear. The hope would be that in due time, the influence of this new 'continuable' bloc would start to sway other journals into being friendly towards – if not outright demanding – a principled approach to transparency and reproducibility.

With sufficient verve and a willingness to weather vagaries, we might just stand a chance of changing quantitative sociology for the better.

[175] See also: the data in Appendix A of this dissertation, showing that only ~1.5% of the quantitative papers published in 'top' sociology journals in 2022 were Bayesian – the remainder used Frequentist methods.

# References


Abbott, Andrew. 1988. "Transcending General Linear Reality." *Sociological Theory*, 169–86.

Achen, Christopher H. 1987. "As Statisticians See Us: Comments on Freedman's Paper 'as Others See Us...'" *Journal of Educational Statistics* 12 (2): 148–50.

Aksoy, Ozan, and Diego Gambetta. 2022. "Commitment Through Sacrifice: How Longer Ramadan Fasting Strengthens Religiosity and Political Islam." *American Sociological Review* 87 (4): 555–83.

Alves, Matheus Facure. 2022. "Causal Inference for the Brave and True." *Causal Inference for The Brave and True - Causal Inference for the Brave and True*. https://matheusfacure.github.io/python-causality-handbook/landing-page.html.

Amutuhaire, Tibelius. 2022. "The Reality of the 'Publish or Perish'concept, Perspectives from the Global South." *Publishing Research Quarterly* 38 (2): 281–94.

Anderson, Samantha F. 2020. "Misinterpreting p: The Discrepancy Between p Values and the Probability the Null Hypothesis Is True, the Influence of Multiple Testing, and Implications for the Replication Crisis." *Psychological Methods* 25 (5): 596.

Arminger, Gerhard, and George W Bohrnstedt. 1987. "Making It Count Even More: A Review and Critique of Stanley Lieberson's Making It Count: The Improvement of Social Theory and Research." *Sociological Methodology* 17: 363–72.

Aurini, Janice, John McLevey, Allyson Stokes, and Rob Gorbet. 2017. "Classroom Robotics and Acquisition of 21st Century Competencies: An Action Research Study of Nine Ontario School Boards." *Report for the Council of Ontario Directors of Education (CODE) and the Ministry of Education, Ontario*.

Axelrod, Robert. 1984. *The Evolution of Cooperation*. Basic Books.

Baker, Monya. 2016. "Reproducibility Crisis." *Nature* 533 (26): 353–66.

Ball, Patrick. 2016a. "Principled Data Processing." https://hrdag.org/talks-discussions/.

———. 2016b. "The Task Is a Quantum of Workflow." *HRDAG*. HRDAG. https://hrdag.org/2016/06/14/the-task-is-a-quantum-of-workflow/.

Barrett, Larry. 2022. "Lessons at HRDAG: Holding Public Institutions Accountable." *HRDAG*. HRDAG. https://hrdag.org/2022/09/12/lessons-at-hrdag-holding-public-institutions-accountable/.

Barringer, Sondra N, Scott R Eliason, and Erin Leahey. 2013. "A History of Causal Analysis in the Social Sciences." *Handbook of Causal Analysis for Social Research*, 9–26.

Bartram, David. 2021. "Cross-Sectional Model-Building for Research on Subjective Well-Being: Gaining Clarity on Control Variables." *Social Indicators Research* 155: 725–43.

Bayes, Thomas. 1763. "LII. An Essay Towards Solving a Problem in the Doctrine of Chances. By the Late Rev. Mr. Bayes, FRS Communicated by Mr. Price, in a Letter to John Canton, AMFR s." *Philosophical Transactions of the Royal Society of London*, no. 53: 370–418.

Begley, C Glenn, and Lee M Ellis. 2012. "Raise Standards for Preclinical Cancer Research." *Nature* 483 (7391): 531–33.

Bentler, PM. 1987. "Structural Modeling and the Scientific Method: Comments on Freedman's Critique." *Journal of Educational Statistics* 12 (2): 151–57.

Berk, Richard A. 2004. *Regression Analysis: A Constructive Critique*. Vol. 11. Sage.

Bernoulli, Jakob. 1713. "Ars Conjectandi: Usum & Applicationem Praecedentis Doctrinae in Civilibus, Moralibus & Oeconomicis." *Translated into English by Oscar Sheynin. Basel: Turneysen Brothers,. Chap*.

Bhanot, Karan. 2017. "Dataset Creation and Cleaning: Web Scraping Using Python – Part 1." 2017. https://towardsdatascience.com/dataset-creation-and-cleaning-web-scraping-using-python-part-1-33afbf360b6b.

Bird, Alexander. 2020. "Understanding the Replication Crisis as a Base Rate Fallacy." *The British Journal for the Philosophy of Science*.

Birkinshaw, Julian, and Jonas Ridderstråle. 2010. "Adhocracy for an Agile Age." *Organization Science* 22 (5): 1286–96.

Blalock, Hubert M. 1962a. "Four-Variable Causal Models and Partial Correlations." *American Journal of Sociology* 68 (2): 182–94.

———. 1962b. "Further Observations on Asymmetric Causal Models." *American Sociological Review* 27 (4): 542–45.

Blalock Jr, Hubert M. 1961. "Correlation and Causality: The Multivariate Case." *Social Forces* 39 (3): 246–51.

———. 1963. "Making Causal Inferences for Unmeasured Variables from Correlations Among Indicators." *American Journal of Sociology* 69 (1): 53–62.

Blei, David M. 2014. "Build, Compute, Critique, Repeat: Data Analysis with Latent Variable Models." *Annual Review of Statistics and Its Application*.

Bollen, Kenneth A. 1995. "Structural Equation Models That Are Nonlinear in Latent Variables: A Least-Squares Estimator." *Sociological Methodology*, 223–51.

Bollen, Kenneth A, and Judea Pearl. 2013. "Eight Myths about Causality and Structural Equation Models." *Handbook of Causal Analysis for Social Research*, 301–28.

Bolthausen, Erwin, and Mario V Wuthrich. 2013. "Bernoulli's Law of Large Numbers." *ASTIN Bulletin: The Journal of the IAA* 43 (2): 73–79.

Bourdieu, Pierre. 1988. *Homo Academicus*. Stanford University Press.

Box, George EP. 1976. "Science and Statistics." *Journal of the American Statistical Association* 71 (356): 791–99.

———. 1980. "Sampling and Bayes' Inference in Scientific Modelling and Robustness." *Journal of the Royal Statistical Society Series A: Statistics in Society* 143 (4): 383–404.

Box, George EP, and William J Hill. 1967. "Discrimination Among Mechanistic Models." *Technometrics* 9 (1): 57–71.

Brayne, Sarah. 2017. "Big Data Surveillance: The Case of Policing." *American Sociological Review* 82 (5): 977–1008.

———. 2020. *Predict and Surveil: Data, Discretion, and the Future of Policing*. Oxford University Press, USA.

Britten, Gregory L, Yara Mohajerani, Louis Primeau, Murat Aydin, Catherine Garcia, Wei-Lei Wang, Benoît Pasquier, BB Cael, and François W Primeau. 2021. "Evaluating the Benefits of Bayesian Hierarchical Methods for Analyzing Heterogeneous Environmental Datasets: A Case Study of Marine Organic Carbon Fluxes." *Frontiers in Environmental Science* 9: 491636.

Browne, Pierson. 2021. *pykrusch* (version 0.1.0). https://github.com/pbrowne88/pykrusch.

Browne, Pierson, Tyler Crick, Rachel Wood, and John McLevey. 2021. *pdpp* (version 0.4.3). https://github.com/UWNETLAB/pdpp.

Burks, Barbara S. 1926. "On the Inadequacy of the Partial and Multiple Correlation Technique." *Journal of Educational Psychology* 17 (8): 532.

Burrows, Roger, and Mike Savage. 2014. "After the Crisis? Big Data and the Methodological Challenges of Empirical Sociology." *Big Data & Society* 1 (1): 2053951714540280.

Busemeyer, Jerome R, and Lawrence E Jones. 1983. "Analysis of Multiplicative Combination Rules When the Causal Variables Are Measured with Error." *Psychological Bulletin* 93 (3): 549.

Cartwright, Nancy. 1999a. "Causal Diversity and the Markov Condition." *Synthese* 121 (1/2): 3–27.

———. 1999b. *The Dappled World: A Study of the Boundaries of Science*. Cambridge University Press.

———. 2004. "Causation: One Word, Many Things." *Philosophy of Science* 71 (5): 805–19.

Christensen, Garret, Jeremy Freese, and Edward Miguel. 2019. *Transparent and Reproducible Social Science Research: How to Do Open Science*. University of California Press.

Cinelli, Carlos, Andrew Forney, and Judea Pearl. 2021. "A Crash Course in Good and Bad Controls." *Sociological Methods & Research*, 00491241221099552.

Clark, Jennifer, Natalia Muhlemann, Fanni Natanegara, Andrew Hartley, Deborah Wenkert, Fei Wang, Frank E Harrell, and Ross Bray. 2022. "Why Are Not There More Bayesian Clinical Trials? Perceived Barriers and Educational Preferences Among Medical Researchers Involved in Drug Development." *Therapeutic Innovation & Regulatory Science*, 1–9.

Clayton, Aubrey. 2021. "Bernoulli's Fallacy." In *Bernoulli's Fallacy*. Columbia University Press.

Cliff, Norman. 1987. "Comments on Professor Freedman's Paper." *Journal of Educational Statistics* 12 (2): 158–60.

Collaboration, Open Science. 2015. "Estimating the Reproducibility of Psychological Science." *Science* 349 (6251): aac4716.

Colling, Lincoln J, and Dénes Szűcs. 2021. "Statistical Inference and the Replication Crisis." *Review of Philosophy and Psychology* 12 (1): 121–47.

Collins, Harry M. 1974. "The TEA Set: Tacit Knowledge and Scientific Networks." *Science Studies* 4 (2): 165–85.

———. 1992. *Changing Order: Replication and Induction in Scientific Practice*. University of Chicago Press.

———. 1998. "The Meaning of Data: Open and Closed Evidential Cultures in the Search for Gravitational Waves." *American Journal of Sociology* 104 (2): 293–338.

———. 2016. "Reproducibility of Experiments: Experimenters' Regress, Statistical Uncertainty Principle, and the Replication Imperative." *Reproducibility: Principles, Problems, Practices, and Prospects*, 65À82.

Collins, Randall. 1998. "The Sociological Eye and Its Blinders." *Contemporary Sociology* 27 (1): 2–7.

Cox, David Roxbee. 1958. "Planning of Experiments."

Cox, Richard T. 1946. "Probability, Frequency and Reasonable Expectation." *American Journal of Physics* 14 (1): 1–13.

Cunningham, Scott. 2021. *Causal Inference: The Mixtape*. Yale university press.

Davidson-Pilon, Cameron. 2015. *Bayesian Methods for Hackers: Probabilistic Programming and Bayesian Inference*. Addison-Wesley Professional.

Dowe, Phil. 1992. "Wesley Salmon's Process Theory of Causality and the Conserved Quantity Theory." *Philosophy of Science* 59 (2): 195–216.

Downey, Allen B. 2021. *Think Bayes*. " O'Reilly Media, Inc.".

Duncan, Otis Dudley. 1975. *Introduction to Structural Equation Models*. Academic Press.

Dushoff, Jonathan, Morgan P Kain, and Benjamin M Bolker. 2019. "I Can See Clearly Now: Reinterpreting Statistical Significance." *Methods in Ecology and Evolution* 10 (6): 756–59.

Efron, Bradley. 1986. "Why Isn't Everyone a Bayesian?" *The American Statistician* 40 (1): 1–5.

Faya, Paul, Perceval Sondag, Steven Novick, Dwaine Banton, John W Seaman Jr, James D Stamey, and Bruno Boulanger. 2021. "The Current State of Bayesian Methods in Nonclinical Pharmaceutical Statistics: Survey Results and Recommendations from the DIA/ASA-BIOP Nonclinical Bayesian Working Group." *Pharmaceutical Statistics* 20 (2): 245–55.

Fisher, R. A. 1935. *The Design of Experiments*. Oliver & Boyd.

Fisher, Ronald. 1955. "Statistical Methods and Scientific Induction." *Journal of the Royal Statistical Society: Series B (Methodological)* 17 (1): 69–78.

Fisher, Ronald A. 1956. "Mathematics of a Lady Tasting Tea." *The World of Mathematics* 3 (part 8): 1514–21.

Fox, John. 1987. "Statistical Models for Non-Experimental Data: A Comment on Freedman." *Journal of Educational Statistics* 12 (2): 161–64.

Freedman, David A. 1987. "As Others See Us: A Case Study in Path Analysis." *Journal of Educational Statistics* 12 (2): 101–28.

Freese, Jeremy. 2007. "Replication Standards for Quantitative Social Science: Why Not Sociology?" *Sociological Methods & Research* 36 (2): 153–72.

Freese, Jeremy, and David Peterson. 2017. "Replication in Social Science." *Annual Review of Sociology* 43: 147–65.

Freese, Jeremy, Tamkinat Rauf, and Jan Gerrit Voelkel. 2022. "Advances in Transparency and Reproducibility in the Social Sciences." *Social Science Research* 107: 102770.

Furnham, Adrian. 2021. "Publish or Perish: Rejection, Scientometrics and Academic Success." *Scientometrics* 126 (1): 843–47.

Gelman, Andrew. 2008. "Objections to Bayesian Statistics." *Bayesian Analysis* 3 (3): 445–49.

———. 2016. "Replication Crisis Crisis: Why i Continue in My Pessimistic Conclusions about Reproducibility." Statistical Modeling, Causal Inference, and Social Science. https://statmodeling.stat.columbia.edu/2016/03/05/29195/.

———. 2018. "The Failure of Null Hypothesis Significance Testing When Studying Incremental Changes, and What to Do about It." *Personality and Social Psychology Bulletin* 44 (1): 16–23.

———. 2019. "'The Book of Why' by Pearl and Mackenzie." Statistical Modeling, Causal Inference, and Social Science. 2019. https://statmodeling.stat.columbia.edu/2019/01/08/book-pearl-mackenzie/.

Gelman, Andrew, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. 2020. *Bayesian Data Analysis, Third Edition*. Chapman and Hall/CRC.

Gelman, Andrew, and Eric Loken. 2016. "The Statistical Crisis in Science." In *The Best Writing on Mathematics 2015*, 305–18. Princeton University Press.

Gelman, Andrew, and Cosma Rohilla Shalizi. 2013. "Philosophy and the Practice of Bayesian Statistics." *British Journal of Mathematical and Statistical Psychology* 66 (1): 8–38.

Gentzkow, M, and J. M. Shapiro. 2014. "Code and Data for the Social Sciences: A Practitioner's Guide."

Geron, Aurelien. 2019. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. "* O'Reilly Media, Inc.".

Gigerenzer, Gerd. 1998. "We Need Statistical Thinking, Not Statistical Rituals." *Behavioral and Brain Sciences* 21 (2): 199–200.

———. 2004. "Mindless Statistics." *The Journal of Socio-Economics* 33 (5): 587–606.

———. 2018. "Statistical Rituals: The Replication Delusion and How We Got There." *Advances in Methods and Practices in Psychological Science* 1 (2): 198–218.

Gill, Jeff, and Simon Heuberger. 2019. "Bayesian Modeling and Inference: A Postmodern Perspective." *LC Curini & J. Franzese, Robert J., Eds,"Handbook of Research Methods in Political Science & International Relations," Sage*.

Gleditsch, Nils Petter, Patrick James, James Lee Ray, and Bruce Russett. 2003. "Editors' Joint Statement: Minimum Replication Standards for International Relations Journals." *International Studies Perspectives* 4 (1): 105.

Haavelmo, Trygve. 1943. "The Statistical Implications of a System of Simultaneous Equations." *Econometrica, Journal of the Econometric Society*, 1–12.

Halford, Susan, and Mike Savage. 2017. "Speaking Sociologically with Big Data: Symphonic Social Science and the Future for Big Data Research." *Sociology* 51 (6): 1132–48.

Hartzell, Caroline. 2015. "Data Access and Research Transparency (DA-RT): A Joint Statement by Political Science Journal Editors." *Conflict Management and Peace Science* 32 (4): 355–55.

Heckman, James J. 1992. "Haavelmo and the Birth of Modern Econometrics: A Review of the History of Econometric Ideas by Mary Morgan." *Journal of Economic Literature* 30 (2): 876–86.

Holland, Paul W. 1986. "Statistics and Causal Inference." *Journal of the American Statistical Association* 81 (396): 945–60.

Hope, Keith. 1984. *As Others See Us: Schooling and Social Mobility in Scotland and the United States*. Cambridge University Press.

———. 1987. "Barren Theory or Petty Craft? A Response to Professor Freedman." *Journal of Educational Statistics* 12 (2): 129–47.

Huang, Yimin, and Marco Valtorta. 2012. "Pearl's Calculus of Intervention Is Complete." *arXiv Preprint arXiv:1206.6831*.

Imbens, Guido W. 2020. "Potential Outcome and Directed Acyclic Graph Approaches to Causality: Relevance for Empirical Practice in Economics." *Journal of Economic Literature* 58 (4): 1129–79.

Ioannidis, John PA. 2018. "The Proposal to Lower p Value Thresholds to. 005." *Jama* 319 (14): 1429–30.

Jacobs, Jerry A. 2016. "Journal Rankings in Sociology: Using the h Index with Google Scholar." *The American Sociologist* 47: 192–224.

Jacoby, William G, Sophia Lafferty-Hess, and Thu-Mai Christian. 2017. "Should Journals Be Responsible for Reproducibility." *Inside Higher Ed* 17.

Jaynes, Edwin T. 2003. *Probability Theory: The Logic of Science*. Cambridge university press.

Joffe, Marshall M, Wei Peter Yang, and Harold I Feldman. 2010. "Selective Ignorability Assumptions in Causal Inference." *The International Journal of Biostatistics* 6 (2).

Joreskog, Karl G, and Marielle Van Thillo. 1972. "LISREL: A General Computer Program for Estimating a Linear Structural Equation System Involving Multiple Indicators of Unmeasured Variables."

Karlin, Samuel. 1987. "Path Analysis in Genetic Epidemiology and Alternatives." *Journal of Educational Statistics* 12 (2): 165–77.

Kenny, David A, and Charles M Judd. 1984. "Estimating the Nonlinear and Interactive Effects of Latent Variables." *Psychological Bulletin* 96 (1): 201.

Key, Ellen M. 2016. "How Are We Doing? Data Access and Replication in Political Science." *PS: Political Science & Politics* 49 (2): 268–72.

Kiai, Ava. 2019. "To Protect Credibility in Science, Banish 'Publish or Perish'." *Nature Human Behaviour* 3 (10): 1017–18.

King, Gary. 1995. "Replication, Replication." *PS: Political Science & Politics* 28 (3): 444–52.

Kolmogorov, Andrei Nikolaevich, and Albert T Bharucha-Reid. 2018. *Foundations of the Theory of Probability: Second English Edition*. Courier Dover Publications.

Konigsberg, Lyle W, and Susan R Frankenberg. 2013. "Bayes in Biological Anthropology." *American Journal of Physical Anthropology* 152: 153–84.

Kratz, Fabian, Bettina Pettinger, and Michael Grätz. 2022. "At Which Age Is Education the Great Equalizer? A Causal Mediation Analysis of the (in-) Direct Effects of Social Origin over the Life Course." *European Sociological Review* 38 (6): 866–81.

Kruschke, John K. 2012. "Graphical Model Diagrams in Doing Bayesian Data Analysis Versus Traditional Convention." *Doing Bayesian Data Analysis*. https://doingbayesiandataanalysis.blogspot.com/2012/05/graphical-model-diagrams-in-doing.html?showComment=1354757045570#c4747595504523148906.

———. 2014. "Doing Bayesian Data Analysis: A Tutorial with r, JAGS, and Stan."

———. 2021. "Bayesian Analysis Reporting Guidelines." *Nature Human Behaviour*, 1–10.

Kruschke, John K, Herman Aguinis, and Harry Joo. 2012. "The Time Has Come: Bayesian Methods for Data Analysis in the Organizational Sciences." *Organizational Research Methods* 15 (4): 722–52.

Kuhn, Thomas S. 1970. *The Structure of Scientific Revolutions*. Vol. 111. Chicago University of Chicago Press.

Kumar, Ravin, Colin Carroll, Ari Hartikainen, and Martin Osvaldo. 2019. "ArviZ a unified library for exploratory analysis of Bayesian models in Python." *Journal of Open Source Software*. https://doi.org/10.21105/joss.01143.

Lanfear, Charles C. 2022. "Collective Efficacy and the Built Environment." *Criminology* 60 (2): 370–96.

Laplace, Pierre Simon de. 1820. *Theorie Analytique Des Probabilites*. Vol. 7. Courcier.

Lattimore, Finnian, and David Rohde. 2019a. "Causal Inference with Bayes Rule." *arXiv Preprint arXiv:1910.01510*.

———. 2019b. "Replacing the Do-Calculus with Bayes Rule." *arXiv Preprint arXiv:1906.07125*.

Lee, Chelsea. 2012. "Citing a Whole Periodical." APA Style Blog. 2012. https://blog.apastyle.org/apastyle/2012/09/citing-a-whole-periodical.html.

Legewie, Joscha, and Nino José Cricco. 2022. "Long-Term Exposure to Neighborhood Policing and the Racial/Ethnic Gap in High School Graduation." *Demography* 59 (5): 1739–61.

Lieberson, Stanley. 1985. *Making It Count: The Improvement of Social Research and Theory*. Univ of California Press.

Lundberg, Ian, Rebecca Johnson, and Brandon M Stewart. 2021. "What Is Your Estimand? Defining the Target Quantity Connects Statistical Evidence to Theory." *American Sociological Review* 86 (3): 532–65.

Lynch, Scott M, and Bryce Bartlett. 2019. "Bayesian Statistics in Sociology: Past, Present, and Future." *Annual Review of Sociology* 45: 47–68.

Malinsky, Daniel, Ilya Shpitser, and Thomas Richardson. 2019. "A Potential Outcomes Calculus for Identifying Conditional Path-Specific Effects." In *The 22nd International Conference on Artificial Intelligence and Statistics*, 3080–88. PMLR.

McElreath, Richard. 2020. *Statistical Rethinking: A Bayesian Course with Examples in r and STAN*. Chapman and Hall.

———. 2021a. "Regression, Fire, and Dangerous Things (1/3)." Elements of Evolutionary Anthropology. 2021. https://elevanth.org/blog/2021/06/15/regression-fire-and-dangerous-things-1-3/.

———. 2021b. *causal salad 2021*. https://github.com/rmcelreath/causal_salad_2021.

———. 2023. "Statistical Rethinking 2023 - 20 - Horoscopes." YouTube. 2023. https://www.youtube.com/watch?v=qwF-st2NGTU&list=PLDcUM9US4XdPz-KxHM4XHt7uUVGWWVSus&index=21.

