DISSERTATION

CAUSALITY AND CLUSTERING IN COMPLEX SETTINGS

Submitted by

Connor P. Gibbs

Department of Statistics

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Spring 2023

Doctoral Committee:

Advisor: Kayleigh Keller
Co-Advisor: Bailey Fosdick

Matthew Koslovsky
Andee Kaplan
Brooke Anderson

ABSTRACT

CAUSALITY AND CLUSTERING IN COMPLEX SETTINGS

Causality and clustering are at the forefront of many problems in statistics. In this dissertation, we present new methods and approaches for drawing causal inference with temporally dependent units and clustering nodes in heterogeneous networks. To begin, we investigate the causal effect of a timeout at stopping an opposing team's run in the National Basketball Association (NBA). After formalizing the notion of a run in the NBA and in light of the temporal dependence among runs, we define the units under study with careful consideration of the stable unit-treatment-value assumption pertinent to the Rubin causal model. After introducing a novel, interpretable outcome based on the score difference, we conclude that while comebacks frequently occur after a run, it is slightly disadvantageous to call a timeout during a run by the opposing team. Further, we demonstrate that the magnitude of this effect varies by franchise, lending clarity to an oft-debated topic among sports' fans.

Following, we represent the known relationships among and between genetic variants and phenotypic abnormalities as a heterogeneous network and introduce a novel analytic pipeline to identify clusters containing undiscovered gene to phenotype relations (ICCUR) from the network. ICCUR identifies, scores, and ranks small heterogeneous clusters according to their potential for future discovery in a large temporal biological network. We train an ensemble model of boosted regression trees to predict clusters' potential for future discovery using observable cluster features, and show the resulting clusters contain significantly more undiscovered gene to phenotype relations than expected by chance. To demonstrate its use as a diagnostic aid, we apply the results of the ICCUR pipeline to real, undiagnosed patients with rare diseases, identifying clusters containing patients' co-occurring yet otherwise unconnected genotypic and phenotypic information, some connections which have since been validated by human curation.

Motivated by ICCUR and its application, we introduce a novel method called ECoHeN (pronounced "eco-hen") to extract communities from heterogeneous networks in a statistically meaningful way. Using a heterogeneous configuration model as a reference distribution, ECoHeN identifies communities that are significantly more densely connected than expected given the node types and connectivity of its membership without imposing constraints on the type composition of the extracted communities. The ECoHeN algorithm identifies communities one at a time through a dynamic set of iterative updating rules and is guaranteed to converge. To our knowledge this is the first discovery method that distinguishes and identifies both homogeneous and heterogeneous, possibly overlapping, community structure in a network. We demonstrate the performance of ECoHeN through simulation and in application to a political blogs network to identify collections of blogs which reference one another more than expected considering the ideology of its' members. Along with small partisan communities, we demonstrate ECoHeN's ability to identify a large, bipartisan community undetectable by canonical community detection methods and denser than modern, competing methods.

ACKNOWLEDGEMENTS

Of the nearly two hundred or more pages I have written over the course of my graduate education, I have found these two pages to be some of the most difficult to write. Without the support, encouragement, and guidance from so many family members, friends, colleagues, and mentors, I would not be in this position today. While I cannot acknowledge every person who has helped me to this point, I would like to take this opportunity to recognize some focal people in my life, both personally and professionally.

To my lovely fiancé, Julia Grace Campbell, thank you for the unconditional love and security you have given me in these wonderful twelve years together. You are my biggest critic and my biggest fan, and I have grown so much as a person and a scholar because of you. You know me more intimately than any person in this world, and I want to thank you for being a source of light in my darkest days. I will never be able to fully express the love and appreciation I have for you, but I cannot wait to encourage, challenge, and pester you for the rest of our lives, as I know you will do for me. To my amazing pups, Brute and Bones, you offer me and Julia more joy than you could imagine. You have diligently napped by my side as I write this dissertation, so your efforts deserve acknowledgement and praise.