McGrayne, Sharon Bertsch. 2011. *The Theory That Would Not Die*. Yale University Press.

McLevey, John. 2019. "Epistemic and Evidential Cultures." *Sage Research Methods Foundations. London: Sage. Https://Doi.Org/10.4135/9781526421036840315*.

———. 2021. *Doing Computational Social Science: A Practical Introduction*. Sage.

McLevey, John, Pierson Browne, and Tyler Crick. 2021. "Reproducibility and Principled Data Processing." In *Handbook of Computational Social Science, Volume 2*, 108–24. Routledge.

McShane, Blakeley B, David Gal, Andrew Gelman, Christian Robert, and Jennifer L Tackett. 2019. "Abandon Statistical Significance." *The American Statistician* 73 (sup1): 235–45.

Meng, Xiao-Li. 1994. "Posterior Predictive $p$-Values." *The Annals of Statistics* 22 (3): 1142–60.

Menzies, Peter, and Huw Price. 1993. "Causation as a Secondary Quality." *The British Journal for the Philosophy of Science* 44 (2): 187–203.

Mintzberg, Henry, and Alexandra McHugh. 1985. "Strategy Formation in an Adhocracy." *Administrative Science Quarterly*, 160–97.

Morgan, Stephen L, and Christopher Winship. 2015. *Counterfactuals and Causal Inference*. Cambridge University Press.

Muthen, Bengt O. 1987. "Response to Freedman's Critique of Path Analysis: Improve Credibility by Better Methodological Training." *Journal of Educational Statistics* 12 (2): 178–84.

Muthén, Bengt. 1984. "A General Structural Equation Model with Dichotomous, Ordered Categorical, and Continuous Latent Variable Indicators." *Psychometrika* 49 (1): 115–32.

Natanegara, Fanni, Beat Neuenschwander, John W Seaman Jr, Nelson Kinnersley, Cory R Heilmann, David Ohlssen, and George Rochester. 2014. "The Current State of Bayesian Methods in Medical Product Development: Survey Results and Recommendations from the DIA Bayesian Scientific Working Group." *Pharmaceutical Statistics* 13 (1): 3–12.

Neyman, Jerzy. 1923. "On the Application of Probability Theory to Agricultural Experiments. Essay on Principles." *Ann. Agricultural Sciences*, 1–51.

Neyman, Jerzy, and Egon Sharpe Pearson. 1933. "IX. On the Problem of the Most Efficient Tests of Statistical Hypotheses." *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 231 (694-706): 289–337.

Nosek, Brian A, Charles R Ebersole, Alexander C DeHaven, and David T Mellor. 2018. "The Preregistration Revolution." *Proceedings of the National Academy of Sciences* 115 (11): 2600–2606.

O'Neil, Cathy. 2016. *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. Broadway books.

Otárola-Castillo, Erik, and Melissa G Torquato. 2018. "Bayesian Statistics in Archaeology." *Annual Review of Anthropology* 47: 435–53.

Pashler, Harold, and Eric–Jan Wagenmakers. 2012. "Editors' Introduction to the Special Section on Replicability in Psychological Science: A Crisis of Confidence?" *Perspectives on Psychological Science* 7 (6): 528–30.

Pearl, Judea. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan kaufmann.

———. 1993. "[Bayesian Analysis in Expert Systems]: Comment: Graphical Models, Causality and Intervention." *Statistical Science* 8 (3): 266–69.

———. 1994. "A Probabilistic Calculus of Actions." In *Uncertainty Proceedings 1994*, 454–62. Elsevier.

———. 1995. "Causal Diagrams for Empirical Research." *Biometrika* 82 (4): 669–88.

———. 2009a. "Causal Inference in Statistics: An Overview." *Statistics Surveys* 3: 96–146.

———. 2009b. *Causality*. Cambridge university press.

Pearl, Judea, Madelyn Glymour, and Nicholas P Jewell. 2016. *Causal Inference in Statistics: A Primer*. John Wiley & Sons.

Pearl, Judea, and Dana Mackenzie. 2018. *The Book of Why: The New Science of Cause and Effect*. Basic books.

Pearl, Judea, and Thomas S Verma. 1995. "A Theory of Inferred Causation." In *Studies in Logic and the Foundations of Mathematics*, 134:789–811. Elsevier.

Pearson, Egon S. 1962. "Some Thoughts on Statistical Inference." *The Annals of Mathematical Statistics* 33 (2): 394–403.

Peixoto, Tiago P. 2021. "Descriptive Vs. Inferential Community Detection: Pitfalls, Myths and Half-Truths." *arXiv Preprint arXiv:2112.00183*.

Pennycook, Gordon, and Valerie A Thompson. 2016. "Base-Rate Neglect." In *Cognitive Illusions*, 44–61. Psychology Press.

Perezgonzalez, Jose D. 2015. "Fisher, Neyman-Pearson or NHST? A Tutorial for Teaching Data Testing." *Frontiers in Psychology* 6: 223.

Popper, Karl. 2009. "Science: Conjectures and Refutations." *The. Philosophy of Science: An Historical Anthology. Oxford: Wiley*, 471–88.

Pridemore, William Alex, Matthew C Makel, and Jonathan A Plucker. 2018. "Replication in Criminology and the Social Sciences." *Annual Review of Criminology* 1: 19–38.

Ragin, Charles C. 1987. *The Comparative Method: Moving Beyond Qualitative and Quantitative Strategies*. Univ of California Press.

Rogers, Everett M. 1976. "New Product Adoption and Diffusion." *Journal of Consumer Research* 2 (4): 290–301.

Rosenbaum, Paul R, and Donald B Rubin. 1983. "The Central Role of the Propensity Score in Observational Studies for Causal Effects." *Biometrika* 70 (1): 41–55.

Rubin, Donald B. 1974. "Estimating Causal Effects of Treatments in Randomized and Nonrandomized Studies." *Journal of Educational Psychology* 66 (5): 688.

———. 1977. "Assignment to Treatment Group on the Basis of a Covariate." *Journal of Educational Statistics* 2 (1): 1–26.

———. 1978. "Bayesian Inference for Causal Effects: The Role of Randomization." *The Annals of Statistics*, 34–58.

———. 1980a. "Bias Reduction Using Mahalanobis-Metric Matching." *Biometrics*, 293–98.

———. 1980b. "Randomization Analysis of Experimental Data: The Fisher Randomization Test Comment." *Journal of the American Statistical Association* 75 (371): 591–93.

———. 1981. "Estimation in Parallel Randomized Experiments." *Journal of Educational Statistics* 6 (4): 377–401.

———. 1986. "Statistics and Causal Inference: Comment: Which Ifs Have Causal Answers." *Journal of the American Statistical Association* 81 (396): 961–62.

———. 1990a. "Comment: Neyman (1923) and Causal Inference in Experiments and Observational Studies." *Statistical Science* 5 (4): 472–80.

———. 1990b. "Formal Mode of Statistical Inference for Causal Effects." *Journal of Statistical Planning and Inference* 25 (3): 279–92.

———. 2004a. "Direct and Indirect Causal Effects via Potential Outcomes." *Scandinavian Journal of Statistics* 31 (2): 161–70.

———. 2004b. *Multiple Imputation for Nonresponse in Surveys*. Vol. 81. John Wiley & Sons.

———. 2005. "Causal Inference Using Potential Outcomes: Design, Modeling, Decisions." *Journal of the American Statistical Association* 100 (469): 322–31.

———. 2008. "For Objective Causal Inference, Design Trumps Analysis."

Salganik, Matthew J. 2019. *Bit by Bit: Social Research in the Digital Age*. Princeton University Press.

Salmon, Wesley C. 1998. *Causality and Explanation*. Oxford University Press.

Salvatier, John, Thomas V. Wiecki, and Christopher Fonnesbeck. 2016. "Probabilistic Programming in Python Using PyMC3." *PeerJ Computer Science* 2 (April): e55. https://doi.org/10.7717/peerj-cs.55.

Schneider, Tinu, and Mark James Adams. 2013. *Diagram for hierarchical models* (version 1.0). https://github.com/tinu-schneider/DBDA_hierach_diagram.

Schölkopf, Bernhard. 2022. "Causality for Machine Learning." In *Probabilistic and Causal Inference: The Works of Judea Pearl*, 765–804.

Schooler, Jonathan W. 2014. "Metascience Could Rescue the 'Replication Crisis'." *Nature* 515 (7525): 9–9.

Sekhon, Jasjeet. 2008. "The Neyman—Rubin Model of Causal Inference and Estimation via Matching Methods."

Severini, Thomas A. 1993. "Bayesian Interval Estimates Which Are Also Confidence Intervals." *Journal of the Royal Statistical Society: Series B (Methodological)* 55 (2): 533–40.

Sharma, Amit, and Emre Kiciman. 2020. "DoWhy: An End-to-End Library for Causal Inference." *arXiv Preprint arXiv:2011.04216*.

Shpitser, Ilya, and Judea Pearl. 2006. "Identification of Joint Interventional Distributions in Recursive Semi-Markovian Causal Models." In *Proceedings of the National Conference on Artificial Intelligence*, 21:1219. 2. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Simon, Herbert A. 1954. "Spurious Correlation: A Causal Interpretation." *Journal of the American Statistical Association* 49 (267): 467–79.

Simpson, Edward H. 1951. "The Interpretation of Interaction in Contingency Tables." *Journal of the Royal Statistical Society: Series B (Methodological)* 13 (2): 238–41.

Stokes, Allyson, Janice Aurini, Rob Gorbet, and John McLevey. 2022. "Using Robotics to Support the Acquisition of 21st Century Competencies: Promising (and Practical) Directions." *Canadian Journal of Education*.

Sung, Lillian, Jill Hayden, Mark L Greenberg, Gideon Koren, Brian M Feldman, and George A Tomlinson. 2005. "Seven Items Were Identified for Inclusion When Reporting a Bayesian Analysis of a Clinical Study." *Journal of Clinical Epidemiology* 58 (3): 261–68.

Szucs, Denes, and John Ioannidis. 2017. "When Null Hypothesis Significance Testing Is Unsuitable for Research: A Reassessment." *Frontiers in Human Neuroscience* 11: 390.

Textor, Johannes, Benito van der Zander, Mark S Gilthorpe, Maciej Liśkiewicz, and George TH Ellison. 2016. "Robust Causal Inference Using Directed Acyclic Graphs: The r Package 'Dagitty'." *International Journal of Epidemiology* 45 (6): 1887–94. https://doi.org/10.1093/ije/dyw341.

Van Dalen, Hendrik P. 2021. "How the Publish-or-Perish Principle Divides a Science: The Case of Economists." *Scientometrics* 126 (2): 1675–94.

VanderWeele, Tyler J, and Miguel A Hernan. 2013. "Causal Inference Under Multiple Versions of Treatment." *Journal of Causal Inference* 1 (1): 1–20.

Von Wright, Georg Henrik. 1968. "An Essay in Deontic Logic and the General Theory of Action: With a Bibliography of Deontic and Imperative Logic."

Wagenmakers, Eric-Jan, Maarten Marsman, Tahira Jamil, Alexander Ly, Josine Verhagen, Jonathon Love, Ravi Selker, et al. 2018. "Bayesian Inference for Psychology. Part i: Theoretical Advantages and Practical Ramifications." *Psychonomic Bulletin & Review* 25 (1): 35–57.

Waltman, Ludo, and Nees Jan Van Eck. 2012. "The Inconsistency of the h-Index." *Journal of the American Society for Information Science and Technology* 63 (2): 406–15.

Westera, Wim. 2021. "Comparing Bayesian Statistics and Frequentist Statistics in Serious Games Research." *International Journal of Serious Games* 8 (1): 27–44.

Wiggins, Bradford J, and Cody D Christopherson. 2019. "The Replication Crisis in Psychology: An Overview for Theoretical and Philosophical Psychology." *Journal of Theoretical and Philosophical Psychology* 39 (4): 202.

Wodtke, Geoffrey T, Sagi Ramaj, and Jared Schachner. 2022. "Toxic Neighborhoods: The Effects of Concentrated Poverty and Environmental Lead Contamination on Early Childhood Development." *Demography* 59 (4): 1275–98.

Wright, Sewall. 1920. "The Relative Importance of Heredity and Environment in Determining the Piebald Pattern of Guinea-Pigs." *Proceedings of the National Academy of Sciences* 6 (6): 320–32.

———. 1921. "Correlation and Causation."

———. 1934. "The Method of Path Coefficients." *The Annals of Mathematical Statistics* 5 (3): 161–215.

Wuchty, Stefan, Benjamin F Jones, and Brian Uzzi. 2007. "The Increasing Dominance of Teams in Production of Knowledge." *Science* 316 (5827): 1036–39.

Wunsch, Guillaume, Federica Russo, and Michel Mouchart. 2010. "Do We Necessarily Need Longitudinal Data to Infer Causal Relations?" *Bulletin of Sociological Methodology/Bulletin de Méthodologie Sociologique* 106 (1): 5–18.

# Appendix A – Results from Review of Causal Methods

Overall classification of corpus:

| Effect Estimate | Count |
|---|---|
| No Effect Estimate | 291 |
| Frequentist | 279 |
| Bayesian | 4 |

The causal strategies I identified are tabulated below:

| Strategy | Count |
|---|---|
| Experiment | 22 |
| SEM | 17 |
| Quasi-Experiment | 16 |
| Panel Data | 13 |
| Instrumental Variable | 11 |
| Natural Experiment | 9 |
| Potential Outcomes | 7 |
| Propensity Score | 5 |
| Path-Blocking | 2 |

## Causal Classification by Journal

### American Journal of Sociology

| Causal? | Count |
|---|---|
| Explicitly Non-Causal | 2 |
| Ambiguous | 11 |
| Limited | 2 |
| Explicitly Causal | 2 |
| No Effect Estimate (Not Reviewed) | 18 |

### American Sociological Review

| Causal? | Count |
|---|---|
| Explicitly Non-Causal | 5 |
| Ambiguous | 5 |
| Limited | 3 |
| Explicitly Causal | 10 |
| No Effect Estimate (Not Reviewed) | 13 |

**Annual Review of Sociology**

| Causal? | Count |
|---|---|
| Explicitly Non-Causal | 0 |
| Ambiguous | 0 |
| Limited | 0 |
| Explicitly Causal | 0 |
| No Effect Estimate (Not Reviewed) | 24 |

**Canadian Journal of Sociology**

| Causal? | Count |
|---|---|
| Explicitly Non-Causal | 0 |
| Ambiguous | 2 |
| Limited | 0 |
| Explicitly Causal | 0 |
| No Effect Estimate (Not Reviewed) | 7 |

**Canadian Review of Sociology and Anthropology**

| Causal? | Count |
|---|---|
| Explicitly Non-Causal | 5 |
| Ambiguous | 7 |
| Limited | 2 |
| Explicitly Causal | 0 |
| No Effect Estimate (Not Reviewed) | 26 |

**Criminology**

| Causal? | Count |
|---|---|
| Explicitly Non-Causal | 5 |
| Ambiguous | 5 |
| Limited | 3 |
| Explicitly Causal | 3 |
| No Effect Estimate (Not Reviewed) | 8 |

**Demography**

| Causal? | Count |
|---|---|
| Explicitly Non-Causal | 15 |
| Ambiguous | 35 |
| Limited | 7 |
| Explicitly Causal | 12 |
| No Effect Estimate (Not Reviewed) | 18 |

**European Sociological Review**

| Causal? | Count |
|---|---|
| Explicitly Non-Causal | 16 |
| Ambiguous | 24 |
| Limited | 3 |
| Explicitly Causal | 8 |
| No Effect Estimate (Not Reviewed) | 3 |

**Information Communication and Society**

| Causal? | Count |
|---|---|
| Explicitly Non-Causal | 12 |
| Ambiguous | 20 |
| Limited | 2 |
| Explicitly Causal | 3 |
| No Effect Estimate (Not Reviewed) | 96 |

**Journal of Marriage and the Family**

| Causal? | Count |
|---|---|
| Explicitly Non-Causal | 14 |
| Ambiguous | 29 |
| Limited | 3 |
| Explicitly Causal | 5 |
| No Effect Estimate (Not Reviewed) | 17 |

**Sociology**

| Causal? | Count |
|---|---|
| Explicitly Non-Causal | 0 |
| Ambiguous | 1 |
| Limited | 2 |
| Explicitly Causal | 0 |
| No Effect Estimate (Not Reviewed) | 61 |

# Appendix B – Code for Example Simulations in Chapter 2

## two_schools.py

```
from scipy import stats
import pandas as pd

N = 300

W = stats.bernoulli(p=11/30)
R = stats.bernoulli(p=5/19)
RW = stats.bernoulli(p=10/11)

Y =  stats.bernoulli(p=21/140)
YR = stats.bernoulli(p=3/50)
YW = stats.bernoulli(p=9/10)
YRW =stats.bernoulli(p=72/100)


# Springfield
series_list = []

for i in range(N):
    w = int(W.rvs(1))

    if w:
        r = int(RW.rvs(1))
    else:
        r = int(R.rvs(1))

    if r and w:
        y = int(YRW.rvs(1))
    elif r:
        y = int(YR.rvs(1))
    elif w:
        y = int(YW.rvs(1))
    else:
        y = int(Y.rvs(1))


    series_list.append((pd.Series({'y':y, 'w':w, 'r':r})))
```

```
df = pd.concat(series_list, axis=1, ignore_index=True).T


r1tot = len(df[df['r'] == 1])
r0tot = N - r1tot

r0y1 = sum(df[df['r'] == 0]['y'])
r1y1 = sum(df[df['r'] == 1]['y'])


print(" ")
print("Springfield Unstratified: ")
print(f"R=0: {r0y1} in {r0tot} met threshold ({round(100*r0y1/r0tot)}%)")
print(f"R=1: {r1y1} in {r1tot} met threshold ({round(100*r1y1/r1tot)}%)")
print(" ")

r0w0df = df.query("r==0 and w==0")
r0w0tot = len(r0w0df)
r0w0y1 = sum(r0w0df['y'])

r0w1df = df.query("r==0 and w==1")
r0w1tot = len(r0w1df)
r0w1y1 = sum(r0w1df['y'])

r1w1df = df.query("r==1 and w==1")
r1w1tot = len(r1w1df)
r1w1y1 = sum(r1w1df['y'])

r1w0df = df.query("r==1 and w==0")
r1w0tot = len(r1w0df)
r1w0y1 = sum(r1w0df['y'])



print(" ")
print("Springfield Stratified:       W=0                                    W=1")
print(f"R=0: {r0w0y1} in {r0w0tot} met threshold ({round(100*r0w0y1/r0w0tot)}%)," +
      f"        {r0w1y1} in {r0w1tot} met threshold ({round(100*r0w1y1/r0w1tot)}%)")
print(f"R=1: {r1w0y1} in {r1w0tot} met threshold ({round(100*r1w0y1/r1w0tot)}%),"+
      f"           {r1w1y1} in {r1w1tot} met threshold ({round(100*r1w1y1/r1w1tot)}%)")



# Shelbyville
series_list = []
RAND = stats.bernoulli(p=0.5)

for i in range(N):
    w = int(W.rvs(1))
    r = int(RAND.rvs(1))

    if r and w:
        y = int(YRW.rvs(1))
```

```
    elif r:
        y = int(YR.rvs(1))
    elif w:
        y = int(YW.rvs(1))
    else:
        y = int(Y.rvs(1))


    series_list.append((pd.Series({'y':y, 'w':w, 'r':r})))

df = pd.concat(series_list, axis=1, ignore_index=True).T


r1tot = len(df[df['r'] == 1])
r0tot = N - r1tot

r0y1 = sum(df[df['r'] == 0]['y'])
r1y1 = sum(df[df['r'] == 1]['y'])


print(" ")
print("Shelbyville Experiment: ")
print(f"R=0: {r0y1} in {r0tot} met threshold ({round(100*r0y1/r0tot)}%)")
print(f"R=1: {r1y1} in {r1tot} met threshold ({round(100*r1y1/r1tot)}%)")
print(" ")
```

# Appendix C – Python code for ‘pdpp‘

The code contained in this appendix is a full and true copy[176] of the non-empty Python code files for `pdpp` (version `0.4.5`) as indexed in PYPI at the time of this dissertation's publication.[177]

Each of the following code blocks contains the code from one of the modules in the `pdpp` package.

## `main.py`

```
"""The primary access point for the `pdpp` package.

Every call to `pdpp` from the command line is routed through this
module.
"""

from pdpp.utils.task_directory_test import TaskDirectory
from pdpp.utils.directory_test import in_project_directory
from pdpp.utils.rem_slash import rem_slash
from pdpp.styles.graph_style import default_graph_style,
greyscale_graph_style, base_graph_style
import os
import click
from typing import List, Type


GRAPH_STYLE_LIST: List[Type[base_graph_style]] = [default_graph_style,
                                                  greyscale_graph_style]
CONTEXT_SETTINGS = dict(help_option_names=['-h', '--help'])
FILES_LIST = ['png', 'pdf', 'jpg']


@click.group()
def main() -> None:
    """A command line tool to automate the use of the Principled Data Processing
     methodology for reproducibility."""
    pass


@main.command(short_help="Prepares a directory to become a pdpp project")
def init() -> None:
    from pdpp.templates.populate_new_project import populate_new_project
```

[176] Some code has been reformatted to better fit onto the printed page; the code is otherwise unaltered.
[177] The package's non-Python files are not included here, as they do not in any way contribute to or alter the package's behaviour.

```
    populate_new_project()


@main.command(short_help="""Create a new task directory
""",
              context_settings=CONTEXT_SETTINGS)
@click.option('--dirname', '-d',
              type=TaskDirectory(),
              prompt="What do you want to call the new task directory?"
                     "(Use all lower case, no spaces, and cannot be"
                     "'_import_' or '_export_')",
              help="This is what you want to name your new task directory. "
                   "It must be all lower case and contain no spaces."
              )
def new(dirname):
    """Creates a new directory named <dirname>,
    with subdirectories 'input', 'output', and 'src'.
    Adds dodo.py to <dirname>. Use this to create a new task directory."""
    in_project_directory()
    from pdpp.tasks.standard_task import StandardTask
    StandardTask(target_dir = rem_slash(dirname)).initialize_task()
    click.echo(f"Your new task directory, {dirname}, was created.")


@main.command(short_help="""Configure a task's dependencies and source code
""")
def rig():
    in_project_directory()
    from pdpp.questions.question_0 import q0
    q0().rig_task()


@main.command(short_help="""Create a new task directory with no automation assumptions
""",)
@click.option('--dirname', '-s',
              type=TaskDirectory(),
              prompt="What do you want to call the new custom task directory?"
                     "(Use all lower case, no spaces, "
                     "and cannot be '_import_' or '_export_')",
              help="This is what you want to name your new task directory. "
                   "It must be all lower case and contain no spaces."
              )
def custom(dirname: str):
    in_project_directory()
    from pdpp.tasks.custom_task import CustomTask
    CustomTask(target_dir = rem_slash(dirname)).initialize_task()
    click.echo(f"Your new task directory, {dirname}, was created.")


@main.command(short_help="""Create a new sub-project directory
""",)
@click.option('--dirname', '-s',
              type=TaskDirectory(),
```

```
              prompt="What do you want to call the new subproject task directory?"
                     "(Use all lower case, no spaces, "
                     "and cannot be '_import_' or '_export_')",
              help="This is what you want to name your new task directory. "
                   "It must be all lower case and contain no spaces."
              )
def sub(dirname):
    """Creates a new subproject in a directory named <dirname>.
    Adds dodo.py to <dirname>. Use this to create a new task directory."""
    in_project_directory()
    from pdpp.tasks.sub_task import SubTask
    SubTask(target_dir = rem_slash(dirname)).initialize_task()
    click.echo(f"Your new subproject, {dirname}, was created.")


# graph
@main.command(short_help="""Graph this project's dependency structure
""", context_settings=CONTEXT_SETTINGS)
@click.option('--files', '-f',
              type=click.Choice(FILES_LIST),
              prompt="What file format would you prefer as an output?",
              help="The dependency graph can be outputted in "
              ".png and/or .pdf formats. Default is to output png.",
              default="png")
@click.option(
    '--style', '-s',
    type=click.Choice([s.NAME for s in GRAPH_STYLE_LIST]),
    prompt="What color scheme would you prefer?",
    help="The dependency graph can be outputted in one of a variety of styles.",
    default=default_graph_style.NAME,
    )
def graph(files, style):
    in_project_directory()
    from pdpp.utils.graph_dependencies import depgraph
    """Creates a dependency graph to visualize how the tasks in
    your project relate to each other."""
    full_style = next((s for s in GRAPH_STYLE_LIST if s.NAME == style), None)
    depgraph(files, full_style)


# run
@main.command(short_help="""Execute the project's tasks""")
def run():
    in_project_directory()
    from pdpp.automation.doit_run import doit_run
    doit_run()


# enable
@main.command(short_help="Selects which tasks will resolve when the project is run")
def enable():
    in_project_directory()
    from pdpp.automation.task_enabler import task_enabler
```

```
    task_enabler()

#extant
@main.command(short_help="Add standard automation to an unautomated directory")
def extant():
    in_project_directory()
    from pdpp.questions.question_extant import q_extant
    q_extant().rig_task()
```

## automation/doit_run.py

```
from pdpp.automation.task_creator import gen_many_tasks, task_all

def doit_run():
    import doit
    doit.run(globals())

if __name__ == '__main__':
    doit_run()
```

## automation/link_task.py

```
from pdpp.tasks.base_task import BaseTask
from pdpp.automation.mylinker import file_linker, dir_linker
from typing import List
from posixpath import join
import os


def make_link_task(task: BaseTask, disabled_list: List[str], final_dep_list: List):

    for task_with_dependency, dependency_metadata in task.dep_files.items():

        link_action_list = []
        link_dep_list = []
        link_targ_list = []

        if task_with_dependency not in disabled_list:

            file_link_start = [join(task_with_dependency,
            dependency_metadata.task_out, f) for f in dependency_metadata.file_list]
            file_link_end = [join(
            task.target_dir, task.IN_DIR, f) for f in dependency_metadata.file_list]

            link_action_list.extend(
            [(file_linker, [fls, fle]) for fls, fle in list(
            zip(file_link_start, file_link_end))])

            dir_link_start = [join(
            task_with_dependency,
            dependency_metadata.task_out, f) for f in dependency_metadata.dir_list]
```

```
            dir_link_end = [join(
            task.target_dir, task.IN_DIR, f) for f in dependency_metadata.dir_list]

            link_action_list.extend(
            [(dir_linker, [dls, dle]) for dls, dle in list(
            zip(dir_link_start, dir_link_end))])

            link_dep_list.extend(file_link_start)
            link_targ_list.extend(file_link_end)

            for dir_dependency in dependency_metadata.dir_list:
                path_to_dep_dir = join(
                dependency_metadata.task_name, dependency_metadata.task_out)
                startdir = os.getcwd()
                os.chdir(path_to_dep_dir)
                for root, _, filenames in os.walk(dir_dependency):
                    for filename in filenames:

                        subdir_filepath_start = join(
                        dependency_metadata.task_name,
                        dependency_metadata.task_out, root, filename)
                        link_dep_list.append(subdir_filepath_start)

                        subdir_filepath_end = join(
                        task.target_dir, task.IN_DIR, root, filename)
                        link_targ_list.append(subdir_filepath_end)
                os.chdir(startdir)

            final_dep_list.extend(link_targ_list)

            yield {
                'basename': '_task_{}_LINK_TO_{}'.format(
                task_with_dependency, task.target_dir),
                'actions': link_action_list,
                'file_dep': link_dep_list,
                'targets': link_targ_list,
                'clean': True,
            }
```

## automation/mylinker.py

```
from os import link, remove
from shutil import rmtree, copytree


def file_linker(link_start, link_end):
    try:
        link(link_start, link_end)
    except FileExistsError:
        remove(link_end)
        link(link_start, link_end)
```

```
def dir_linker(link_start, link_end):
    try:
        copytree(link_start, link_end, copy_function=link)
    except FileExistsError:
        rmtree(link_end)
        copytree(link_start, link_end, copy_function=link)
```

## automation/task_creator.py

```
from pdpp.utils.directory_test import get_pdpp_tasks
from typing import List
from pdpp.tasks.base_task import BaseTask
from pdpp.automation.link_task import make_link_task


def find_dependencies_from_others(
                                task: BaseTask,
                                loaded_tasks: List[BaseTask]
                                ) -> List[str]:
    dependencies = []

    for other_task in loaded_tasks:
        dependencies.extend(other_task.provide_dependencies(task))

    return [d for d in set(dependencies)] # This is just a hackier way of unpacking
    # everything; pylance type-checking REALLY didn't like list(set(dependencies))
    # for some reason

def gen_many_tasks():
    # 1. Get all of the tasks in the current scope
    loaded_tasks = get_pdpp_tasks()

    # 2. Create a list of all the disabled tasks loaded in task #1
    disabled_list = [t.target_dir for t in loaded_tasks if not t.enabled]

    for task in loaded_tasks:

        target_list = find_dependencies_from_others(task, loaded_tasks)

        # Although it's hacky, 'dep_list' is altered inside of 'make_link_task'.
        # Doit places strict requirements on the format used by
        # task information, and the use of the nested 'yield' statements
        # make it difficult to get data out of the 'make_link_task' function
        # in a pythonic way. May the most exalted one - our BDFL - take pity upon me.

        dep_list = task.provide_src_dependencies()

        yield make_link_task(task, disabled_list, dep_list)

        if task.enabled and task.RUN_VALID:

            yield {
```

```
                'basename': f'{task.target_dir}',
                'actions': task.provide_run_actions(),
                'file_dep': dep_list,
                'targets': target_list,
                'clean': True,
            }


def task_all():
    yield gen_many_tasks()

if __name__ == '__main__':
    import doit
    doit.run(globals())
```

## automation/task_enabler.py

```
from pdpp.utils.directory_test import get_runnable_tasks
from pdpp.styles.prompt_style import custom_style_fancy
from questionary import prompt, Choice


def task_enabler():

    runnable_tasks = get_runnable_tasks()

    choice_list = []

    for task in runnable_tasks:
        choice_list.append(Choice(
            title=task.target_dir,
            value=task,
            checked=task.enabled,

        ))

    questions_1 = [
        {
            'type': 'checkbox',
            'message': "Select the tasks which will be run when 'pdpp run' is called",
            'name': 'enabled',
            'choices': choice_list,
        }
    ]

    try:
        enabled_list = prompt(questions_1, style=custom_style_fancy)['enabled']
    except IndexError:
        print('There are no valid tasks in this project directory!')
        enabled_list = []

    for task in runnable_tasks:
```

```
        if task in enabled_list:
            task.enable()
        elif task not in enabled_list:
            task.disable()
        else:
            raise Exception("SOMETHING WENT WRONG WITH ENABLE FLOW CONTROL")
```

## languages/extension_parser.py

```
from pdpp.tasks.base_task import BaseTask
from pdpp.languages.language_enum import Language
from typing import List


def extension_parser(task: BaseTask) -> List[str]:
    extension_list = []

    for entry in task.src_files:
        extension_list.append(str(entry).split('.')[-1].lower())

    language_list = []

    for entry in extension_list:
        if entry == 'py':
            language_list.append(Language.PYTHON.value)
        elif entry == 'r' or entry == 'rscript':
            language_list.append(Language.R.value)
        else:
            language_list.append('???')

    return language_list
```

## languages/language_enum.py

```
from enum import Enum

class Language(Enum):
    PYTHON = 'Python'
    R = "R"
    NULL = ""
```

## languages/language_parser.py

```
from pdpp.tasks.base_task import BaseTask
from pdpp.languages.runners import python_runner, r_runner
from pdpp.languages.language_enum import Language
```

```
def parse_language(task: BaseTask):

    if task.language == Language.PYTHON.value:
        return python_runner
    elif task.language == Language.R.value:
        return r_runner
    else:
        raise Exception
```

## languages/runners.py

```
from os import name
from subprocess import run
from posixpath import join
from pdpp.tasks.base_task import BaseTask

def python_runner(script_name: str, task: BaseTask):
    if name == 'posix':
        python_caller = 'python3'
    else:
        python_caller = 'python'
    run([python_caller, script_name], check=True, cwd=join(
        task.target_dir, task.SRC_DIR))

def r_runner(script_name, target_dir, src_dir):
    run(['Rscript', script_name], check=True, cwd=join(target_dir, src_dir))

# TODO: Fix the project runner and bring it in line with the other runners
def project_runner(task: BaseTask):
    run(["doit"], check=True, cwd=task.target_dir)
```

## questions/question_0.py

```
from questionary import prompt, Choice
from click import clear as click_clear
from pdpp.styles.prompt_style import custom_style_fancy
from pdpp.utils.directory_test import get_riggable_tasks
from pdpp.tasks.base_task import BaseTask


def q0() -> BaseTask:
    '''
    This question is used to select the task you wish to alter with pdpp.
    '''

    tasks = get_riggable_tasks()

    click_clear()

    choice_list = []
```

```
    for task in tasks:
        choice_list.append(
            Choice(
                title=task.target_dir,
                value=task,
            )
        )


    questions_0 = [
        {
            'type': 'list',
            'name': 'target_dir',
            'message': 'Select the task you would like to rig:',
            'choices': choice_list
        }
    ]

    return prompt(questions_0, style=custom_style_fancy)['target_dir']
```

## questions/question_1.py

```
from questionary import prompt, Choice
from click import clear as click_clear
from pdpp.styles.prompt_style import custom_style_fancy
from pdpp.tasks.base_task import BaseTask
from typing import List


def q1(dep_tasks: List[BaseTask], task: BaseTask) -> List[BaseTask]:
    """
    This question is used to determine which other tasks in the project structure
    are dependencies of the current task.
    """

    click_clear()

    choice_list = []

    '''
    First, add all the project subdirectories (riggable_subdirectories) returned
    from Question 0.
    When this process encounters the task being rigged (target_dir), add it to
    the list as a disabled entry.
    '''

    for dep_task in dep_tasks:
        if dep_task.target_dir == task.target_dir:
            choice_list.append(
                Choice(
                    title=dep_task.target_dir,
                    value=dep_task,
```

```
                    disabled='This is the selected task'
                )
            )

        else:
            choice_list.append(
                Choice(
                    title=dep_task.target_dir,
                    value=dep_task,
                    checked=dep_task.target_dir in task.dep_files
                )
            )

    if len(choice_list) < 1:
        return []

    questions_1 = [
        {
            'type': 'checkbox',
            'message': 'Select tasks which contain dependencies for'
            ' "{}"'.format(task.target_dir),
            'name': 'dep_tasks',
            'choices': choice_list,
        }
    ]

    return prompt(questions_1, style=custom_style_fancy)['dep_tasks']
```

## questions/question_2.py

```
from questionary import Separator, prompt, Choice
from click import clear as click_clear
from posixpath import join
import os
from pdpp.styles.prompt_style import custom_style_fancy
from pdpp.tasks.base_task import BaseTask
from pdpp.utils.ignorelist import ignorelist
from typing import Tuple, List, Dict
from os import DirEntry
from pdpp.templates.dep_dataclass import dep_dataclass



def q2(
       selected_dep_tasks: List[BaseTask],
       task: BaseTask
       ) -> Dict[str, dep_dataclass]:
    """
    A question which asks users to indicate which individual files
    (drawn from a list of those contained in the output directories
    of the tasks indicated in question #1)
    are required as dependencies for the current task.
```

```
    """

    click_clear()

    q2input: Dict[BaseTask, List[DirEntry[str]]] = {}
    import_input = []

    for selected_task in selected_dep_tasks:

        search_dir = join(selected_task.target_dir, selected_task.OUT_DIR)

        results = [r for r in os.scandir(search_dir) if r.name not in ignorelist]

        if results:
            q2input[selected_task] = results


    choice_list = []

    for key, values in q2input.items():

        choice_list.append(Separator('\n= ' + key.target_dir + ' ='))

        for value in values:
            try:
                checked = value.name in task.dep_files[key.target_dir].dir_list
                 or value.name in task.dep_files[key.target_dir].file_list
            except KeyError:
                checked = False
            except TypeError:
                checked = False

            title = value.name

            if value.is_dir():
                title += " (This is a directory)"

            choice_list.append(
                Choice(
                    title = title,
                    value = (key, value),
                    checked= checked,
                )
            )

    questions_2 =[
        {
            'type': 'checkbox',
            'message': 'Select the dependency files for "{}"'.format(task.target_dir),
            'name': 'dependencies',
            'choices': choice_list,
        }
    ]
```

```
    response_dict = {}

    responses: List[Tuple[BaseTask, DirEntry[str]]]

    if questions_2[0]['choices']:
        responses = prompt(questions_2, style=custom_style_fancy)['dependencies']
    else:
        return {}

    dep_task_set = set([t for t, f in responses])

    for dep_task in dep_task_set:
        file_list = [f.name for t, f in responses if t == dep_task and f.is_file()]
        dir_list = [d.name for t, d in responses if t == dep_task and d.is_dir()]

        response_dict[dep_task.target_dir] = dep_dataclass(
            task_out = dep_task.OUT_DIR,
            task_name = dep_task.target_dir,
            file_list = file_list,
            dir_list = dir_list
        )

    return response_dict
```

## questions/question_3.py

```
from questionary import prompt, Choice
from click import clear as click_clear
from os import scandir, DirEntry
from posixpath import join
from pdpp.styles.prompt_style import custom_style_fancy
from pdpp.utils.ignorelist import ignorelist
from pdpp.tasks.base_task import BaseTask
from typing import List


def q3(task: BaseTask) -> List[str]:
    """
    A question which asks users to indicate which scripts in the chosen task's
    'src'  should be run to produce this task's targets.
    """

    click_clear()

    source_files = []
    source_choices = []
    src_loc = join(task.target_dir, task.SRC_DIR)

    source_files = [s for s in scandir(src_loc) if
        ((s.name not in ignorelist) and (s.is_file()))]

    for entry in source_files:
```

```
        source_choices.append(
            Choice(
                title=entry.name,
                value=entry,
                checked= entry.name in task.src_files
            )
        )

    if len(source_files) < 2:
        return [s.name for s in source_files]

    question_3 = [{
            'type': 'checkbox',
            'message': 'Select the source file(s) for "{}"'.format(task.target_dir),
            'name': 'source',
            'choices': source_choices,
        }]

    final_choices: List[DirEntry[str]] = prompt(
        question_3, style=custom_style_fancy)['source']

    return [s.name for s in final_choices]
```

## questions/question_4.py

```
from questionary import prompt
from click import clear as click_clear
from pdpp.styles.prompt_style import custom_style_fancy
from pdpp.tasks.base_task import BaseTask
from pdpp.languages.language_enum import Language
from pdpp.languages.extension_parser import extension_parser


def q4(task: BaseTask) -> str:
    """
    This question is used to determine the language of the source code this task
    will run.
    """

    click_clear()

    language_list = extension_parser(task)

    if len(set(language_list)) != 1 or '???' in language_list:

        message = ("""Note that pdpp does not currently support automating tasks that
        contain scripts written in more than one programming language.

        If this task contains source code from multiple languages,
        please separate all source code written in dissimilar
        languages into different tasks. Failing to do so will
        cause an exception at runtime.
```

```
        If all of this task's source code is written in the same
        pdpp-supported language, you can indicate the appropriate
        language below.

        If the programming language you would like to use is not
        indicated below, or you would like to use more than one programming
        language in the same task, consider creating a custom pdpp task using
        the 'pdpp custom' command.

        Select the programming language used:
        """)

        question_4 = [{
            'type': 'list',
            'name': 'language',
            'message': message,
            'choices': [
                {
                    'name': Language.PYTHON.value
                },
                {
                    'name': Language.R.value
                }
            ],
        }]

        return prompt(question_4, style=custom_style_fancy)["language"]

    else:
        return language_list[0]
```

## questions/question_extant.py

```
from questionary import prompt, Choice
from click import clear as click_clear
from pdpp.styles.prompt_style import custom_style_fancy
from pdpp.utils.directory_test import get_pdpp_tasks
from pdpp.tasks.standard_task import StandardTask
import os


def q_extant() -> StandardTask:
    '''
    This question is used to select the directory you wish to automate.
    '''

    task_dirs = [d.target_dir for d in get_pdpp_tasks()]

    click_clear()

    dir_list = [
```

```
        d.name for d in os.scandir() if d.name not in task_dirs and d.is_dir()]

    choice_list = []

    print(task_dirs)
    print(dir_list)

    for _dir in dir_list:
        choice_list.append(
            Choice(
                title=_dir,
                value=_dir,
            )
        )


    questions_0 = [
        {
            'type': 'list',
            'name': 'target_dir',
            'message': 'Select the directory you would like to automate:',
            'choices': choice_list
        }
    ]

    dirname = prompt(questions_0, style=custom_style_fancy)['target_dir']


    return StandardTask(target_dir = dirname)
```

## styles/graph_style.py

```
class base_graph_style():
    NAME:str = ''


class default_graph_style(base_graph_style):
    NAME:str = 'default'
    # Task Nodes:
    TASK_NODE_STYLE = 'filled'
    TASK_NODE_SHAPE = 'box3d'
    TASK_NODE_PENWIDTH = 1,
    TASK_NODE_FONTCOLOR = 'black'
    DISABLED_TASK_COLOR = 'dimgrey'
    DISABLED_TASK_FONTCOLOR = TASK_NODE_FONTCOLOR
    DISABLED_TASK_STYLE = 'dashed'
    START_TASK_COLOR = '#3E8DCF'
    END_TASK_COLOR = '#E95C3F'
    MID_TASK_COLOR = '#F2A93B'

    # File Nodes:
    FILE_NODE_STYLE = 'filled'
```

```
    FILE_FILE_SHAPE = 'note'
    FILE_DIR_SHAPE = 'tab'
    FILE_NODE_COLOR = '#bbd698'
    FILE_NODE_PENWIDTH = 1

    # Source Nodes:
    SOURCE_STYLE = 'filled'
    SOURCE_FONT_COLOR = 'black'
    SOURCE_FILL_COLOR = '#9d6edb'
    SOURCE_SHAPE = 'component'
    SOURCE_PENWIDTH = 1

    # Edges:
    EDGE_COLOR = '#828282'
    EDGE_PEN_WIDTH = 1.5


class greyscale_graph_style(base_graph_style):
    NAME:str = 'greyscale'
    # Task Nodes:
    TASK_NODE_STYLE = 'filled'
    TASK_NODE_SHAPE = 'box3d'
    TASK_NODE_PENWIDTH = 1,
    TASK_NODE_FONTCOLOR = 'black'
    DISABLED_TASK_COLOR = 'black'
    DISABLED_TASK_FONTCOLOR = 'black'
    DISABLED_TASK_STYLE = 'dashed'
    START_TASK_COLOR = '#c4c4c4'
    MID_TASK_COLOR = '#b0b0b0'
    END_TASK_COLOR = '#787878'

    # File Nodes:
    FILE_NODE_STYLE = 'filled'
    FILE_FILE_SHAPE = 'note'
    FILE_DIR_SHAPE = 'tab'
    FILE_NODE_COLOR = '#f0f0f0'
    FILE_NODE_PENWIDTH = 1

    # Source Nodes:
    SOURCE_STYLE = 'filled'
    SOURCE_FONT_COLOR = 'black'
    SOURCE_FILL_COLOR = 'white'
    SOURCE_SHAPE = 'component'
    SOURCE_PENWIDTH = 1

    # Edges:
    EDGE_COLOR = '#828282'
    EDGE_PEN_WIDTH = 1.5
```

## styles/prompt_style.py

```
from prompt_toolkit.styles import Style

custom_style_fancy = Style([
    ('qmark', 'fg:#673ab7 bold'),      # token in front of the question
    ('question', 'bold'),              # question text
    ('answer', 'fg:#f44336 bold'),     # submitted answer text behind the question
    ('pointer', 'fg:#673ab7 bold'),    # pointer used in select and checkbox prompts
    ('selected', 'fg:#cc5454'),        # style for a selected item of a checkbox
    ('separator', 'fg:#cc5454'),       # separator in lists
    ('instruction', '')                # user instructions for select, rawselect
])
```

## tasks/base_task.py

```
from typing import Dict, List
from pdpp.templates.dep_dataclass import dep_dataclass
from pdpp.utils.yaml_task import dump_self
from pdpp.utils.execute_at_target import execute_at_target
from pdpp.languages.language_enum import Language


class BaseTask():
    """
    This is a docstring.
    """
    def __init__(
        self,
        target_dir
        ):

        self.target_dir: str
        self.dep_files: Dict[str, dep_dataclass]
        self.src_files: List
        self.language: str = Language.NULL.value
        self.enabled: bool


    FILENAME = ".pdpp_task.yaml"
    RIG_VALID = True # Can be rigged
    TRG_VALID = False # Can have targets
    DEP_VALID = True # Can contain dependencies for other tasks
    SRC_VALID = False # Should soucre code be automatically parsed?
    RUN_VALID = True # Has actions that should be executed at runtime
    IN_DIR: str
    OUT_DIR: str
    SRC_DIR: str

    def provide_run_actions(self) -> List:
        return []
```

```
    def provide_src_dependencies(self) -> List:
        return []

    def provide_dependencies(self, asking_task) -> List[str]:
        try:
            return self.dep_files[asking_task.target_dir].compile_targets()
        except KeyError:
            return []

    def rig_task(self):

        from pdpp.utils.directory_test import get_dependency_tasks
        from pdpp.questions import q1, q2, q3, q4
        from pdpp.utils.immediate_link import immediate_link

        # Ask dependency questions:
        dep_tasks = get_dependency_tasks()
        selected_dep_tasks = q1(dep_tasks, self)
        self.dep_files = q2(selected_dep_tasks, self)

        # Ensure source compliance:
        if self.SRC_VALID:
            self.src_files = q3(self)
            self.language = q4(self)

        # Implement links:
        # TODO: Consider creating a check for files that aren't part of the
        # existing dependency structure; if there are any, ask if users want
        # them deleted or to be preserved.
        immediate_link(self)

        # Finally, save self to YAML:
        #execute_at_target(dump_self, self)
        self.save_self()

    def initialize_task(self):
        raise NotImplementedError

    def save_self(self):
        execute_at_target(dump_self, self)

    def enable(self):
        self.enabled = True
        self.save_self()

    def disable(self):
        self.enabled = False
        self.save_self()
```