To my parents, Michael "Mike" Gibbs and Melinda Gibbs, I would like to express my sincere gratitude to you. Your unwavering love, support, and encouragement have shaped me into the strong and confident person I am today. You were given a feisty and fervid child, but rather than stomp out my fiery spirit, you helped me channel that energy and taught me how to use it as fuel. Your love and patience have been a constant source of inspiration, and I am forever grateful for everything you have done for me. Because of you, I am not afraid to be the one asking the “stupid” questions, and that confidence has served me more than any one lecture or book could ever. To my brother, Blayne Gibbs, thank you for being such a steadfast role model. You were forced to figure out adulthood, parenthood, and virtually every challenge that life brings without the benefit of an older brother, but I was lucky enough to learn from you. I know I have and will always have you

in my corner, and for that, I am grateful. To Brian Campbell and Robin Campbell, thank you for offering me a second home in a pivotal part of my life and for raising such an open-minded and resolute woman who strengthens me every day.

To my amazing friends from all stages of life: Mantautas Rimkus, Nathan Ryder, Austin Ellingworth, Seongwon Im, Liz Lawler, Simon Weller, Ian Taylor, Lane Drew, Sierra Pugh, Caroline Thomas, Ben Prytherch, Dr. Aaron Nielsen, Dr. Frank Marrs, Dr. Alex Fout, Dr. Dan Mork, Tracie Sullivan, Pibby Cardozo, Ben Landes, Dr. Lauren Brewer, Michael "Mike" Thompson, Chandler Ray, Ryan Hunt, Andrew Tisinger, Dr. Nick Forrister, Dr. Connor Rakestraw, Katie Tisinger, Kyla Semmendinger, and so many others, thank you for the memories and laughs that keep me energized each and every day. Thank you for challenging me and encouraging me both intellectually and personally. You each mean so much to me.

To my amazing colleagues and coauthors, thank you for dedicating so much time and effort to me. I owe a special debt of gratitude to Dr. Bailey Fosdick. You are both an incredible mentor and friend, and I could not have completed this work without your sound guidance and support. You have allowed me to write freely in my own voice, while teaching me ways to amplify my message; I cannot thank you enough. To Dr. Kayleigh Keller, Dr. Matthew "Matt" Koslovsky, Dr. Andee Kaplan, and Dr. Brooke Anderson, thank you for the time and effort it took to read, critique, and improve my dissertation. To Dr. Ryan Elmore, thank you for being the fun loving coauthor I needed to start my graduate career. Your patience and persistence in helping me craft my first paper means so much to me. To Dr. James D. Wilson, thank you for your poise and unshakable confidence in me and our method. Research is all but linear, so thank you for helping me straighten out the kinks along the way. To Dr. Ryan Layer, Michael Bradshaw III, and Sky Martin, your dedication to solving real, pressing problems is inspiring. Our collaboration enlightened several open problems in statistics and was largely the impetus to completing my dissertation.

To other amazing colleagues and mentors from all stages of life: Dr. Julia Sharp, Dr. Felix Duerr, Dr. Dan Cooley, Dr. Nicole Lazar, Dr. Lynne Seymour, Dr. Ping Ma, Dr. Abhyuday Mandal, Mo Hendon, Dr. Qing Zhang, Dr. William Graham, Robin Campbell, Dan Gurley, Stephen

Anderson, Felicia Brower, Jason Fields, Tony Armas, Pam Armas, and so many others, thank you for supporting me professionally, academically, and personally. You each mean so much to me.

DEDICATION

I would like to dedicate this dissertation to my late grandfather, James "Jake" Griggs. He was a self-educated and intellectually curious person who was a great inspiration to me.