## tasks/custom_task.py

```
from pdpp.tasks.base_task import BaseTask
from typing import List, Dict
from os import mkdir
from posixpath import join
from pdpp.utils.execute_at_target import execute_at_target
from pdpp.templates.create_in_out_src import create_in_out_src
from pdpp.languages.language_enum import Language
from pdpp.templates.dep_dataclass import dep_dataclass
from pdpp.languages.runners import project_runner


class CustomTask(BaseTask):
    """
    This is the class documentation
    """
    def __init__(
            self,
            target_dir: str = ""
            ):

        self.target_dir: str = target_dir
        self.dep_files: Dict[str, dep_dataclass] = {}
        self.src_files: List = []
        self.language: str = Language.NULL.value
        self.enabled: bool = True


    RIG_VALID = True # Can be rigged
    TRG_VALID = True # Can have targets
    DEP_VALID = True # Can contain dependencies for other tasks
    SRC_VALID = False # Should soucre code be automatically parsed?
    RUN_VALID = True # Has actions that should be executed at runtime
    IN_DIR = "input"
    OUT_DIR = "output"
    SRC_DIR = "src"

    def provide_run_actions(self) -> List:
        return [(project_runner, [self], {})]

    def provide_src_dependencies(self) -> List:
        return [join(self.target_dir, self.SRC_DIR, s) for s in self.src_files]

    def initialize_task(self):

        # Create directory structure:
        mkdir(self.target_dir)
        with open(join(self.target_dir, 'dodo.py'), 'w') as stream:
            stream.close()
        execute_at_target(create_in_out_src, self)

        # From here on out, rigging is identical to creating anew:
        self.rig_task()
```

## tasks/export_task.py

```
from pdpp.tasks.base_task import BaseTask
from typing import List, Dict
from os import mkdir
from pdpp.utils.execute_at_target import execute_at_target
from pdpp.utils.yaml_task import dump_self
from pdpp.languages.language_enum import Language
from pdpp.templates.dep_dataclass import dep_dataclass


class ExportTask(BaseTask):
    """
    This is the class documentation
    """

    def __init__(
            self,
            target_dir: str = "_export_"
            ):

        self.target_dir: str = "_export_"
        self.dep_files: Dict[str, dep_dataclass] = {}
        self.src_files: List = []
        self.language: str = Language.NULL.value
        self.enabled: bool = True


    RIG_VALID = True # Can be rigged
    TRG_VALID = False # Can have targets
    DEP_VALID = False # Can contain dependencies for other tasks
    SRC_VALID = False # Should soucre code be automatically parsed?
    RUN_VALID = False # Has actions that should be executed at runtime
    IN_DIR = "./"
    OUT_DIR = "./"
    SRC_DIR = "./"

    def initialize_task(self):
        mkdir(self.target_dir)
        execute_at_target(dump_self, self)
```

## tasks/import_task.py

```
from pdpp.tasks.base_task import BaseTask
from typing import List, Dict
from os import mkdir
from pdpp.utils.execute_at_target import execute_at_target
from pdpp.utils.yaml_task import dump_self
from pdpp.languages.language_enum import Language
from pdpp.templates.dep_dataclass import dep_dataclass
```

```
class ImportTask(BaseTask):
    """
    This is the class documentation
    """

    def __init__(
            self,
            target_dir: str = "_import_"
            ):

        self.target_dir: str = "_import_"
        self.dep_files: Dict[str, dep_dataclass] = {}
        self.src_files: List = []
        self.language: str = Language.NULL.value
        self.enabled: bool = True


    RIG_VALID = False # Can be rigged
    TRG_VALID = True # Can have targets
    DEP_VALID = True # Can contain dependencies for other tasks
    SRC_VALID = False # Should soucre code be automatically parsed?
    RUN_VALID = False # Has actions that should be executed at runtime
    IN_DIR = "./"
    OUT_DIR = "./"
    SRC_DIR = "./"

    def rig_task(self):
        raise NotImplementedError

    def provide_dependencies(self, asking_task: BaseTask) -> List[str]:
        return []

    def initialize_task(self):
        mkdir(self.target_dir)
        execute_at_target(dump_self, self)
```

## tasks/standard_task.py

```
from pdpp.tasks.base_task import BaseTask
from typing import List, Dict
from os import mkdir
from posixpath import join
from pdpp.utils.execute_at_target import execute_at_target
from pdpp.templates.create_in_out_src import create_in_out_src
from pdpp.languages.language_enum import Language
from pdpp.templates.dep_dataclass import dep_dataclass
from pdpp.languages.language_parser import parse_language


class StandardTask(BaseTask):
```

```
    """
    This is the class documentation
    """

    def __init__(
            self,
            target_dir: str = ""
            ):

        self.target_dir: str = target_dir
        self.dep_files: Dict[str, dep_dataclass] = {}
        self.src_files: List = []
        self.language: str = Language.NULL.value
        self.enabled: bool = True


    RIG_VALID = True # Can be rigged
    TRG_VALID = True # Can have targets
    DEP_VALID = True # Can contain dependencies for other tasks
    SRC_VALID = True # Should soucre code be automatically parsed?
    RUN_VALID = True # Has actions that should be executed at runtime
    IN_DIR = "input"
    OUT_DIR = "output"
    SRC_DIR = "src"

    def provide_run_actions(self) -> List:
        runner = parse_language(self)

        return [(runner, [s, self], {}) for s in self.src_files]

    def provide_src_dependencies(self) -> List:
        return [join(self.target_dir, self.SRC_DIR, s) for s in self.src_files]

    def initialize_task(self):

        # Create directory structure:
        mkdir(self.target_dir)
        execute_at_target(create_in_out_src, self)

        # From here on out, rigging is identical to creating anew:
        self.rig_task()
```

## tasks/sub_task.py

```
from pdpp.tasks.base_task import BaseTask
from typing import List, Dict
from os import mkdir
from pdpp.utils.execute_at_target import execute_at_target
from pdpp.templates.populate_new_project import populate_new_project
from pdpp.languages.language_enum import Language
from pdpp.templates.dep_dataclass import dep_dataclass
from pdpp.languages.runners import project_runner
```

```
class SubTask(BaseTask):
    """
    This is the class documentation
    """

    def __init__(
            self,
            target_dir: str = ""
            ):

        self.target_dir: str = target_dir
        self.dep_files: Dict[str, dep_dataclass] = {}
        self.src_files: List = []
        self.language: str = Language.NULL.value
        self.enabled: bool = True


    RIG_VALID = True # Can be rigged
    TRG_VALID = True # Can have targets
    DEP_VALID = True # Can contain dependencies for other tasks
    SRC_VALID = False # Should soucre code be automatically parsed?
    RUN_VALID = True # Has actions that should be executed at runtime
    IN_DIR = "_import_"
    OUT_DIR = "_export_"
    SRC_DIR = "./"

    def provide_run_actions(self) -> List:
        return [(project_runner, [self], {})]

    def initialize_task(self):

        # Create directory structure:
        mkdir(self.target_dir)
        execute_at_target(populate_new_project, self)

        # From here on out, rigging is identical to creating anew:
        self.rig_task()
```

## templates/create_gitignore.py

The majority of the following code is copied from the `Python.gitignore` file made available by GitHub via the `gitignore` repository (https://github.com/github/gitignore). All files therein are available under the Creative Commons 0 1.0 license, releasing each such work into the public domain with no need for attribution.

```
gitignore_text='''
# Byte-compiled / optimized / DLL files
__pycache__/
*.py[cod]
*$py.class
```

```
# C extensions
*.so

# Distribution / packaging
.Python
build/
develop-eggs/
dist/
downloads/
eggs/
.eggs/
lib/
lib64/
parts/
sdist/
var/
wheels/
share/python-wheels/
*.egg-info/
.installed.cfg
*.egg
MANIFEST

# PyInstaller
#  Usually these files are written by a python script from a template
#  before PyInstaller builds the exe, so as to inject date/other infos into it.
*.manifest
*.spec

# Installer logs
pip-log.txt
pip-delete-this-directory.txt

# Unit test / coverage reports
htmlcov/
.tox/
.nox/
.coverage
.coverage.*
.cache
nosetests.xml
coverage.xml
*.cover
*.py,cover
.hypothesis/
.pytest_cache/
cover/

# Translations
*.mo
*.pot

# Django stuff:
```

```
*.log
local_settings.py
db.sqlite3
db.sqlite3-journal

# Flask stuff:
instance/
.webassets-cache

# Scrapy stuff:
.scrapy

# Sphinx documentation
docs/_build/

# PyBuilder
.pybuilder/
target/

# Jupyter Notebook
.ipynb_checkpoints

# IPython
profile_default/
ipython_config.py

# pyenv
#   For a library or package, you might want to ignore these files since the code is
#   intended to run in multiple environments; otherwise, check them in:
# .python-version

# pipenv
#   According to pypa/pipenv#598, it is recommended to include Pipfile.lock in version control.
#   However, in case of collaboration, if having platform-specific dependencies or dependencies
#   having no cross-platform support, pipenv may install dependencies that don't work, or not
#   install all needed dependencies.
#Pipfile.lock

# PEP 582; used by e.g. github.com/David-OConnor/pyflow
__pypackages__/

# Celery stuff
celerybeat-schedule
celerybeat.pid

# SageMath parsed files
*.sage.py

# Environments
.env
.venv
env/
venv/
```

```
ENV/
env.bak/
venv.bak/

# Spyder project settings
.spyderproject
.spyproject

# Rope project settings
.ropeproject

# mkdocs documentation
/site

# mypy
.mypy_cache/
.dmypy.json
dmypy.json

# Pyre type checker
.pyre/

# pytype static type analyzer
.pytype/

# Cython debug symbols
cython_debug/

# PDPP-specific
_import_/*
!_import_/.gitkeep
_export_/*
!_export_/.gitkeep
!.pdpp_export.yaml

*/input/*
*/output/*
!*.gitkeep

# doit-specific
*.doit.db*
'''

def create_gitignore():
    with open('.gitignore', 'x') as stream:
        stream.write(gitignore_text)
```

## templates/create_in_out_src.py

```
from pdpp.tasks.base_task import BaseTask
from os import makedirs
from posixpath import join
```

```
def create_in_out_src(task: BaseTask):

    """
    Creates the input, output, and src directories, as well as a
    source file and .gitkeep files, in the new task.
    """

    makedirs("input")
    makedirs("output")
    makedirs("src")

    open(join("input", ".gitkeep"), "a").close()
    open(join("output", ".gitkeep"), "a").close()
    open(join("src", ".gitkeep"), "a").close()
    open(join("src", (task.target_dir + ".py")), "a").close()
```

## templates/dep_dataclass.py

```
from dataclasses import dataclass
from typing import List
from posixpath import join

@dataclass
class dep_dataclass():
    task_out: str
    task_name: str
    file_list: List[str]
    dir_list: List[str]

    def compile_targets(self):
        files: List[str] = [join(
            self.task_name, self.task_out, f) for f in self.file_list]
        dirs: List[str]  = [join(
            self.task_name, self.task_out, f) for f in self.dir_list]

        full_list = [*files, *dirs]

        return full_list
```

## templates/dodo_template.py

```
def create_dodo_template():

    f = open("dodo.py", "x")
    text = '''from pdpp.automation.task_creator import gen_many_tasks, task_all
import doit
doit.run(globals())
    '''
    f.write(text)
    f.close()
```

## templates/populate_new_project.py

```
from pdpp.templates.create_gitignore import create_gitignore
from pdpp.templates.dodo_template import create_dodo_template
from pdpp.tasks.export_task import ExportTask
from pdpp.tasks.import_task import ImportTask


def populate_new_project(_ = None):
    """
    This is run whenever `pdpp new` and `pdpp sub` is invoked.
    It checks to see whether the necessary support structure is in place.
    This is also run in a new pdpp subproject.
    """

    ExportTask().initialize_task()
    ImportTask().initialize_task()
    create_dodo_template()
    create_gitignore()
```

## utils/directory_test.py

```
from posixpath import exists, join
from os import scandir, listdir
from pdpp.tasks.base_task import BaseTask
from pdpp.utils.yaml_task import load_task
from typing import List, Tuple, Iterator


class NotInProjectException(Exception):
    """
    Raised when a user attempts to use most pdpp commands from outside
    project directory (project directories contain a dodo.py file)
    """
    pass


def in_project_directory():

    if exists("dodo.py") and len(listdir()) > 0:
        pass

    else:
        print("""Please run this command from an existing project directory
        (project directories contain a dodo.py file)""")
        raise NotInProjectException


def pdpp_directory_test(dir_) -> bool:
    """
    This function tests a directory to see if it is a valid pdpp-compliant
    task directory.
```

```
    """

    return exists(join(dir_, BaseTask.FILENAME))


def get_pdpp_directories() -> Iterator[Tuple[str]]:
    """
    Returns a list of all the riggable directories at this level of the project
    (or subproject)
    """

    return zip(*[(f, BaseTask.FILENAME) for f in [
    r.name for r in scandir() if r.is_dir()] if pdpp_directory_test(f)])


def get_pdpp_tasks() -> List[BaseTask]:
    """
    Returns a list containing loaded instances of all pdpp tasks in the current
    project. (Does not recursively search subprojects).
    """

    directories, filenames = get_pdpp_directories()

    return [load_task(task) for task in map(join, directories, filenames)]


def get_riggable_tasks() -> List[BaseTask]:

    return [task for task in get_pdpp_tasks() if task.RIG_VALID]


def get_dependency_tasks() -> List[BaseTask]:

    return [task for task in get_pdpp_tasks() if task.DEP_VALID]


def get_target_tasks() -> List[BaseTask]:

    return [task for task in get_pdpp_tasks() if task.TRG_VALID]


def get_runnable_tasks() -> List[BaseTask]:

    return [task for task in get_pdpp_tasks() if task.RUN_VALID]
```

## utils/execute_at_target.py

```
from os import chdir, getcwd

# VERY IMPORTANT: Use the 'execute_at_target' function wrapper to
# take actions inside task directories; do not manually
# move the target directory!!
```

```
def execute_at_target(func, task):
    original_dir = getcwd()
    chdir(task.target_dir)
    func(task)
    chdir(original_dir)
```

## utils/graph_dependencies.py

```
import networkx as nx
import os
from pdpp.tasks.base_task import BaseTask
from typing import List
from pdpp.automation.task_creator import find_dependencies_from_others
from posixpath import join
from pdpp.styles.graph_style import default_graph_style
import pydot

if os.name == 'nt':
    os.environ["PATH"] += os.pathsep + 'C:/Program Files (x86)/Graphviz2.38/bin/'
    os.environ["PATH"] += os.pathsep + 'C:/Program Files/Graphviz2.38/bin/'


def src_links(target_dir: str, source_file: str, G, gs):

    """
    This is a docstring
    """

    G.add_edge(
        source_file,
        target_dir,
        dir='none',
        color=gs.EDGE_COLOR,
        penwidth=gs.EDGE_PEN_WIDTH
    )
    #G.add_edge(target_dir, source_file, dir='none', color=gs.EDGE_COLOR, penwidth=2)
    G.nodes[source_file]['style'] = gs.SOURCE_STYLE
    G.nodes[source_file]['fontcolor'] = gs.SOURCE_FONT_COLOR
    G.nodes[source_file]['fillcolor'] = gs.SOURCE_FILL_COLOR
    G.nodes[source_file]['shape'] = gs.SOURCE_SHAPE
    G.nodes[source_file]['penwidth'] = gs.SOURCE_PENWIDTH
    G.nodes[source_file]['categ'] = 'source'


def node_colour(G, gs):

    """
    This is a docstring
    """

    for node in G:
        if G.nodes[node]['categ'] == 'disabled':
```

```
            G.nodes[node]['fillcolor'] = gs.DISABLED_TASK_COLOR
            G.nodes[node]['fontcolor'] = gs.DISABLED_TASK_FONTCOLOR
            G.nodes[node]['style'] = gs.DISABLED_TASK_STYLE
        elif G.in_degree(node) == 0:
            G.nodes[node]['fillcolor'] = gs.START_TASK_COLOR
        elif G.out_degree(node) == 0:
            G.nodes[node]['fillcolor'] = gs.END_TASK_COLOR
        else:
            G.nodes[node]['fillcolor'] = gs.MID_TASK_COLOR


def export_graph(G, output_name, files):

    """
    This is a docstring
    """

    # This loop checks for and removes any isolated nodes

    remlist = [node for node in G if G.degree[node] == 0]

    for node in remlist:
        G.remove_node(node)

    toPdot = nx.drawing.nx_pydot.to_pydot # type:ignore
    N:pydot.Dot = toPdot(G)

    N.obj_dict['attributes']['concentrate'] = 'true' # Combines edges
    N.obj_dict['attributes']['rankdir'] = 'LR' # Runs graph left-to-right
    N.obj_dict['attributes']['dpi'] = 300

    # Add all source nodes to subgraphs with their tasks so that they are
    # displayed on the same rank:

    for edge in N.get_edge_list():
        if N.get_node(edge.get_source())[0].get('categ') == "source":
            S = pydot.Subgraph(rank = 'same')
            S.add_node(N.get_node(edge.get_source())[0])
            S.add_node(N.get_node(edge.get_destination())[0])
            N.add_subgraph(S)

    ext = "." + files

    N.write(output_name + ext, prog='dot', format=files)



def depgraph(files='png', gs=default_graph_style):

    """
    This is a docstring
    """
```

```
from pdpp.utils.directory_test import get_pdpp_tasks

all_tasks: List[BaseTask] = get_pdpp_tasks()

SOURCE = nx.DiGraph()
SPARSE = nx.DiGraph()

nodes = []
edges = []
disabled_nodes = []

"""
This section populates the graph with nodes and edges from
dependency tasks to dependent tasks
"""

for task in all_tasks:
    if task.enabled:
        nodes.append(task.target_dir)
    else:
        disabled_nodes.append(task.target_dir)
    for linkage in task.dep_files:
        edges.append((linkage, task.target_dir))

"""
This section creates the SPARSE graph, consisting only of
edges indicating dependencies between tasks
"""

SPARSE.add_nodes_from(
    nodes,
    style=gs.TASK_NODE_STYLE,
    shape=gs.TASK_NODE_SHAPE,
    penwidth=gs.TASK_NODE_PENWIDTH,
    categ="task"
    )
SPARSE.add_nodes_from(
    disabled_nodes,
    style=gs.TASK_NODE_STYLE,
    shape=gs.TASK_NODE_SHAPE,
    penwidth=gs.TASK_NODE_PENWIDTH,
    categ="disabled"
    )
SPARSE.add_edges_from(
    edges,
    color=gs.EDGE_COLOR,
    penwidth=gs.EDGE_PEN_WIDTH
    )
node_colour(SPARSE, gs)

output_name = "dependencies_sparse"
export_graph(SPARSE, output_name, files)
```

```
    """
    The SOURCE graph can be built out from the SPARSE graph;
    it simply adds source files and draws edges between them and their tasks
    """

    SOURCE = SPARSE.copy()

    for task in all_tasks:
        for source_file in task.src_files:
            src_links(task.target_dir, source_file, SOURCE, gs)


    output_name = "dependencies_source"
    export_graph(SOURCE, output_name, files)

    """
    The FILE graph is built from scratch, using edges to represent
    the connections between tasks and the files that they have as either
    targets (implicitly defined as a file they output that another task
    relies upon) or dependencies (defined explicitly)
    """

    FILE = nx.create_empty_copy(SPARSE)

    for task in all_tasks:

        # Add edges from dependency files
        for dep_dataclass in task.dep_files.values():

            # ADD FILES
            for dep_file in dep_dataclass.file_list:
                dep_name = join(
                    dep_dataclass.task_name,
                    dep_dataclass.task_out,
                    dep_file
                )
                FILE.add_node(
                    dep_name,
                    style=gs.FILE_NODE_STYLE,
                    shape=gs.FILE_FILE_SHAPE,
                    fillcolor=gs.FILE_NODE_COLOR,
                    categ='file',
                    label=dep_file,
                    penwidth=gs.FILE_NODE_PENWIDTH
                    )
                FILE.add_edge(
                    dep_name,
                    task.target_dir,
                    color=gs.EDGE_COLOR,
                    penwidth=gs.EDGE_PEN_WIDTH
                    )

            # ADD DIRECTORIES
```

```
            for dep_dir in dep_dataclass.dir_list:
                dep_name = join(
                    dep_dataclass.task_name,
                    dep_dataclass.task_out,
                    dep_dir
                )
                dep_label = dep_dir + "/"
                FILE.add_node(
                    dep_name,
                    style=gs.FILE_NODE_STYLE,
                    shape=gs.FILE_DIR_SHAPE,
                    fillcolor=gs.FILE_NODE_COLOR,
                    categ='file',
                    label=dep_label,
                    penwidth=gs.FILE_NODE_PENWIDTH
                    )
                FILE.add_edge(
                    dep_name,
                    task.target_dir,
                    color=gs.EDGE_COLOR,
                    penwidth=gs.EDGE_PEN_WIDTH
                    )

        # Add edges to targets (as defined by others)
        target_list = find_dependencies_from_others(task, all_tasks)

        for full_target_path in target_list:
            target_name = full_target_path.split('/')[-1]
            FILE.add_edge(
                task.target_dir,
                full_target_path,
                color=gs.EDGE_COLOR,
                penwidth=gs.EDGE_PEN_WIDTH)


    output_name = "dependencies_file"
    export_graph(FILE, output_name, files)

    ALL = FILE.copy()
    for task in all_tasks:
        for source_file in task.src_files:
            src_links(task.target_dir, source_file, ALL, gs)

    output_name = "dependencies_all"
    export_graph(ALL, output_name, files)

if __name__ == '__main__':
    depgraph()
```

## utils/ignorelist.py

```
ignorelist = [".gitkeep", "__pycache__", ".ipynb_checkpoints", ".pdpp_task.yaml"]
link_ignorelist = [".gitkeep", ".pdpp_task.yaml"]
```

## utils/immediate_link.py

```
from os import link, walk, remove
from posixpath import join
from pdpp.tasks.base_task import BaseTask
from shutil import rmtree, copytree
from pdpp.utils.ignorelist import ignorelist


def immediate_link(task:BaseTask):
    """
    This is a docstring.
    """

    #The following block cleans the input directory by deleting everything
    # in it recursively, except for things on the ignorelist

    in_dir = join(task.target_dir, task.IN_DIR)
    for root, dirs, files in walk(in_dir):
        for name in [f for f in files if f not in ignorelist]:
            remove(join(in_dir, name))
        for name in dirs:
            rmtree(join(in_dir, name))

    for key, value in task.dep_files.items():

        out_dir = join(key, value.task_out)

        for file_entry in value.file_list:
            pre_link = join(out_dir, file_entry)
            post_link = join(in_dir, file_entry)
            link(pre_link, post_link)

        for directory_entry in value.dir_list:
            pre_link = join(out_dir, directory_entry)
            post_link = join(in_dir, directory_entry)
            copytree(pre_link, post_link, copy_function=link)
```

## utils/rem_slash.py

```
def rem_slash(string_in):
    """
    A simple function which takes in a string and returns it stripped of
    double backslashes, single forward slashes, and spaces.
    """
```

```
    return str(string_in).replace("\\", "").replace("/", "").replace(" ", "")
```

## utils/task_directory_test.py

```
import click
from pdpp.utils.rem_slash import rem_slash
import os
from os.path import isdir, join


class ExistingTaskDirectory(click.types.StringParamType):
    """
    Custom type for directories to check that it is already an existing directory,
    and that it is a proper task directory (meaning it includes input, output, and
    src subdirectories).
    """

    def convert(self, value, param, ctx):
        directory = rem_slash(super(
            ExistingTaskDirectory, self).convert(value, param, ctx))

        if directory not in os.listdir():
            raise self.fail("""This directory does not exist.
            Please choose an existing task directory.""", param, ctx) # type: ignore

        elif not isdir(join(
            directory, 'src'
        )) or not isdir(join(
            directory, 'input'
        )) or not isdir(join(directory, 'output')):
            raise self.fail("This is not a valid task directory.\n"
                            """Please choose an existing task directory that you
                            created by running pdpp task.""")# type: ignore

        return directory


class TaskDirectory(click.types.StringParamType):
    """
    Custom type for directories with no spaces, and to ensure that the directory
    name does not already exist. This will keep prompting the user for a proper
    input type if they enter a directory name with spaces or a directory name
    that already exists.
    """

    def convert(self, value, param, ctx):
        directory = super(TaskDirectory, self).convert(value, param, ctx)
        if ' ' in directory:
            raise self.fail(
            "\n No spaces allowed. Please try again. \n", param, ctx) # type: ignore

        if directory == "_import_":
```

```
            raise self.fail(
            "\n'_import_' is reserved and is not a valid task name. Please try again. \n",
            param,
            ctx
            ) # type: ignore

        if directory == "_export_":
            raise self.fail(
            "\n'export' is reserved and is not a valid task name.  Please try again. \n",
            param,
            ctx
            ) # type: ignore

        # It appears to already be checking for it all to be lower case,
        # but I'm not sure where..

        if directory in os.listdir():
            raise self.fail(
            "\nThis task directory already exists.\nPlease choose another name for your directory. \n",
            param,
            ctx
            ) # type: ignore

        return directory
```

## utils/yaml_task.py

```
import yaml

def dump_self(task):
    with open(type(task).FILENAME, 'w') as stream:
        yaml.dump(task, stream, default_flow_style=False)


def load_task(task_path):
    from pdpp.tasks.custom_task import CustomTask
    from pdpp.tasks.export_task import ExportTask
    from pdpp.tasks.import_task import ImportTask
    from pdpp.tasks.standard_task import StandardTask
    from pdpp.tasks.sub_task import SubTask
    with open(task_path, 'r') as stream:
        return yaml.load(stream, Loader=yaml.Loader)
```

# Appendix D – Installation Instructions for ‘pdpp‘

What follows is a expanded version of the installation instructions found in the readme for the `pdpp` package, which can be found at https://github.com/UWNETLAB/pdpp

## Installation Prerequisites

Aside from an up-to-date installation of `python` and `pip` (installation instructions for which can be found here), the `pdpp` package depends on `graphviz`, which must be installed before attempting to install `pdpp`. Installation instructions for `graphviz` can be found at the GraphViz installation instructions page.

## Installation

To install `pdpp`, use the following terminal command:

```
pip install pdpp
```

## Further Reading

Readers unfamiliar with the use of Git and GitHub should consult the tutorial found here: https://docs.github.com/en/get-started/quickstart/

Readers unfamiliar with the use of Python should consult the Python beginner’s guide (https://wiki.python.org/moin/BeginnersGuide), or consult the list of resources found here: https://www.python.org/about/gettingstarted/

For a more detailed set of tutorials on the use of Python, terminals, and Jupyter Notebooks (which are not necessary but can be highly useful), interested readers should consult *Doing Computational Social Science: A Practical Introduction* (2021).

# Appendix E – Python code for ‘pykrusch‘

The code contained in this appendix is a full and true copy[178] of the non-empty Python code files for `pykrusch` (version `0.1.1`) as indexed in PYPI at the time of this dissertation's publication.[179]

Each of the following code blocks contains the code from one of the modules in the `pykrusch` package.

## `__init__.py`

```
from pykrusch.main import krusch
```

## `config.py`

```
DATA_LETTER = "X"
LATEX_NAMES = True

PARAM_CONVERSION = {
    "mu": "μ",
    "lam": "λ",
    "sigma": "σ",
    "alpha": "α",
    "beta": "β",
    "gamma": "γ",
    "nu": "ν",
    "tau": "τ",
    "pi": "π",
    "lambda": "λ",
    "theta": "θ",
}

NODE_COLOUR_GV = "grey97"
NODE_COLOUR_MPL = "#F7F7F7"

PARAM_FONT_SIZE = "35.0"
PARAM_FONT_SIZE_NUMERICAL = "30.0"
DIST_FONT_SIZE = "40.0"
DIST_FONT_SIZE_SMALL = "20.0"

IMAGE_SCALE_FACTOR = 1
DIST_IMAGE_WIDTH = 82.5
```

[178] Some code has been reformatted to better fit onto the printed page; the code is otherwise unaltered.
[179] The package's non-Python files are not included here, as they do not in any way contribute to or alter the package's behaviour.

```
DIST_IMAGE_HEIGHT = 61.5

DISTROGRAM_LINE_WIDTH = 5
DISCRETE_LINE_WIDTH = 2


if LATEX_NAMES:
    DETERMINISTIC_TEXT_SCALE = 1
else:
    DETERMINISTIC_TEXT_SCALE = 0.7
DETERMINISTIC_TEXT_HEIGHT = 80

TEMP_IMG_DIR = ".PYKRUSCHFIGTEMP"

DPI = 300
ENGINE = "dot"

LINSPACE_N = 500
```