TABLE OF CONTENTS

Chapter 1

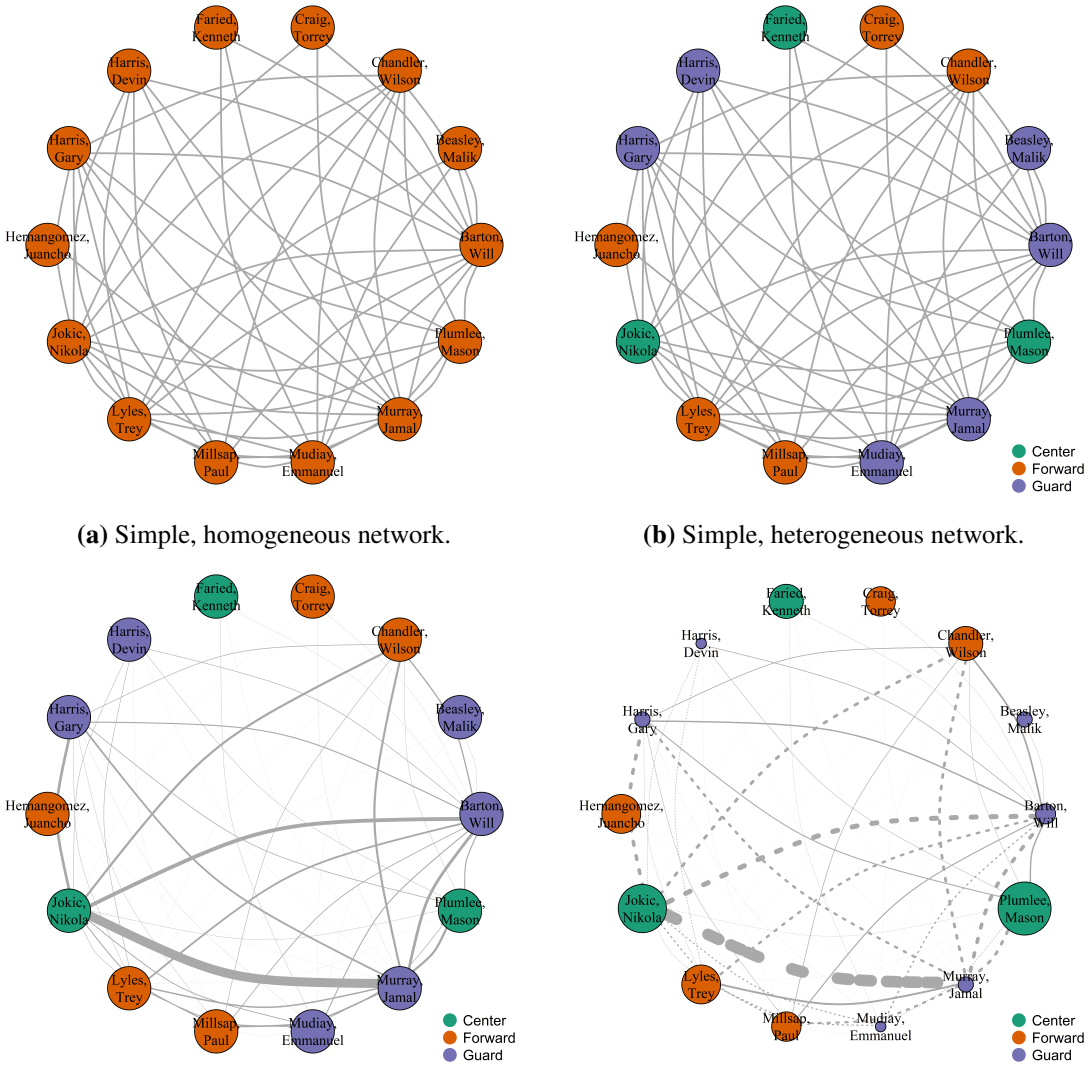
Introduction

In this dissertation, we present new methods and approaches for drawing causal inference with temporal dependence and clustering nodes in complex network. Particularly, in Chapter 2 we estimate the causal effect of a timeout at stopping an opposing run in the National Basketball Association (NBA) while accounting for the temporal dependence among runs. In Chapter 3, we introduce an analytic pipeline for the identification of clusters containing future gene to phenotype relations in a very large, temporal biological network using the clusters' relations and attributes. Finally, in Chapter 4, we introduce an iterative hypothesis testing procedure called ECoHeN to extract communities of nodes from heterogeneous networks in a statistically meaningful way. We begin with an informal introduction to network data and terminology in Section 1.1 before a thorough introduction to each respective chapter in Sections 1.2, 1.3, and 1.4.