## dist.py

```
from __future__ import annotations
from pykrusch.param import (
    NumericalParameter,
    NamedParameter,
    UnknownParameter,
)
from pykrusch.param import recursive_rv_search
from pykrusch.figureControl import FC, MathImage, MathImageFound
from pykrusch.graphviz.distrogramsMPL import (
    plot_distrogram,
    plot_unknown_distrogram,
    plot_discrete_distrogram,
)
from pykrusch.graphviz.latexText import string_to_latex, render_img_from_latex
from typing import TYPE_CHECKING
from collections import namedtuple
import scipy.stats as sps
import numpy as np
import pymc as pm

np.seterr(divide="ignore")

if TYPE_CHECKING:
    from pykrusch.param import Parameter
    from pykrusch.parsePymc.randomVariable import RandomVar
    from arviz import InferenceData

_ = 0

partup = namedtuple(
    "Param",
```

```
    [
        "greek_name",
        "pos",
        "meaning",
        "op",
    ],
    defaults=[None],
)


def reciprocal(n):
    return 1.0 / n


class Dist:
    def __init__(self, owner, varname=""):
        # These attributes are to be overwritten by the specific classes
        self.needed_params: list[partup]
        self.type: str
        self.distimage: str
        self.fz: sps.rv_continuous | sps.rv_discrete
        self.numerical: bool = False
        self.plot_posterior: bool = False
        self.posterior_data: InferenceData | None = None
        self.scipy = True

        # These attributes are NOT to be overwritten
        self.owner = owner
        self.params: list[Parameter] = self.get_params()
        self.varname = varname
        self.num_params: int = len(self.params)

    def get_params(self):
        parameters = []
        for needed_param in self.needed_params:
            result = recursive_rv_search(self.owner.get_parents()[needed_param.pos])
            result.give_slot(needed_param.pos)
            result.give_greek(needed_param.greek_name)
            result.give_meaning(needed_param.meaning)

            # Apply the special operation to numerical values that get fed through
            if needed_param.op and isinstance(result, NumericalParameter):
                result.value = needed_param.op(result.value)

            parameters.append(result)

        return parameters

    @property
    def dist_name_latex(self):
        mi: MathImage = FC.fig_path(FC, symbols=self.varname)
        if not isinstance(mi, MathImageFound):
            render_img_from_latex(string_to_latex(self.varname), mi.filepath)
```

```
        return mi.filepath

    ### TODO: include something here to change between pdf and pmf
    ### depending on whether the distribution is continuous or discrete

    @property
    def distimage(self):
        mi_distrogram: MathImage = FC.fig_path(FC)

        plot_distrogram(
            dist=self,
            mi=mi_distrogram,
            plot_posterior=self.plot_posterior,
            posterior_data=self.posterior_data,
        )
        return mi_distrogram.filepath

    def ppf(self, x):
        return self.fz.ppf(x)

    def pdf(self, x):
        return self.fz.pdf(x)

    @property
    def xmin(self):
        return self.ppf(0.01) - self.xbuffer

    @property
    def xmax(self):
        return self.ppf(0.99) + self.xbuffer

    @property
    def xbuffer(self):
        range = self.ppf(0.99) - self.ppf(0.01)
        return range * 0.1

    def __repr__(self):
        return str(self.type)

    def __str__(self):
        return self.__repr__()


class UnknownDist(Dist):
    def __init__(self, owner, varname=""):
        self.type = "unknown"
        self.numerical = False
        self.owner = owner
        self.params: list[Parameter] = self.get_params()
        self.varname = varname

        self.num_params = 1
```

```
    # need to make sure that all of these are

    def get_params(self):
        parameters = []
        for param_slot in self.owner.get_parents()[3:]:
            result = recursive_rv_search(param_slot)
            result.greek_name = "?"
            result.meaning = " "

            if isinstance(result, NamedParameter):
                parameters.append(result)

        if len(parameters) == 0:
            result = UnknownParameter()
            result.greek_name = "?"
            result.meaning = " "
            result.slot = 1

            parameters.append(result)

        return parameters

    @property
    def distimage(self):
        mi_distrogram: MathImage = FC.fig_path(FC)

        plot_unknown_distrogram(dist=self, mi=mi_distrogram)
        return mi_distrogram.filepath


class Normal(Dist):
    def __init__(self, owner, varname=""):
        self.type = "Normal"
        self.needed_params = [
            partup("mu", 3, "location"),
            partup("sigma", 4, "scale"),
        ]
        super().__init__(owner, varname)

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.fz = sps.norm(
                loc=self.params[0].value,
                scale=self.params[1].value,
            )
            self.numerical = True
        else:
            self.fz = sps.norm()


class Exponential(Dist):
    def __init__(self, owner, varname=""):
        self.type = "Exponential"
        self.needed_params = [
```

```
            partup("lambda", 3, "rate"),
        ]
        super().__init__(owner, varname)

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.fz = sps.expon(scale=1 / self.params[0].value)
            self.numerical = True
        else:
            self.fz = sps.expon(1)


class Gamma(Dist):
    def __init__(self, owner, varname=""):
        self.type = "Gamma"
        self.needed_params = [
            partup("alpha", 3, "shape"),
            partup("beta", 4, "rate", reciprocal),
        ]
        super().__init__(owner, varname)

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.fz = sps.gamma(
                a=self.params[0].value,
                scale=1 / self.params[1].value,
            )
            self.numerical = True
        else:
            self.fz = sps.gamma(2, 2)


class Weibull(Dist):
    def __init__(self, owner, varname=""):
        self.type = "Weibull"
        self.needed_params = [partup("alpha", 3, "shape"), partup("beta", 4, "scale")]
        super().__init__(owner, varname)

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.numerical = True
        else:
            self.fz = sps.weibull_min(2, 2)

    def pdf(self, x):
        if self.numerical:
            return sps.weibull_min.pdf(
                x,
                c=self.params[0].value,
                scale=self.params[1].value,
            )
        else:
            return self.fz(x)

    def ppf(self, x):
        if self.numerical:
```

```
            return sps.weibull_min.ppf(
                x,
                c=self.params[0].value,
                scale=self.params[1].value,
            )
        else:
            return self.fz(x)


class Uniform(Dist):
    def __init__(self, owner, varname=""):
        self.type = "uniform"
        self.needed_params = [partup("", 3, "lower"), partup("", 4, "upper")]
        super().__init__(owner, varname)

        if all(isinstance(p, NumericalParameter) for p in self.params):
            lower = self.params[0].value
            upper = self.params[1].value
            scale = upper - lower
            self.fz = sps.uniform(loc=lower, scale=scale)
            self.numerical = True
        else:
            self.fz = sps.uniform()


class TruncatedNormal(Dist):
    def __init__(self, owner, varname=""):
        self.type = "truncated normal"
        self.needed_params = [
            partup("mu", 3, "location"),
            partup("sigma", 4, "scale"),
            partup("", 5, "lower"),
            partup("", 6, "upper"),
        ]
        super().__init__(owner, varname)

        fz = sps.truncnorm

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.fz = fz(
                loc=self.params[0].value,
                scale=self.params[1].value,
                a=self.params[2].value,
                b=self.params[3].value,
            )
            self.numerical = True
        else:
            self.fz = fz(
                loc=0,
                scale=1,
                a=-1,
                b=1,
            )
```

```
class Beta(Dist):
    def __init__(self, owner, varname=""):
        self.type = "Beta"
        self.needed_params = [
            partup("alpha", 3, "shape"),
            partup("beta", 4, "shape"),
        ]
        super().__init__(owner, varname)

        fz = sps.beta

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.fz = fz(
                a=self.params[0].value,
                b=self.params[1].value,
            )
            self.numerical = True
        else:
            self.fz = fz(
                a=3,
                b=5,
            )

    @property
    def xmin(self):
        return 0

    @property
    def xmax(self):
        return 1


class Kumaraswamy(Dist):
    def __init__(self, owner, varname=""):
        self.type = "kumaraswamy"
        self.needed_params = [
            partup("alpha", 3, "shape"),
            partup("beta", 4, "shape"),
        ]
        super().__init__(owner, varname)

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.a = self.params[0].value
            self.b = self.params[1].value
            self.numerical = True
        else:
            self.a = 3
            self.b = 5

    def pdf(self, x):
        a = self.a
        b = self.b
```

```
        return a * b * (x ** (a - 1)) * ((1 - (x**a)) ** (b - 1))

    @property
    def xmin(self):
        return 0

    @property
    def xmax(self):
        return 1


class Laplace(Dist):
    def __init__(self, owner, varname=""):
        self.type = "laplace"
        self.needed_params = [
            partup("mu", 3, "location"),
            partup("b", 4, "scale"),
            #             partup("", 5, ""),
            #             partup("", 6, ""),
        ]
        super().__init__(owner, varname)

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.mu = self.params[0].value
            self.b = self.params[1].value
            #             _ = self.params[2].value,
            #             _ = self.params[3].value,

            self.numerical = True
        else:
            self.mu = 3
            self.b = 5
        #                 _ = 7,
        #                 _ = 9,

        self.fz: sps.rv_continuous = sps.laplace(
            scale=self.b,
            loc=self.mu
            # _=self._,
            # _=self._,
        )

    def ppf(self, x):
        return self.fz.ppf(x)

    def pdf(self, x):
        return self.fz.pdf(x)


class StudentT(Dist):
    def __init__(self, owner, varname=""):
        self.type = "Student T"
        self.needed_params = [
```

```
            partup("nu", 3, "d.o.f."),
            partup("mu", 4, "location"),
            partup("sigma", 5, "scale"),
            #             partup("", 6, ""),
        ]
        super().__init__(owner, varname)

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.nu = self.params[0].value
            self.mu = self.params[1].value
            self.sigma = self.params[2].value
            #             _ = self.params[3].value,

            self.numerical = True
        else:
            self.nu = 3
            self.mu = 5
            self.sigma = 7
        #                 _ = 9,

        self.fz: sps.rv_continuous = sps.t

    def ppf(self, x):
        return self.fz.ppf(x, df=self.nu, loc=self.mu, scale=self.sigma)

    def pdf(self, x):
        return self.fz.pdf(x, df=self.nu, loc=self.mu, scale=self.sigma)


class HalfStudent(Dist):
    def __init__(self, owner, varname=""):
        self.type = "Half Student"
        self.needed_params = [
            partup("nu", 3, "d.o.f."),
            partup("sigma", 4, "scale"),
            #             partup("", 5, ""),
            #             partup("", 6, ""),
        ]
        super().__init__(owner, varname)
        self.scipy = False

        self.nu: float
        self.sigma: float
        self.___: float
        self.____: float

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.nu = self.params[0].value
            self.sigma = self.params[1].value
            self.numerical = True

        else:
            self.nu = 3
```

```
            self.sigma = 5

    def pdf(self, x):
        fz = pm.HalfStudentT.dist(nu=self.nu, sigma=self.sigma)
        out = pm.logp(fz, x).eval()
        return np.e ** (out)

    @property
    def xmin(self):
        return -0.1

    @property
    def xmax(self):
        return 5


class Cauchy(Dist):
    def __init__(self, owner, varname=""):
        self.type = "Cauchy"
        self.needed_params = [
            partup("alpha", 3, "location"),
            partup("beta", 4, "scale"),
            #             partup("", 5, ""),
            #             partup("", 6, ""),
        ]
        super().__init__(owner, varname)

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.a = self.params[0].value
            self.b = self.params[1].value
            #             self._ = self.params[2].value,
            #             self._ = self.params[3].value,

            self.numerical = True
        else:
            self.a = 0
            self.b = 1
        #                 self._ = 7,
        #                 self._ = 9,

        self.fz: sps.rv_continuous = sps.cauchy

    def ppf(self, x):
        return self.fz.ppf(x, loc=self.a, scale=self.b)

    def pdf(self, x):
        return self.fz.pdf(x, loc=self.a, scale=self.b)


class HalfCauchy(Dist):
    def __init__(self, owner, varname=""):
        self.type = "Half Cauchy"
        self.needed_params = [
```

```
            partup("beta", 4, "scale"),
            #             partup("", 5, ""),
            #             partup("", 6, ""),
        ]
        super().__init__(owner, varname)

        self.beta: float
        self.__: float
        self.___: float
        self.____: float

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.beta = self.params[0].value
            #             self.___ = self.params[2].value
            #             self.____ = self.params[3].value

            self.numerical = True
        else:
            self.beta = 3
        #                 self.___ = 7
        #                 self.____ = 9

        self.fz: sps.rv_continuous = sps.halfcauchy

    def ppf(self, x):
        result = self.fz.ppf(x, scale=self.beta)
        return result

    def pdf(self, x):
        return self.fz.pdf(x, scale=self.beta)


class LogNormal(Dist):
    def __init__(self, owner, varname=""):
        self.type = "LogNormal"
        self.needed_params = [
            partup("mu", 3, "location"),
            partup("sigma", 4, "scale"),
            #             partup("", 6, ""),
        ]
        super().__init__(owner, varname)

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.mu = (self.params[0].value,)
            self.sigma = (self.params[1].value,)
            #             self._ = self.params[3].value,

            self.numerical = True
        else:
            self.mu = (3,)
            self.sigma = (5,)
        #                 self._ = 9,
```

```
        self.fz: sps.rv_continuous = sps.lognorm

    def ppf(self, x):
        return self.fz.ppf(x, 1, loc=self.mu, scale=self.sigma)

    def pdf(self, x):
        return self.fz.pdf(x, 1, loc=self.mu, scale=self.sigma)

    @property
    def xmin(self):
        return self.ppf(0.1) - self.xbuffer

    @property
    def xmax(self):
        return self.ppf(0.9) + self.xbuffer

    @property
    def xbuffer(self):
        range = self.ppf(0.9) - self.ppf(0.1)
        return range * 0.1


class ChiSquared(Dist):
    def __init__(self, owner, varname=""):
        self.type = "Chi-Squared"
        self.needed_params = [
            partup("nu", 3, "d.o.f."),
            #             partup("", 5, ""),
            #             partup("", 6, ""),
        ]
        super().__init__(owner, varname)

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.nu = self.params[0].value
            #             self._ = self.params[2].value,
            #             self._ = self.params[3].value,

            self.numerical = True
        else:
            self.nu = 3
        #                 self._ = 7,
        #                 self._ = 9,

        self.fz: sps.rv_continuous = sps.chi2(
            df=self.nu,
            #             _=self._,
            #             _=self._,
        )


class HalfNormal(Dist):
    def __init__(self, owner, varname=""):
        self.type = "Half Normal"
```

```
        self.needed_params = [
            partup("sigma", 4, "scale"),
            #             partup("", 5, ""),
            #             partup("", 6, ""),
        ]
        super().__init__(owner, varname)

        self.sigma: float
        self.__: float
        self.___: float
        self.____: float

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.sigma = self.params[0].value
            #             self.___ = self.params[2].value
            #             self.____ = self.params[3].value

            self.numerical = True
        else:
            self.sigma = 3
        #                 self.___ = 7
        #                 self.____ = 9

        self.fz: sps.rv_continuous = sps.halfnorm

    def ppf(self, x):
        result = self.fz.ppf(x, scale=self.sigma)
        return result

    def pdf(self, x):
        return self.fz.pdf(x, scale=self.sigma)


class Wald(Dist):
    def __init__(self, owner, varname=""):
        self.type = "Wald"
        self.needed_params = [
            partup("mu", 3, "location"),
            partup("lam", 4, "precision"),
            partup("alpha", 5, "shift"),
            #             partup("", 6, ""),
        ]
        super().__init__(owner, varname)

        self.mu: float
        self.lam: float
        self.alpha: float
        # self.____: float

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.mu = self.params[0].value
            self.lam = self.params[1].value
            self.alpha = self.params[2].value
```

```
            #             self.____ = self.params[3].value

            self.numerical = True
        else:
            self.mu = 3
            self.lam = 5
            self.alpha = 7

    #                 self.____ = 9

    def pdf(self, x):
        mu = self.mu
        lam = self.lam
        term1 = (lam / (2 * np.pi)) ** (1 / 2)
        term2 = x ** ((-3) / 2)
        term3 = (-lam / (2 * x)) * (((x - mu) / mu) ** 2)
        return term1 * term2 * (np.e ** (term3))

    @property
    def xmin(self):
        return 0

    @property
    def xmax(self):
        return 10


class Pareto(Dist):
    def __init__(self, owner, varname=""):
        self.type = "Pareto"
        self.needed_params = [
            partup("alpha", 3, "shape"),
            partup("m", 4, "scale"),
            #             partup("", 5, ""),
            #             partup("", 6, ""),
        ]
        super().__init__(owner, varname)

        self.alpha: float
        self.m: float
        self.___: float
        self.____: float

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.alpha = self.params[0].value
            self.m = self.params[1].value
            #             self.___ = self.params[2].value
            #             self.____ = self.params[3].value

            self.numerical = True
        else:
            self.alpha = 3
            self.m = 5
```

```
        #                 self.___ = 7
        #                 self.____ = 9

        self.fz: sps.rv_continuous = sps.pareto

    def ppf(self, x):
        return self.fz.ppf(x, b=self.alpha, scale=self.m)

    def pdf(self, x):
        return self.fz.pdf(x, b=self.alpha, scale=self.m)


class InverseGamma(Dist):
    def __init__(self, owner, varname=""):
        self.type = "Inverse Gamma"
        self.needed_params = [
            partup("alpha", 3, "shape"),
            partup("beta", 4, "scale"),
            #             partup("", 5, ""),
            #             partup("", 6, ""),
        ]
        super().__init__(owner, varname)

        self.alpha: float
        self.beta: float
        self.___: float
        self.____: float

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.alpha = self.params[0].value
            self.beta = self.params[1].value
            #             self.___ = self.params[2].value
            #             self.____ = self.params[3].value

            self.numerical = True
        else:
            self.alpha = 3
            self.beta = 5
        #                 self.___ = 7
        #                 self.____ = 9

        self.fz: sps.rv_continuous = sps.invgamma

    def ppf(self, x):
        return self.fz.ppf(x, a=self.alpha, scale=self.beta)

    def pdf(self, x):
        return self.fz.pdf(x, a=self.alpha, scale=self.beta)


class ExGaussian(Dist):
    def __init__(self, owner, varname=""):
        self.type = "Exponential Gaussian"
```

```
        self.needed_params = [
            partup("mu", 3, "location"),
            partup("sigma", 4, "scale"),
            partup("nu", 5, "exponential"),
            #             partup("", 6, ""),
        ]
        super().__init__(owner, varname)

        self.mu: float
        self.sigma: float
        self.nu: float
        self.____: float

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.mu = self.params[0].value
            self.sigma = self.params[1].value
            self.nu = self.params[2].value
            #             self.____ = self.params[3].value

            self.numerical = True
        else:
            self.mu = 3
            self.sigma = 5
            self.nu = 7
        #                 self.____ = 9

        self.fz: sps.rv_continuous = sps.exponnorm

    def ppf(self, x):
        return self.fz.ppf(x, K=self.nu, loc=self.mu, scale=self.sigma)

    def pdf(self, x):
        return self.fz.pdf(x, K=self.nu, loc=self.mu, scale=self.sigma)


class VonMises(Dist):
    def __init__(self, owner, varname=""):
        self.type = "Von Mises"
        self.needed_params = [
            partup("mu", 3, "location"),
            partup("kappa", 4, "concentration"),
            #             partup("", 5, ""),
            #             partup("", 6, ""),
        ]
        super().__init__(owner, varname)

        self.mu: float
        self.kappa: float
        self.___: float
        self.____: float

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.mu = self.params[0].value
```

```
            self.kappa = self.params[1].value
            #             self.___ = self.params[2].value
            #             self.____ = self.params[3].value

            self.numerical = True
        else:
            self.mu = 3
            self.kappa = 5
        #                 self.___ = 7
        #                 self.____ = 9

        self.fz: sps.rv_continuous = sps.vonmises

    def ppf(self, x):
        return self.fz.ppf(x, kappa=self.kappa, loc=self.mu)

    def pdf(self, x):
        return self.fz.pdf(x, kappa=self.kappa, loc=self.mu)


class SkewNormal(Dist):
    def __init__(self, owner, varname=""):
        self.type = "Skew Normal"
        self.needed_params = [
            partup("mu", 3, "location"),
            partup("sigma", 4, "scale"),
            partup("alpha", 5, "skew"),
            #             partup("", 6, ""),
        ]
        super().__init__(owner, varname)

        self.mu: float
        self.sigma: float
        self.alpha: float
        self.____: float

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.mu = self.params[0].value
            self.sigma = self.params[1].value
            self.alpha = self.params[2].value
            #             self.____ = self.params[3].value

            self.numerical = True
        else:
            self.mu = 3
            self.sigma = 5
            self.alpha = 7
        #                 self.____ = 9

        self.fz: sps.rv_continuous = sps.skewnorm

    def ppf(self, x):
        return self.fz.ppf(x, a=self.alpha, loc=self.mu, scale=self.sigma)
```

```
    def pdf(self, x):
        return self.fz.pdf(x, a=self.alpha, loc=self.mu, scale=self.sigma)


class Triangular(Dist):
    def __init__(self, owner, varname=""):
        self.type = "Triangular"
        self.needed_params = [
            partup("", 3, "lower"),
            partup("c", 4, "mode"),
            partup("", 5, "upper"),
            #             partup("", 6, ""),
        ]
        super().__init__(owner, varname)

        self.lower: float
        self.c: float
        self.upper: float
        self.____: float

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.lower = self.params[0].value
            self.c = self.params[1].value
            self.upper = self.params[2].value
            #             self.____ = self.params[3].value

            self.numerical = True
        else:
            self.lower = 3
            self.c = 5
            self.upper = 7
        #                 self.____ = 9

        self.fz: sps.rv_continuous = sps.triang

    def ppf(self, x):
        c = (self.c - self.lower) / (self.upper - self.lower)
        upper = self.upper - self.lower
        result = self.fz.ppf(x, c=c, loc=self.lower, scale=upper)
        return result

    def pdf(self, x):
        c = (self.c - self.lower) / (self.upper - self.lower)
        upper = self.upper - self.lower
        result = self.fz.pdf(x, c=c, loc=self.lower, scale=upper)
        return result


class Gumbel(Dist):
    def __init__(self, owner, varname=""):
        self.type = "Gumbel"
        self.needed_params = [
            partup("mu", 3, "location"),
```

```
            partup("beta", 4, "scale"),
            #             partup("", 5, ""),
            #             partup("", 6, ""),
        ]
        super().__init__(owner, varname)

        self.mu: float
        self.beta: float
        self.___: float
        self.____: float

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.mu = self.params[0].value
            self.beta = self.params[1].value
            #             self.___ = self.params[2].value
            #             self.____ = self.params[3].value

            self.numerical = True
        else:
            self.mu = 3
            self.beta = 5
        #                 self.___ = 7
        #                 self.____ = 9

        self.fz: sps.rv_continuous = sps.gumbel_r

    def ppf(self, x):
        return self.fz.ppf(x, loc=self.mu, scale=self.beta)

    def pdf(self, x):
        return self.fz.pdf(x, loc=self.mu, scale=self.beta)


class Logistic(Dist):
    def __init__(self, owner, varname=""):
        self.type = "Logistic"
        self.needed_params = [
            partup("mu", 3, "location"),
            partup("beta", 4, "scale"),
            #             partup("", 5, ""),
            #             partup("", 6, ""),
        ]
        super().__init__(owner, varname)

        self.mu: float
        self.s: float
        self.___: float
        self.____: float

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.mu = self.params[0].value
            self.s = self.params[1].value
            #             self.___ = self.params[2].value
```

```
            #             self.____ = self.params[3].value

            self.numerical = True
        else:
            self.mu = 3
            self.s = 5
        #                 self.___ = 7
        #                 self.____ = 9

        self.fz: sps.rv_continuous = sps.logistic

    def ppf(self, x):
        return self.fz.ppf(x, loc=self.mu, scale=self.s)

    def pdf(self, x):
        return self.fz.pdf(x, loc=self.mu, scale=self.s)


class LogitNormal(Dist):
    def __init__(self, owner, varname=""):
        self.type = "Logit Normal"
        self.needed_params = [
            partup("mu", 3, "location"),
            partup("sigma", 4, "scale"),
            #             partup("", 5, ""),
            #             partup("", 6, ""),
        ]
        super().__init__(owner, varname)
        self.scipy = False

        self.mu: float
        self.sigma: float
        self.___: float
        self.____: float

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.mu = self.params[0].value
            self.sigma = self.params[1].value
            #             self.___ = self.params[2].value
            #             self.____ = self.params[3].value

            self.numerical = True
        else:
            self.mu = 1
            self.sigma = 2

    #                 self.___ = 7
    #                 self.____ = 9

    def pdf(self, x):
        fz = pm.LogitNormal.dist(mu=self.mu, sigma=self.sigma)
        out = pm.logp(fz, x).eval()
        return np.e ** (out)
```

```
    @property
    def xmin(self):
        return -0.1

    @property
    def xmax(self):
        return 1.1

    @property
    def distimage(self):
        mi_distrogram: MathImage = FC.fig_path(FC)

        plot_unknown_distrogram(dist=self, mi=mi_distrogram)
        return mi_distrogram.filepath


class Rice(Dist):
    def __init__(self, owner, varname=""):
        self.type = "rice"
        self.needed_params = [
            partup("b", 3, "shape"),
            partup("sigma", 4, "scale"),
            #             partup("", 5, ""),
            #             partup("", 6, ""),
        ]
        super().__init__(owner, varname)

        self.b: float
        self.sigma: float
        self.___: float
        self.____: float

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.b = self.params[0].value
            self.sigma = self.params[1].value
            #             self.___ = self.params[2].value
            #             self.____ = self.params[3].value

            self.numerical = True
        else:
            self.b = 3
            self.sigma = 5
        #                 self.___ = 7
        #                 self.____ = 9

        self.fz: sps.rv_continuous = sps.rice

    def ppf(self, x):
        return self.fz.ppf(x, b=self.b, scale=self.sigma)

    def pdf(self, x):
        return self.fz.pdf(x, b=self.b, scale=self.sigma)
```

```
class Moyal(Dist):
    def __init__(self, owner, varname=""):
        self.type = "Moyal"
        self.needed_params = [
            partup("mu", 3, "location"),
            partup("sigma", 4, "scale"),
            #             partup("", 5, ""),
            #             partup("", 6, ""),
        ]
        super().__init__(owner, varname)

        self.mu: float
        self.sigma: float
        self.___: float
        self.____: float

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.mu = self.params[0].value
            self.sigma = self.params[1].value
            #             self.___ = self.params[2].value
            #             self.____ = self.params[3].value

            self.numerical = True
        else:
            self.mu = 3
            self.sigma = 5
        #                 self.___ = 7
        #                 self.____ = 9

        self.fz: sps.rv_continuous = sps.moyal

    def ppf(self, x):
        return self.fz.ppf(x, loc=self.mu, scale=self.sigma)

    def pdf(self, x):
        return self.fz.pdf(x, loc=self.mu, scale=self.sigma)


class AsymmetricLaplace(Dist):
    def __init__(self, owner, varname=""):
        self.type = "Asymmetric Laplace"
        self.needed_params = [
            partup("kappa", 4, "symmetry"),
            partup("mu", 5, "location"),
            partup("b", 3, "scale"),
            #             partup("", 6, ""),
        ]
        super().__init__(owner, varname)

        self.kappa: float
        self.mu: float
        self.b: float
        self.____: float
```

```
        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.kappa = self.params[0].value
            self.mu = self.params[1].value
            self.b = self.params[2].value
            #             self.____ = self.params[3].value

            self.numerical = True
        else:
            self.kappa = 3
            self.mu = 5
            self.b = 7
        #                 self.____ = 9

        self.fz: sps.rv_continuous = sps.laplace_asymmetric

    def ppf(self, x):
        return self.fz.ppf(x, kappa=self.kappa, loc=self.mu, scale=self.b)

    def pdf(self, x):
        return self.fz.pdf(x, kappa=self.kappa, loc=self.mu, scale=self.b)


class DiscreteDist(Dist):
    def pmf(self, x):
        return self.fz.pmf(x)

    @property
    def distimage(self):
        mi_distrogram: MathImage = FC.fig_path(FC)

        plot_discrete_distrogram(
            dist=self,
            mi=mi_distrogram,
            plot_posterior=self.plot_posterior,
            posterior_data=self.posterior_data,
        )
        return mi_distrogram.filepath

    @property
    def xmin(self):
        return self.ppf(0.01)

    @property
    def xmax(self):
        return self.ppf(0.99)


class DiscreteUniform(DiscreteDist):
    def __init__(self, owner, varname=""):
        self.type = "Discrete Uniform"
        self.needed_params = [
            partup("", 3, "lower"),
            partup("", 4, "upper"),
```

```
        ]
        super().__init__(owner, varname)

        self.lower: float
        self.upper: float

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.lower = self.params[0].value
            self.upper = self.params[1].value
            self.numerical = True

        else:
            self.lower = 3
            self.upper = 5

        self.fz: sps.rv_discrete = sps.randint

    def ppf(self, x):
        return self.fz.ppf(x, low=self.lower, high=self.upper)

    def pmf(self, x):
        results = self.fz.pmf(x, low=self.lower, high=self.upper)
        return results

    @property
    def xmin(self):
        return self.lower

    @property
    def xmax(self):
        return self.upper


class Binomial(DiscreteDist):
    def __init__(self, owner, varname=""):
        self.type = "Binomial"
        self.needed_params = [
            partup("n", 3, "trials"),
            partup("p", 4, "P(success)"),
            #             partup("", 5, ""),
            #             partup("", 6, ""),
        ]
        super().__init__(owner, varname)

        self.n: float
        self.p: float
        self.___: float
        self.____: float

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.n = self.params[0].value
            self.p = self.params[1].value
            #             self.___ = self.params[2].value
```

```
            #             self.____ = self.params[3].value

            self.numerical = True
        else:
            self.n = 10
            self.p = 0.3
        #                 self.___ = 7
        #                 self.____ = 9

        self.fz: sps.rv_discrete = sps.binom

    def ppf(self, x):
        return self.fz.ppf(x, n=self.n, p=self.p)

    def pmf(self, x):
        return self.fz.pmf(x, n=self.n, p=self.p)

    @property
    def xmin(self):
        return self.ppf(0.01)

    @property
    def xmax(self):
        return self.ppf(0.99)


class BetaBinomial(DiscreteDist):
    def __init__(self, owner, varname=""):
        self.type = "Beta Binomial"
        self.needed_params = [
            partup("n", 3, "trials"),
            partup("alpha", 4, "shape"),
            partup("beta", 5, "shape"),
            #             partup("", 6, ""),
        ]
        super().__init__(owner, varname)

        self.n: float
        self.alpha: float
        self.beta: float
        self.____: float

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.n = self.params[0].value
            self.alpha = self.params[1].value
            self.beta = self.params[2].value
            #             self.____ = self.params[3].value

            self.numerical = True
        else:
            self.n = 3
            self.alpha = 0.5
            self.beta = 7
```

```
        #                 self.____ = 9

        self.fz: sps.rv_discrete = sps.betabinom

    def ppf(self, x):
        return self.fz.ppf(x, n=self.n, a=self.alpha, b=self.beta)

    def pmf(self, x):
        return self.fz.pmf(x, n=self.n, a=self.alpha, b=self.beta)


class Bernoulli(DiscreteDist):
    def __init__(self, owner, varname=""):
        self.type = "Bernoulli"
        self.needed_params = [
            partup("p", 3, "probability"),
            #             partup("", 5, ""),
            #             partup("", 6, ""),
        ]
        super().__init__(owner, varname)

        self.p: float
        self.__: float
        self.___: float
        self.____: float

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.p = self.params[0].value
            #             self.___ = self.params[2].value
            #             self.____ = self.params[3].value

            self.numerical = True
        else:
            self.p = 0.7
        #                 self.___ = 7
        #                 self.____ = 9

        self.fz: sps.rv_discrete = sps.bernoulli

    def ppf(self, x):
        return self.fz.ppf(x, p=self.p)

    def pmf(self, x):
        return self.fz.pmf(x, p=self.p)
```

(cont'd)

```
class Poisson(DiscreteDist):
    def __init__(self, owner, varname=""):
        self.type = "Poisson"
        self.needed_params = [
            partup("mu", 3, "occurrences"),
            #             partup("", 5, ""),
            #             partup("", 6, ""),
```

```
        ]
        super().__init__(owner, varname)

        self.mu: float
        self.__: float
        self.___: float
        self.____: float

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.mu = self.params[0].value
            #             self.___ = self.params[2].value
            #             self.____ = self.params[3].value

            self.numerical = True
        else:
            self.mu = 3
        #                 self.___ = 7
        #                 self.____ = 9

        self.fz: sps.rv_discrete = sps.poisson

    def ppf(self, x):
        return self.fz.ppf(x, mu=self.mu)

    def pmf(self, x):
        return self.fz.pmf(x, mu=self.mu)


class NegativeBinomial(DiscreteDist):
    def __init__(self, owner, varname=""):
        self.type = "Negative Binomial"
        self.needed_params = [
            partup("n", 3, "successes"),
            partup("p", 4, "probability"),
            #             partup("", 5, ""),
            #             partup("", 6, ""),
        ]
        super().__init__(owner, varname)

        self.n: float
        self.p: float
        self.___: float
        self.____: float

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.n = self.params[0].value
            self.p = self.params[1].value
            #             self.___ = self.params[2].value
            #             self.____ = self.params[3].value

            self.numerical = True
        else:
            self.n = 3
```

```
            self.p = 0.5
        #                 self.___ = 7
        #                 self.____ = 9

        self.fz: sps.rv_discrete = sps.nbinom

    def ppf(self, x):
        return self.fz.ppf(x, n=self.n, p=self.p)

    def pmf(self, x):
        return self.fz.pmf(x, n=self.n, p=self.p)

class Geometric(DiscreteDist):
    def __init__(self, owner, varname=""):
        self.type = "Geometric"
        self.needed_params = [
            partup("p", 3, "probability"),
            #             partup("", 5, ""),
            #             partup("", 6, ""),
        ]
        super().__init__(owner, varname)

        self.p: float
        self.__: float
        self.___: float
        self.____: float

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.p = self.params[0].value
            #             self.___ = self.params[2].value
            #             self.____ = self.params[3].value

            self.numerical = True
        else:
            self.p = 0.4
        #                 self.___ = 7
        #                 self.____ = 9

        self.fz: sps.rv_discrete = sps.geom

    def ppf(self, x):
        return self.fz.ppf(x, p=self.p)

    def pmf(self, x):
        return self.fz.pmf(x, p=self.p)


class Hypergeometric(DiscreteDist):
    def __init__(self, owner, varname=""):
        self.type = "Hypergeometric"
        self.needed_params = [
            partup("N", 3, "population"),
            partup("k", 4, "successes"),
```

```
            partup("n", 5, "samples"),
            #             partup("", 6, ""),
        ]
        super().__init__(owner, varname)

        self.N: float
        self.k: float
        self.n: float
        self.____: float

        if all(isinstance(p, NumericalParameter) for p in self.params):
            self.N = self.params[0].value + self.params[1].value
            self.k = self.params[0].value
            self.n = self.params[2].value
            #             self.____ = self.params[3].value

            self.numerical = True
        else:
            self.N = 25
            self.k = 5
            self.n = 15
        #                 self.____ = 9

        self.fz: sps.rv_discrete = sps.hypergeom

    def ppf(self, x):
        return self.fz.ppf(x, M=self.N, n=self.k, N=self.n)

    def pmf(self, x):
        return self.fz.pmf(x, M=self.N, n=self.k, N=self.n)


dist_dict: dict = {
    "UNKNOWN": UnknownDist,
    "exponential_rv": Exponential,
    "normal_rv": Normal,
    "gamma_rv": Gamma,
    "weibull_rv": Weibull,
    "uniform_rv": Uniform,
    "truncated_normal_rv": TruncatedNormal,
    "beta_rv": Beta,
    "kumaraswamy_rv": Kumaraswamy,
    "laplace_rv": Laplace,
    "studentt_rv": StudentT,
    "cauchy_rv": Cauchy,
    "halfcauchy_rv": HalfCauchy,
    "lognormal_rv": LogNormal,
    "chisquare_rv": ChiSquared,
    "halfnormal_rv": HalfNormal,
    "wald_rv": Wald,
    "pareto_rv": Pareto,
    "invgamma_rv": InverseGamma,
    "exgaussian_rv": ExGaussian,
```

```
    "vonmises_rv": VonMises,
    "skewnormal_rv": SkewNormal,
    "triangular_rv": Triangular,
    "halfstudentt_rv": HalfStudent,
    "gumbel_rv": Gumbel,
    "logistic_rv": Logistic,
    "logit_normal_rv": LogitNormal,
    "rice_rv": Rice,
    "moyal_rv": Moyal,
    "asymmetriclaplace_rv": AsymmetricLaplace,
    "binomial_rv": Binomial,
    "beta_binomial_rv": BetaBinomial,
    "bernoulli_rv": Bernoulli,
    "poisson_rv": Poisson,
    "nbinom_rv": NegativeBinomial,
    "discrete_uniform_rv": DiscreteUniform,
    "geometric_rv": Geometric,
    "hypergeometric_rv": Hypergeometric,
}


def get_rv_dist(node: RandomVar) -> Dist:
    owner = node.owner
    dist_type = str(owner).split("{")[0]

    # Get the distribution type from the dictionary thereof:
    try:
        return dist_dict[dist_type](owner, node.name)
    except KeyError:
        print(f"PyKrusch doesn't recognize {dist_type}")
        unknown_dist = dist_dict["UNKNOWN"](owner, node.name)
        unknown_dist.type = f"{dist_type}"
        if unknown_dist.type.endswith("_rv"):
            unknown_dist.type = unknown_dist.type.rpartition("_rv")[0]
        return unknown_dist
```

## figureControl.py

```
from os import mkdir, remove, rmdir
from pathlib import PosixPath
from pykrusch.config import *


class MathImage:
    def __init__(
        self,
        filepath: PosixPath,
        portnum: int,
        edges_from=[],
        op_id=None,
    ):
        self.filepath = filepath
```

```
        self.portnum = portnum
        self.edges_from: list = edges_from
        self.op_id = op_id


class MathImageFound(MathImage):
    def __init__(
        self,
        filepath: PosixPath,
        portnum: int,
        edges_from=[],
        op_id=None,
    ):
        self.filepath = filepath
        self.portnum = portnum
        self.edges_from: list = edges_from
        self.op_id = op_id


class FC:
    fignum = 0
    temppath = PosixPath(TEMP_IMG_DIR)

    symbol_dict = {}
    other_imgs = []

    def fig_init(fc):
        try:
            mkdir(fc.temppath)
        except FileExistsError:
            pass

    def fig_path(
        fc,
        edges_from=[],
        symbols="",
        op_id=None,
    ) -> MathImage | MathImageFound:
        fc.fignum += 1

        portnum = int(fc.fignum)

        if symbols and symbols in fc.symbol_dict:
            filepath = fc.symbol_dict[symbols]
            return MathImageFound(
                filepath=filepath, portnum=portnum, edges_from=edges_from, op_id=op_id
            )

        filepath = fc.temppath / f"{fc.fignum}.png"

        if symbols:
            fc.symbol_dict[symbols] = filepath
        else:
```

```
            fc.other_imgs.append(filepath)

        return MathImage(
            filepath=filepath, portnum=portnum, edges_from=edges_from, op_id=op_id
        )

    def fig_clean(fc):
        for f in fc.symbol_dict.values():
            try:
                remove(f)
            except FileNotFoundError:
                pass

        for f in fc.other_imgs:
            try:
                remove(f)
            except FileNotFoundError:
                pass

        try:
            rmdir(fc.temppath)
        except FileNotFoundError:
            pass
```

## main.py

```
from __future__ import annotations
from typing import TYPE_CHECKING

import click
from pykrusch.config import *

from shutil import rmtree

if TYPE_CHECKING:
    from pygraphviz import AGraph
    from pymc import Model
    from arviz import InferenceData




# region
def posterior_data_check(
    MODEL: Model,
    posterior_trace: InferenceData | None = None,
    posterior_pickle: str | None = None,
    sample_posterior: bool = False,
):
    too_many = SystemExit(
        """More than one posterior source was passed. Please pass at most
        one of posterior_data, posterior_pickle, or sample_posterior=True."""
```

```
    )

    if posterior_trace and posterior_pickle:
        raise too_many
    if posterior_trace and sample_posterior:
        raise too_many
    if posterior_pickle and sample_posterior:
        raise too_many
    if posterior_trace:
        print(" ")
        print("Using supplied trace data.")
        print(" ")
        return True, posterior_trace
    if posterior_pickle:
        import pickle

        print(" ")
        print("Opening model trace pickle.")
        print(" ")
        with open(posterior_pickle, "rb") as stream:
            return True, pickle.load(stream)
    if sample_posterior:
        print(" ")
        print("Sampling pymc model.")
        print(" ")
        import pymc as pm
        with MODEL:
            return True, pm.sample()

    return False, None


def model_check(
    MODEL,
    model_file,
    model_name,
):
    if not MODEL and not model_file:
        raise SystemExit(
            """Please provide either MODEL (in the form of a pymc Model object), or model_file.
Running pykrusch from the command line requires model_file."""
        )
    if MODEL and model_file:
        raise SystemExit(
            """Cannot pass both MODEL and model_file. Please provide exactly one of MODEL, model_file."
        )
    if model_file and not model_name:
        raise SystemExit("""Use of model_file requires model_name.""")
    if MODEL:
        return MODEL
    if model_file and model_name:
        from importlib.util import spec_from_file_location, module_from_spec
        import sys
```

```
        spec = spec_from_file_location("_MODEL_FROM_FILE", model_file)
        mod = module_from_spec(spec)
        sys.modules["_MODEL_FROM_FILE"] = mod
        spec.loader.exec_module(mod)
        return mod.__dict__[model_name]


@click.command(
    context_settings=dict(help_option_names=["-h", "--help"]),
    help="""Visualize Bayesian model structure.
Takes a Bayesian model specified in pymc and outputs an image portraying the dependency
structure formed by the model's random and deterministic variables, including prior
distribution (and, if specified, posterior distribution).""",
)
@click.argument(
    "model_file",
    type=click.Path(exists=True)
)
@click.option(
    "-n",
    "--model_name",
    default="model",
    help="Name of the model object in model_file. Defaults to 'model'.",
)
@click.option(
    "-p",
    "--posterior_pickle",
    default=None,
    help="Path to the .pkl file containing the sampling trace (only needed for plotting posterior).",
)
@click.option(
    "-s",
    "--sample_posterior",
    is_flag=True,
    default=False,
    help="Flag for sampling the posterior from the model. Can be slow (only needed if plotting posterio
)
@click.option(
    "-o",
    "--outname",
    default="krusch.png",
    help="Path of the output image. Defaults to 'krusch.png'.",
)
@click.option(
    "-r",
    "--retain_figs",
    is_flag=True,
    default=False,
    help="Flag for retaining the intermediary figures (defaults to False).",
)
def main(
    model_file: str | None = None,
    model_name: str | None = None,
```

```
    posterior_pickle: str | None = None,
    sample_posterior: bool = False,
    outname: str = "krusch.png",
    retain_figs=False,
):


    krusch_(
        pymc_model=None,
        model_file=model_file,
        model_name=model_name,
        posterior_trace=None,
        posterior_pickle=posterior_pickle,
        sample_posterior=sample_posterior,
        outname=outname,
        retain_figs=retain_figs,
    )


# endregion


def krusch(
        pymc_model: Model,
        posterior_trace: InferenceData|None = None,
        outname: str = "krusch.png"):
    """Produces a visualization of a `pymc`-specified Bayesian Generalized Linear Model.

    Args:

        pymc_model (Model):
            The `pymc` `Model` object to be visualized.

        posterior_trace (InferenceData | None, optional):
            Optional: argument specifying the `arviz` `InferenceData` object containing the
            posterior information needed to plot the posterior distribution for each viable
            variable in the model. If supplied, `pykrusch` will plot the posterior to the best
            of its ability. If this argument is not supplied (or is `None`), the posterior
            distributions will not be plotted.

        outname (str, optional): _description_.
            Optional: filename of the resulting visualization. Defaults to "krusch.png".
    """
    krusch_(
        pymc_model=pymc_model,
        model_file=None,
        model_name=None,
        posterior_trace=posterior_trace,
        posterior_pickle=None,
        sample_posterior=None,
        outname=outname,
        retain_figs=False,
    )
```

```
def krusch_(
    pymc_model: Model | None = None,
    model_file: str | None = None,
    model_name: str | None = None,
    posterior_trace: InferenceData | None = None,
    posterior_pickle: str | None = None,
    sample_posterior: bool = False,
    outname: str = "krusch.png",
    retain_figs=False,
):

    from pykrusch.parsePymc.createGraph import create_graph
    from pykrusch.figureControl import FC


    MODEL = model_check(pymc_model, model_file, model_name)

    posterior_data: InferenceData | None = None
    plot_posterior: bool = False

    # Get pymc Model object. If directly supplied as argument to
    # MODEL, simply return MODEL. Otherwise, import the file
    # indicated in model_file and grab the model identified by
    # model_name

    plot_posterior, posterior_data = posterior_data_check(
        MODEL=MODEL,
        posterior_trace=posterior_trace,
        posterior_pickle=posterior_pickle,
        sample_posterior=sample_posterior,
    )

    # This initializes the figure numbering system
    # for the temporary figures
    FC.fig_init(FC)

    ###
    # CORE LOOP: Create graph does most of the heavy lifting
    ###
    graph: AGraph = create_graph(
        MODEL,
        plot_posterior,
        posterior_data,
    )
    graph.graph_attr["dpi"] = DPI
    graph.draw(outname, prog=ENGINE)

    if not retain_figs:
        rmtree(TEMP_IMG_DIR)
```

## param.py

```
from pytensor.graph.basic import Apply
from pykrusch.config import PARAM_CONVERSION, PARAM_FONT_SIZE, PARAM_FONT_SIZE_NUMERICAL
from pykrusch.figureControl import MathImage, FC, MathImageFound
from pykrusch.graphviz.latexText import render_img_from_latex, string_to_latex


class Parameter:
    def __init__(self):
        self.name: str
        self.meaning: str
        self.printout: str
        self.type: str

        # This gets supplied in dist.py after instantiation
        self.greek_name: str
        self.slot: int

    def __str__(self):
        return self.__repr__()

    def __repr__(self):
        return f"({self.type}, {self.printout})"

    def give_greek(self, greek):
        if greek in PARAM_CONVERSION:
            self.greek_name = PARAM_CONVERSION[greek]
        else:
            self.greek_name = greek

    def give_slot(self, slot):
        self.slot = slot

    def give_symbol(self):
        return self.greek_name, PARAM_FONT_SIZE

    def give_meaning(self, meaning):
        self.meaning = meaning


class NumericalParameter(Parameter):
    def __init__(self, value):
        self.name = ""
        self.value = value
        self.printout = str(value)
        self.type = "NumericalParameter"

    def give_symbol(self):
        if float(self.value).is_integer():
            numerical_arg = int(float(self.value))
        else:
            numerical_arg = float(self.value)
```

```
        if self.greek_name:
            return self.greek_name + f" = {numerical_arg}", PARAM_FONT_SIZE_NUMERICAL
        else:
            return f"{numerical_arg}", PARAM_FONT_SIZE_NUMERICAL


class NamedParameter(Parameter):
    def __init__(self, name):
        self.name = name
        self.printout = str(name)
        self.type = "NamedParameter"


class NamedData(Parameter):
    def __init__(self, name):
        self.name = name
        self.printout = str(name)
        self.type = "NamedData"

        mi: MathImage = FC.fig_path(FC, symbols=self.name)
        if not isinstance(mi, MathImageFound):
            render_img_from_latex(string_to_latex(self.name), mi.filepath)
        self.mi = mi


class DataParameter(Parameter):
    def __init__(self):
        self.name = ""
        # TODO: THIS NEEDS TO BE CHANGED
        self.printout = "DATA_PARAMETER"
        self.type = "DataParameter"


class UnknownParameter(Parameter):
    def __init__(self):
        self.name = ""
        self.printout = "UNKNOWN_PARAMETER"
        self.type = "UnknownParameter"


class ComponentsParameter(Parameter):
    def __init__(self, components):
        self.components = components
        self.name = [c.name for c in self.components]
        self.type = [c.type for c in self.components]
        self.printout = str(self.name)
        self.flat_components: set()

    def __str__(self):
        return self.__repr__()

    def __repr__(self):
        return f"({self.type}, {self.printout})"
```

```
    def give_slot(self, slot):
        self.slot = slot
        for component in self.flat_components:
            component: Parameter
            component.give_slot(slot)

    def give_greek(self, greek):
        self.greek_name = greek
        if greek in PARAM_CONVERSION:
            self.greek_name = PARAM_CONVERSION[greek]
        else:
            self.greek_name = greek

    def give_meaning(self, meaning):
        self.meaning = meaning
        for component in self.flat_components:
            component: Parameter
            component.give_meaning(meaning)

    def flatten_components(self):
        out_set = set()
        for component in self.components:
            if isinstance(component, ComponentsParameter):
                out_set = out_set.union(component.flatten_components())
            else:
                out_set.add(component)

        self.flat_components = out_set

        self.name = [c.name for c in self.flat_components]
        self.type = [c.type for c in self.flat_components]
        self.printout = str(self.name)
        return out_set


def recursive_rv_search(node) -> Parameter:
    # First, attempt to retrieve the node name, if there is one, return it

    parents = node.get_parents()

    try:
        if node.name:
            if hasattr(node, "data") or len(parents) == 0:
                return NamedData(node.name)
            return NamedParameter(node.name)
    except AttributeError:
        pass

    # If we make it past the try block above, it means the node
    # has no name.
    # If an unnamed node has 0 parents, it's either a parameter or data.
    # If it's a parameter, we should be able to retrieve it using
    # `node.data.item()`. If that fails, it's unnamed data.
```

```
    # So, let's get the parents:

    # Does the node have 0 parents? If so, it's either
    # a scalar (usually a parameter in a distribution)
    # or it's data. If it's a scalar, get it and return it.
    # If it's data, we simply call it 'x' for now.
    ###
    # TODO: Consider creating a better representation for data
    # Some kind of class, perhaps?
    ###
    if len(parents) == 0:
        try:
            return NumericalParameter(node.data.item())
        except ValueError:
            return DataParameter()

    # If the node has one parent, we go deeper.
    if len(parents) == 1:
        return recursive_rv_search(parents[0])

    # If the node is a math operation, has multiple parents,
    # and all of the parents are scalars, we'll try evaluating
    # to a scalar:
    if len(parents) == 2:
        try:
            parents[0].data.item()
            parents[1].data.item()
            # If the node type is an 'Apply' pytensor node,
            # We want to evaluate the apply node's CHILD.
            if type(node) == Apply:
                return NumericalParameter(float(node.outputs[0].eval()))
            else:
                return NumericalParameter(float(node.eval()))
        except NameError:
            pass
        except ValueError:
            pass
        except AttributeError:
            pass

    # If the above doesn't work, we check to see if the node is a
    # subtensor, which means that it's probably going to split a named
    # node into more than one component.

    # TODO: Improve handling here: currently, subtensor handling
    # simply treats them as invisible.

    if len(parents) == 2 and str(node).startswith("Subtensor"):
        return recursive_rv_search(parents[0])

        pass

    # If the node has multiple parents, we switch to a list of parameters/inputs.
```

```
    # This is just for debugging purposes.

    comp_list = []
    for parent in parents:
        result = recursive_rv_search(parent)
        comp_list.append(result)

    comp = ComponentsParameter(comp_list)
    comp.flatten_components()

    print(comp.flat_components)

    return comp
```

## graphviz/distrogramsMPL.py

```
from __future__ import annotations
import matplotlib.pyplot as plt
import numpy as np
from scipy import stats
from typing import TYPE_CHECKING
from pykrusch.config import *
from matplotlib.ticker import MaxNLocator

if TYPE_CHECKING:
    from pykrusch.figureControl import MathImage
    from pykrusch.dist import Dist, DiscreteDist
    from arviz.data import InferenceData


def plot_distrogram(
    dist: Dist,
    mi: MathImage,
    plot_posterior: bool = False,
    posterior_data: InferenceData | None = None,
):
    plt.style.use("default")
    xmin = dist.xmin
    xmax = dist.xmax
    ls = np.linspace(xmin, xmax, LINSPACE_N)
    plt.plot(ls, dist.pdf(ls), linewidth=DISTROGRAM_LINE_WIDTH)

    if plot_posterior and dist.numerical and dist.scipy:
        data_vars = list(posterior_data.posterior.data_vars)

        if dist.varname in data_vars:
            vals = posterior_data.posterior[dist.varname].values
            combined_trace = np.concatenate(vals)

            if combined_trace.ndim == 1:
                range_buffer = abs(max(combined_trace) - min(combined_trace)) * 0.1
```

```
                pxmin = min(combined_trace) - range_buffer
                pxmax = max(combined_trace) + range_buffer

                pls = np.linspace(pxmin, pxmax, LINSPACE_N)

                kde = stats.gaussian_kde(combined_trace)

                plt.plot(pls, kde.evaluate(pls), linewidth=DISTROGRAM_LINE_WIDTH)

            else:
                print(f"{dist.varname} did not have exactly one dimension")

        else:
            print(f"{dist.varname} not found in posterior data variables!")

    ax = plt.gca()
    ax.get_yaxis().set_visible(False)
    ax.spines["left"].set_visible(False)
    ax.spines["bottom"].set_visible(False)
    ax.spines["right"].set_visible(False)
    ax.spines["top"].set_visible(False)

    ax.tick_params(axis="both", which="major", labelsize=15)

    if (xmin - 1) < 0 and 0 < (xmax + 1) and dist.numerical and dist.scipy:
        plt.axvline(x=0, alpha=0.1, c="black", linewidth=DISTROGRAM_LINE_WIDTH)

    if not dist.numerical or not dist.scipy:
        plt.axis("off")

    plt.savefig(
        mi.filepath,
        bbox_inches="tight",
        dpi=DPI,
        pad_inches=0,
    )

    plt.close()


def plot_discrete_distrogram(
    dist: Dist | DiscreteDist,
    mi: MathImage,
    plot_posterior: bool = False,
    posterior_data: InferenceData | None = None,
):
    plt.style.use("default")
    xmin = dist.xmin
    xmax = dist.xmax + 1

    ar = np.arange(xmin, xmax)

    plt.bar(
```

```
        ar,
        dist.pmf(ar),
        linewidth=DISCRETE_LINE_WIDTH,
        color="None",
        edgecolor="tab:blue",
    )

    plt.bar([xmin], [-0.00001], color="white")

    if plot_posterior and dist.numerical and dist.scipy:
        data_vars = list(posterior_data.posterior.data_vars)

        if dist.varname in data_vars:
            combined_trace = np.concatenate(
                posterior_data.posterior[dist.varname].values
            )

            pxmin = min(combined_trace)
            pxmax = max(combined_trace) + 1
            par = np.arange(pxmin, pxmax)

            unique, counts = np.unique(combined_trace, return_counts=True)

            ardict = {}

            for i in par:
                ardict[i] = 0
                if i in unique:
                    ardict[i] += float(counts[np.where(unique == i)] / sum(counts))

            plt.bar(
                par,
                list(ardict.values()),
                linewidth=2,
                color="None",
                edgecolor="tab:orange",
                width=0.7,
            )

        else:
            print(f"{dist.varname} not found in posterior data variables!")

    ax = plt.gca()
    ax.xaxis.set_major_locator(MaxNLocator(integer=True))
    ax.get_yaxis().set_visible(False)
    ax.spines["left"].set_visible(False)
    ax.spines["bottom"].set_visible(False)
    ax.spines["right"].set_visible(False)
    ax.spines["top"].set_visible(False)

    ax.tick_params(axis="both", which="major", labelsize=15)

    if (xmin - 1) < 0 and 0 < (xmax + 1) and dist.numerical and dist.scipy:
        plt.axvline(x=0, alpha=0.1, c="black", linewidth=DISTROGRAM_LINE_WIDTH)
```

```
    if not dist.numerical or not dist.scipy:
        plt.axis("off")

    plt.savefig(
        mi.filepath,
        bbox_inches="tight",
        dpi=DPI,
        pad_inches=0,
    )

    plt.close()


def plot_unknown_distrogram(dist: Dist, mi: MathImage):
    plt.style.use("default")
    fig = plt.figure()

    ls = np.linspace(-3, 3, LINSPACE_N)
    plt.plot(
        ls,
        stats.norm.pdf(ls, loc=0, scale=1),
        linewidth=DISTROGRAM_LINE_WIDTH,
        c="white",
    )

    text = fig.text(0.5, 0.5, s="?", ha="center", va="center", size=70, c="tab:blue")
    ax = plt.gca()
    ax.get_yaxis().set_visible(False)
    ax.spines["left"].set_visible(False)
    ax.spines["bottom"].set_visible(False)
    ax.spines["right"].set_visible(False)
    ax.spines["top"].set_visible(False)
    plt.axis("off")
    plt.savefig(
        mi.filepath,
        bbox_inches="tight",
        dpi=DPI,
        pad_inches=0,
    )
    plt.close()
```

## graphviz/latexText.py

```
import matplotlib.pyplot as plt
from pykrusch.config import PARAM_CONVERSION
from re import split
from pykrusch.config import *


def string_to_latex(math_str) -> str:
    parts = []
    # Need to make sure that escaped '_' symbols aren't split:
```

```
    if LATEX_NAMES:
        for str_part in split(r"(?<!\\)_", math_str):
            if str_part in PARAM_CONVERSION:
                parts.append(PARAM_CONVERSION[str_part])
            else:
                parts.append(str_part)

        if len(parts) == 1:
            latex_str = rf"{parts[0]}"
        else:
            the_rest = "\_".join(parts[1:])
            latex_str = rf"{parts[0]}_{{{the_rest}}}"

        return latex_str

    else:
        return math_str.replace("\\", "").replace("{", "").replace("}", "")


def render_img_from_latex(latex_str: str, filepath):
    plt.style.use("default")

    plt.rcParams["mathtext.fontset"] = "cm"

    if LATEX_NAMES:
        mpl_string = rf"${latex_str}$"
    else:
        mpl_string = rf"{latex_str}".replace("\\", "").replace("{", "").replace("}", "")

    fig = plt.figure()
    fig.patch.set_facecolor(NODE_COLOUR_MPL)
    text = fig.text(0.5, 0.5, s=rf"{mpl_string}", ha="center", va="center", size=15)
    plt.savefig(
        filepath,
        bbox_inches="tight",
        dpi=DPI,
        pad_inches=0,
    )
    plt.close()
```

## graphviz/pgvHTML.py

```
from __future__ import annotations
from PIL import Image
from typing import TYPE_CHECKING
from pygraphviz import AGraph
from pykrusch.config import *

if TYPE_CHECKING:
    from pykrusch.figureControl import MathImage
    from pykrusch.parsePymc.randomVariable import RandomVar
    from pykrusch.parsePymc.deterministicVariable import DeterministicVar
```

```
    from pykrusch.dist import Dist
    from pykrusch.param import NamedParameter, NamedData


def make_data_html(param: NamedData, graph):
    label_start = f"""<
    <TABLE BORDER="20" CELLBORDER="2" CELLSPACING="0" VALIGN="BOTTOM">
    <TR>
    """
    label_end = f"""</TR></TABLE>>"""

    label = """"""

    label += label_start

    image = Image.open(param.mi.filepath)
    tw = image.width * DETERMINISTIC_TEXT_SCALE
    th = image.height * DETERMINISTIC_TEXT_SCALE
    label += f"""<TD FIXEDSIZE="TRUE" WIDTH="{tw}" HEIGHT="{th}"
    PORT="{param.mi.portnum}"><IMG SCALE="BOTH" SRC="{param.mi.filepath}">
    </IMG></TD>"""

    label += label_end

    return label


def make_f_html(img_list: list[MathImage], graph: AGraph, f_node: DeterministicVar):
    label_start = f"""<
    <TABLE BORDER="20" CELLBORDER="2" CELLSPACING="0" VALIGN="BOTTOM">
    <TR>
    """
    label_end = f"""</TR></TABLE>>"""

    label = """"""

    label += label_start

    for img in img_list:
        image = Image.open(img.filepath)
        tw = image.width * DETERMINISTIC_TEXT_SCALE
        th = image.height * DETERMINISTIC_TEXT_SCALE
        label += f"""<TD FIXEDSIZE="TRUE" WIDTH="{tw}" HEIGHT="{th}" PORT="{img.portnum}">
        <IMG SCALE="BOTH" SRC="{img.filepath}"></IMG></TD>"""
        for edge_from in img.edges_from:
            if edge_from and graph.has_edge(edge_from, f_node.name, key=img.op_id):
                e = graph.get_edge(edge_from, f_node.name, key=img.op_id)
                e.attr["headport"] = str(img.portnum) + ":n"
                e.attr["tailport"] = "s"

    label += label_end

    return label
```

```
def make_rv_html(rv: RandomVar, graph: AGraph):
    dist: Dist = rv.dist
    tw = DIST_IMAGE_WIDTH * IMAGE_SCALE_FACTOR
    th = DIST_IMAGE_HEIGHT * IMAGE_SCALE_FACTOR

    param_slots = max(dist.num_params, 1)
    slot_width = tw // param_slots

    label_start = f"""<<TABLE BORDER="1" CELLBORDER="0" CELLSPACING="0">
                    <TR>"""

    for param in dist.params:
        param_symbol, font_size = param.give_symbol()

        label_start += f"""
        <TD WIDTH="{slot_width}" PORT="{str(param.slot)}">
            <TABLE CELLPADDING="0" BORDER="0" CELLSPACING="0">
                <TR>
                    <TD><FONT POINT-SIZE="{font_size}">{param_symbol}</FONT></TD>
                </TR>
                <TR>
                    <TD><FONT POINT-SIZE="{DIST_FONT_SIZE_SMALL}">{param.meaning}
                    </FONT></TD>
                </TR>
            </TABLE>
        </TD>"""

    label_start += """</TR>"""

    distimage = dist.distimage

    if distimage:
        label_start += f"""
                <TR>
                    <TD COLSPAN="{param_slots}" WIDTH="{tw}" HEIGHT="{th}">
                    <IMG SCALE="TRUE" SRC="{distimage}"></IMG></TD>
                </TR>"""

    image = Image.open(dist.dist_name_latex)
    tw = image.width * DETERMINISTIC_TEXT_SCALE
    th = image.height * DETERMINISTIC_TEXT_SCALE
    label_end = f"""
                <TR>
                    <TD COLSPAN="{param_slots}" ALIGN="center" HEIGHT="25">
                    <FONT POINT-SIZE="{DIST_FONT_SIZE_SMALL}">{dist.type.upper()}
                    </FONT></TD>

                </TR>
                <TR>
                    <TD WIDTH="{tw}" HEIGHT="{th}" COLSPAN="{param_slots}" ALIGN="center">
                    <IMG SCALE="TRUE" SRC="{dist.dist_name_latex}"></IMG></TD>
                </TR>
```

```
            </TABLE>>"""

    label_start += label_end

    return label_start

    # <TR>
    #     <TD COLSPAN="{param_slots}" ALIGN="center"><FONT POINT-SIZE="{DIST_FONT_SIZE}">{rv.name}</FON
    # </TR>
```

## parsePymc/createGraph.py

```
import pygraphviz as pgv
from pykrusch.parsePymc.randomVariable import specify_rv_nodes, add_rv_nodes
from pykrusch.parsePymc.deterministicVariable import specify_f_nodes, add_f_nodes
from pymc import Model


def create_graph(model: Model, plot_posterior, posterior_data):
    """
    This function serves as a 'main' of sorts: it is resposible for calling
    the other functions needed to get all the nodes in the graph
    and then assemble them into graph format.
    """

    # Start by finding the parameters for each of the
    # free, observed, and deterministic nodes
    rv_nodes = specify_rv_nodes(model.free_RVs, plot_posterior, posterior_data)
    obs_nodes = specify_rv_nodes(
        model.observed_RVs, plot_posterior=False, posterior_data=None
    )
    f_nodes = specify_f_nodes(model.deterministics)

    # Instantiate an empty directed graph
    graph = pgv.AGraph(directed=True, strict=False)

    # Add the nodes to the graph
    add_rv_nodes(rv_nodes, graph)
    add_rv_nodes(obs_nodes, graph)
    add_f_nodes(f_nodes, graph)

    return graph
```

## parsePymc/deterministicVariable.py

```
from pykrusch.figureControl import FC, MathImageFound
from pykrusch.graphviz.latexText import (
    render_img_from_latex,
    string_to_latex,
)
```

```
from pykrusch.graphviz.pgvHTML import make_f_html
from pytensor.graph.basic import Variable
from pygraphviz import AGraph
from pykrusch.config import DATA_LETTER, NODE_COLOUR_GV
import itertools


class DeterministicVar:
    def __init__(self, node: Variable):
        self.name: str = str(node.name)
        self.expression = start_recursive_f_search(node)

        self.shape = "plaintext"
        self.color = NODE_COLOUR_GV
        self.style = "filled"

    def f_add_self_to_graph(self, graph: AGraph):
        img_list = self.self_img_equals()
        img_list += self.expression.math_img_list()
        self.expression.op_to_graph(graph, self.name)
        graph.add_node(
            self.name,
            shape=self.shape,
            color=self.color,
            style=self.style,
            label=make_f_html(img_list, graph, self),
        )

    def self_img_equals(self):
        self_sanitized_name = self.name.replace("_", "\_")
        latex_str = string_to_latex(self_sanitized_name)
        mi_self = FC.fig_path(FC, symbols=latex_str)
        if not isinstance(mi_self, MathImageFound):
            render_img_from_latex(latex_str, mi_self.filepath)
        latex_str_eq = string_to_latex(r"=")
        mi_equals = FC.fig_path(FC, symbols=latex_str_eq)
        if not isinstance(mi_equals, MathImageFound):
            render_img_from_latex(latex_str_eq, mi_equals.filepath)
        return [mi_self, mi_equals]


class MathOP:
    OP_ID = itertools.count()

    def __init__(self, apply_parents, single_arg=False):
        self.op = []
        self.arg1 = recursive_f_search(apply_parents[0])
        if single_arg:
            self.arg2 = None
        else:
            self.arg2 = recursive_f_search(apply_parents[1])
        self.pre = []
        self.post = []
```

```
        self.op_id = "math" + str(MathOP.OP_ID.__next__())
        self.print_arg2 = True

    def __repr__(self):
        if self.arg2:
            return str(f" [{self.arg1} {self.op} {self.arg2}] ")

        else:
            return str(f" [{self.op}{self.arg1}] ")

    def __str__(self):
        return self.__repr__()

    def op_to_graph(self, graph, f_node_name):
        self.arg1.op_to_graph(graph, f_node_name)
        if self.arg2:
            self.arg2.op_to_graph(graph, f_node_name)

    def math_img_list(self) -> list:
        img_list = []
        for i in self.pre:
            img_list.append(self.self_img(i))
        img_list += self.arg1.math_img_list()
        for ii in self.op:
            img_list.append(self.self_img(ii))
        if self.arg2 and self.print_arg2:
            img_list += self.arg2.math_img_list()
        for iii in self.post:
            img_list.append(self.self_img(iii))
        return img_list

    def self_img(self, symbols):
        latex_str = string_to_latex(symbols)
        mi = FC.fig_path(FC, symbols=latex_str)
        if not isinstance(mi, MathImageFound):
            render_img_from_latex(latex_str, mi.filepath)
        return mi

    def latex_as_data(self):
        raise NotImplementedError

    def latex_as_subset(self):
        raise NotImplementedError

    def latex_as_variable(self):
        raise NotImplementedError


### Classes that inherit from MathOps
# region


class Add(MathOP):
```

```
    def __init__(self, apply_parents):
        super().__init__(apply_parents)
        self.op = ["+"]


class Subtract(MathOP):
    def __init__(self, apply_parents):
        super().__init__(apply_parents)
        self.op = ["-"]


class Multiply(MathOP):
    def __init__(self, apply_parents):
        super().__init__(apply_parents)


class Divide(MathOP):
    def __init__(self, apply_parents):
        super().__init__(apply_parents)
        self.pre = ["("]
        self.op = [")", "/", "("]
        self.post = [")"]

class Power(MathOP):
    def __init__(self, apply_parents):
        super().__init__(apply_parents)

        n = "x"

        try:
            n = self.arg2.value
            if n.is_integer():
                n = int(n)
            self.print_arg2 = False
            self.pre = ["("]
            self.post = [f")^{{{n}}}"]
            return
        except AttributeError:
            pass
        try:
            n = self.arg2.name
            self.pre = ["("]
            self.op = [")","^\wedge","("]
            self.post = [f")"]
            return
        except AttributeError:
            pass

        self.print_arg2 = False
        self.pre = ["("]
        self.post = [f")^{{{n}}}"]
```

```
class Exp(MathOP):
    def __init__(self, apply_parents):
        super().__init__(apply_parents, single_arg=True)
        self.pre = [
            r"\mathrm{exp}",
            "(",
        ]
        self.post = [")"]


class SubSet(MathOP):
    def __init__(self, apply_parents):
        super().__init__(apply_parents)
        self.op = []

    # Even if the variable subset is named, we don't want
    # to create a node for it in the graph, so we only
    # ever pass the recursive function along to the first
    # node.
    def op_to_graph(self, graph, f_node_name):
        self.arg1.op_to_graph(graph, f_node_name)

    # Subset is going to have to be special
    def math_img_list(self) -> list:
        return [self.self_img()]

    def self_img(self):
        name1 = self.arg1.latex_as_variable()
        name2 = self.arg2.latex_as_subset()

        if r"_" in name1 and r"{" in name1 and r"}" in name1:
            latex_str = name1.replace(r"}", "") + rf"[{name2}]" + r"}"
        else:
            latex_str = name1 + rf"_{{[{name2}]}}"

        mi = FC.fig_path(
            FC, edges_from=[self.arg1.name], symbols=latex_str, op_id=self.arg1.op_id
        )
        if not isinstance(mi, MathImageFound):
            render_img_from_latex(latex_str, mi.filepath)

        return mi


class Logarithm(MathOP):
    def __init__(self, apply_parents):
        super().__init__(apply_parents, single_arg=True)
        self.pre = [
            "ln",
            "(",
        ]
        self.post = [")"]
```

```
class Logarithm2(MathOP):
    def __init__(self, apply_parents):
        super().__init__(apply_parents, single_arg=True)
        self.pre = [
            "log_2",
            "(",
        ]
        self.post = [")"]


class Logarithm10(MathOP):
    def __init__(self, apply_parents):
        super().__init__(apply_parents, single_arg=True)
        self.pre = [
            r"log_10",
            "(",
        ]
        self.post = [")"]


class Identity(MathOP):
    def __init__(self, apply_parents):
        super().__init__(apply_parents, single_arg=True)
        self.op = []

    def math_img_list(self):
        return self.arg1.math_img_list()

    @property
    def value(self):
        return self.arg1.value

class Subtensor(MathOP):
    def __init__(self, apply_parents):
        super().__init__(apply_parents)
        subset = int(self.arg2.value)
        self.op = [rf"_[{subset}]"]
        #TODO: Consider changing how Subtensor MathOP handles arg2.
        self.arg2 = None


class Reciprocal(MathOP):
    def __init__(self, apply_parents):
        super().__init__(apply_parents, single_arg=True)
        self.op = ["1/"]


class InverseLogit(MathOP):
    def __init__(self, apply_parents):
        super().__init__(apply_parents, single_arg=True)
        self.pre = [r"\mathrm{invlogit}", "("]
        self.post = [")"]
```

```
class Dot(MathOP):
    def __init__(self, apply_parents):
        super().__init__(apply_parents)
        self.op = ["."]


class UnknownOp(MathOP):
    def __init__(self, apply_parents):
        super().__init__(apply_parents, single_arg=True)
        try:
            self.arg2 = recursive_f_search(apply_parents[1])
        except:
            pass
        self.pre = ["("]
        self.op = ["?"]
        self.post = [")"]


# TODO: Add all math operations
mathop_dict = {
    # Old Pytensor Syntax: retained for now, might no longer be useful.
    "Elemwise{add,": {"op": Add},
    "Elemwise{mul,": {"op": Multiply},
    "Elemwise{true_div,": {"op": Divide},
    "Elemwise{pow,": {"op": Power},
    "Elemwise{exp,": {"op": Exp},
    "Elemwise{identity}": {"op": Identity},
    "InplaceDimShuffle": {"op": Identity},
    "dot": {"op": Dot},
    "Sum{": {"op": Identity},
    "Subtensor{": {"op": SubSet},
    "Elemwise{log10,": {"op": Logarithm10},
    "Elemwise{log2,": {"op": Logarithm2},
    "Elemwise{log,": {"op": Logarithm},
    "AdvancedSubtensor": {"op": SubSet},
    "Elemwise{sigmoid,": {"op": InverseLogit},

    # New Pytensor Syntax:

    "Add": {"op": Add},
    "Mul": {"op": Multiply},
    "Sub": {"op": Subtract},
    "True_div": {"op": Divide},
    "Int_div": {"op": Divide},
    "Reciprocal": {"op": Reciprocal},
    "Pow": {"op": Power},
    "Exp": {"op": Exp},
    "Identity": {"op": Identity},
    "InplaceDimShuffle": {"op": Identity},
    "dot": {"op": Dot},
    "Sum{": {"op": Identity},
    "Subtensor{": {"op": SubSet},
    "Log10": {"op": Logarithm10},
```

```
    "Log2": {"op": Logarithm2},
    "Log.": {"op": Logarithm}, # Period is there to differentiate from other log bases
                               # It seems as though all OP names are followed by periods.
    "AdvancedSubtensor": {"op": SubSet},
    "Sigmoid": {"op": InverseLogit},
    "ExpandDims": {"op": Identity},


    "UNKNOWN": {"op": UnknownOp},
}


# endregion


### Other OPs that can result from a recursive f search:


class OtherOP:
    OP_ID = itertools.count()

    def __init__(self):
        self.name: str
        self.op_id = "other" + str(OtherOP.OP_ID.__next__())

    def __repr__(self):
        return str(self.name)

    def __str__(self):
        return self.__repr__()

    def op_to_graph(self, graph: AGraph, f_node_name):
        if graph.has_node(self.name):
            graph.add_edge(self.name, f_node_name, key=self.op_id)

    def self_img(self):
        raise NotImplementedError

    def math_img_list(self) -> list:
        return [self.self_img()]

    def latex_as_data(self):
        return string_to_latex(f"{DATA_LETTER}_{self.name}")

    # Replacing underscores with escaped underscores ensures
    # That no subsetting will happen in the string
    def latex_as_subset(self):
        return string_to_latex(self.name.replace("_", "\_"))

    def latex_as_variable(self):
        return string_to_latex(self.name)

    def latex_as_op(self):
```

```
        raise NotImplementedError

# region
class UnnamedDataOP(OtherOP):
    # This is a class attribute used to keep track of each instance of DataOp that has
    # been added to the graph; the effect will be purely cosmetic, but it will help differentiate
    # the various forms of data
    data_iterator = 1

    def __init__(self):
        super().__init__()
        self.name = f"{UnnamedDataOP.data_iterator}"
        UnnamedDataOP.data_iterator += 1

    def op_to_graph(self, graph, f_node_name):
        pass

    def self_img(self):
        mi = FC.fig_path(FC, symbols=self.latex_as_data(), op_id=self.op_id)
        if not isinstance(mi, MathImageFound):
            render_img_from_latex(self.latex_as_data(), mi.filepath)
        return mi


class NamedDataOP(OtherOP):
    def __init__(self, name):
        super().__init__()
        self.name = f"{name}"

    def self_img(self):
        mi = FC.fig_path(
            FC,
            edges_from=[str(self.name)],
            symbols=self.latex_as_data(),
            op_id=self.op_id,
        )
        if not isinstance(mi, MathImageFound):
            render_img_from_latex(self.latex_as_data(), mi.filepath)
        return mi


class NamedOP(OtherOP):
    def __init__(self, name):
        super().__init__()
        self.name = str(name)

    def self_img(self):
        mi = FC.fig_path(
            FC,
            edges_from=[str(self.name)],
            symbols=self.latex_as_variable(),
            op_id=self.op_id,
        )
```

```
        if not isinstance(mi, MathImageFound):
            render_img_from_latex(self.latex_as_variable(), mi.filepath)
        return mi


class ScalarOP(OtherOP):
    def __init__(self, value):
        super().__init__()
        self.value = value
        if float(self.value).is_integer():
            self.name = str(int(self.value))
        else:
            self.name = str(self.value)

    def __repr__(self):
        return str(self.value)

    def __str__(self):
        return self.__repr__()

    def op_to_graph(self, graph, f_node_name):
        pass

    def self_img(self):
        mi = FC.fig_path(
            FC,
            edges_from=[str(self.name)],
            symbols=self.latex_as_variable(),
            op_id=self.op_id,
        )
        if not isinstance(mi, MathImageFound):
            render_img_from_latex(str(self.value), mi.filepath)
        return mi


# endregion

def recursive_f_search(node: Variable) -> MathOP | ScalarOP | OtherOP:
    # Get the node's number of parents; we'll need to know it later.
    lenparents = len(node.get_parents())

    # If the node has a name, it's either named data or a named random variable.
    # If it has no parents, it's almost certainly data.
    # Otherwise, it's a random variable.
    try:
        if node.name and lenparents == 0:
            return NamedDataOP(str(node.name))

        if node.name and lenparents >= 0:
            return NamedOP(str(node.name))

    except AttributeError:
        pass
```

```
    # If the node has no name and no parents, we can be certain that it's
    # unnamed data -- we merely need to decide if it's a scalar (in which
    # case it's almost certainly a parameter), or if it's unnamed data

    if lenparents == 0:
        try:
            return ScalarOP(float(node.eval()))
        except TypeError:
            return UnnamedDataOP()

    mathop_type = None

    for type in mathop_dict.keys():
        if str(node).startswith(type):
            mathop_type = mathop_dict[type]
            continue

    if not mathop_type:
        print(f"Pykrusch didn't recognize {str(node)} as a valid MathOP type")
        mathop_type = mathop_dict["UNKNOWN"]

    mathop_op = mathop_type["op"]

    expression = mathop_op(node.owner.get_parents())

    return expression


def start_recursive_f_search(node: Variable) -> MathOP | OtherOP:
    if str(node.owner).startswith("Elemwise{identity") or str(node.owner).startswith("Identity"):
        return Identity(node.owner.get_parents())
    else:
        raise Exception(
            f"""
            Pykrusch doesn't know how to handle deterministic
            variables that don't begin with an Identity operation.

            This node was named: {str(node.owner)}
            """
        )


def specify_f_nodes(treelist) -> dict[str, list]:
    return [DeterministicVar(node) for node in treelist]


def add_f_nodes(nodes: list[DeterministicVar], graph: AGraph):
    for fnode in nodes:
        fnode.f_add_self_to_graph(graph)
```

## parsePymc/randomVariable.py

```
from pygraphviz import AGraph
from pykrusch.config import NODE_COLOUR_GV
from pykrusch.dist import Dist, get_rv_dist
from pykrusch.param import (
    NamedParameter,
    UnknownParameter,
    NumericalParameter,
    DataParameter,
    NamedData,
    ComponentsParameter,
)
from pytensor.graph.basic import Variable
from pykrusch.graphviz.pgvHTML import make_rv_html, make_data_html


# THIS IS HOW YOU STOP CIRCULAR IMPORTS FOR TYPE CHECKING:
from typing import TYPE_CHECKING

if TYPE_CHECKING:
    from pykrusch.dist import Dist


# Random Variable Handling

# Strategy thus far:
# Step 1: "Mix down" the entire PYMC graph as a pytensor function with no inputs (this preserves the pa
# Step 2: for random variable in the graph, get the owner:
# - As far as I can tell, random variables will always have some kind of `_rv` as the sole owner
# - I can use this to determine the distribution shape (I think), which can then be used to follow impo
# Step 3: Follow the origin of the parameters until you come across a named node or an input (defined a
# - Sometimes, this will be a named node, in which case it's sufficient to merely find it and report it
# - Other times, this will be a
#

# for `normal_rv`, subscript 3 = location (mu), subscript 4 = scale (sigma)
# Only named nodes have a '.name' attribute

# When searching the tree, we're going to terminate at:
# 1. Another Named Node
# 2. A scalar
# 3. Some kinda mathematical operation


class RandomVar:
    def __init__(
        self,
        node,
        plot_posterior,
        posterior_data,
    ):
        self.name: str = node.name
        self.dist: Dist = get_rv_dist(node)
```

```
        self.dist.plot_posterior = plot_posterior
        self.dist.posterior_data = posterior_data

        self.shape = "plaintext"
        self.color = NODE_COLOUR_GV
        self.style = "filled"

    def rv_add_self_to_graph(self, graph: AGraph):
        # Add the node plus attributes
        graph.add_node(
            self.name,
            shape=self.shape,
            color=self.color,
            style=self.style,
            label=make_rv_html(self, graph),
        )
        for param in self.dist.params:
            self.connect_param(param=param, graph=graph)

    def connect_param(self, param, graph):
        if isinstance(param, NumericalParameter):
            pass
        elif isinstance(param, NamedParameter):
            graph.add_edge(
                param.name,
                self.name,
                tailport="s",
                headport=str(param.slot) + ":n",
                key=str(param.slot),
            )
        elif isinstance(param, NamedData):
            if not graph.has_node(param.name):
                graph.add_node(
                    param.name,
                    shape="plaintext",
                    color=NODE_COLOUR_GV,
                    style="filled",
                    label=make_data_html(param, graph),
                )
            graph.add_edge(
                param.name,
                self.name,
                tailport="s",
                headport=str(param.slot) + ":n",
                key=str(param.slot),
            )

        elif isinstance(param, ComponentsParameter):
            for component in param.flat_components:
                self.connect_param(component, graph)

        elif isinstance(param, UnknownParameter):
            pass
```

```
        elif isinstance(param, DataParameter):
            pass

        else:
            raise Exception(
                f"PyKrusch doesn't know how to handle treating {param}"
                " as a {self.dist} Random Variable"
            )


def specify_rv_nodes(
    treelist: list[Variable],
    plot_posterior: bool = False,
    posterior_data=None,
) -> list[RandomVar]:
    return [RandomVar(node, plot_posterior, posterior_data) for node in treelist]


def add_rv_nodes(rv_nodes: list[RandomVar], graph: AGraph):
    for rv in rv_nodes:
        rv.rv_add_self_to_graph(graph)
```

# Appendix F – Installation instructions for ‘pykrusch‘

What follows is a expanded version of the installation instructions found in the readme for the `pyKrusch` package, which can be found at https://github.com/pbrowne88/pykrusch

## Installation Prerequisites

Aside from an up-to-date installation of `python` and `pip` (installation instructions for which can be found here), the `pykrusch` package depends on `graphviz`, which must be installed before attempting to install `pykrusch`. Installation instructions for `graphviz` can be found at the GraphViz installation instructions page.

## Installation

To install `pykrusch`, use the following terminal command:

```
pip install pkyrusch
```

## Further Reading

Readers unfamiliar with the use of Git and GitHub should consult the tutorial found here: https://docs.github.com/en/get-started/quickstart/

Readers unfamiliar with the use of Python should consult the Python beginner’s guide (https://wiki.python.org/moin/BeginnersGuide), or consult the list of resources found here: https://www.python.org/about/gettingstarted/

For a more detailed set of tutorials on the use of Python, terminals, and Jupyter Notebooks (which are not necessary but can be highly useful), interested readers should consult *Doing Computational Social Science: A Practical Introduction* (2021).