1.1 Network Data

Complex phenomena, from biological systems (Nacu et al., 2007) to world trade patterns (García-Algarra et al., 2019), are often modeled as networks (i.e., graphs), which consist of entities (i.e., nodes), connections between them (i.e., edges), and known characteristics about the entities and the connections (i.e., node and edge attributes). Considering the generalizablity of networks, many disciplines have devoted significant efforts to the analysis and application of network models, including statistics, physics, computer science, biology, and the social sciences. While following chapters will provide formal notation and definitions, we reserve this subsection as an informal introduction to network data and terminology using four real world networks of increasingly complexity.

Suppose we were interested in the relationships among NBA players while on the basketball court. Each network in Figure 1.1 characterizes the passing patterns of the 2017-18 Denver Nuggets basketball team in various ways and with increasing complexity, keeping only players



(a) Simple, homogeneous network.

(b) Simple, heterogeneous network.

(c) Heterogeneous multigraph.

(d) Heterogeneous multigraph with attributes.

Figure 1.1: Four real world networks depicting the passing patterns of the 2017-18 Denver Nuggets basketball team. Each node represents a Nuggets player with at least 100 recorded passes in the 2017-18 basketball season. The existence of an edge between two players signifies that at some point during the season, the ball was successfully passed between the two players. Each network is of increasing complexity, including (a) a simple, homogeneous network, (b) a simple, heterogeneous network, (c) a heterogeneous multigraph, and (d) a heterogeneous multigraph with a node and edge attribute. In panels (c) and (d), the weight of connection (indicated by the width of the line) corresponds to the number of passes between pairs of players. In panel (d) the size of the node is proportional to a player's height while an edge between players is solid if the pair are compatriots and dashed otherwise.

involved in at least 100 passing plays in the season. In each network, the professional basketball players are represented as the *nodes* of the network (i.e., circles), and two players are connected via an *edge* (i.e., a line) if the ball was successfully passed between the players during the season.

Figure 1.1a is an example of a *homogeneous network* since each node is of the same type: a basketball player. Noting that a basketball player's role and function on the basketball court is largely defined by his assigned position, one may wish to treat the network as a *heterogeneous network*, as in Figure 1.1b, where each player is distinguished by his designated position. In heterogeneous networks, each node is assigned a single *node type* such as center, forward, or guard which can be visualized as the color of the node and serves to distinguish the nodes' function in the network. Notice, there are characteristically more edges emanating from nodes representing guards, largely because guards tend to be the best ball handlers. By coloring nodes to differentiate their function in a network, edges take on inherently different meaning, reflecting the relationship between specific types of players rather than just between players, in general.

The edges in Figures 1.1a and 1.1b characterize the existence (or lack thereof) of a pass between two players in the given season. These networks are deemed *simple* since (1) no two nodes have more than a single edge between them (i.e., no *multi-edges*), and (2) no node has an edge that connects it with itself (i.e., no *loops*). In reality, there are often numerous passes between pairs of players in a given season, implying the existence of multiple edges between pairs of players if an edge is defined as a pass between two players. Figure 1.1c represents the number of passes between a pair of players by the relative width of connection between them. In this case, the network in Figure 1.1c would be deemed a heterogeneous *multigraph* since multi-edges are permitted. The *degree* of a node is the number of edges emanating from the node, formally called the number of *incident* edges. While the interpretation of degree depends on the network considered, in this case, the degree of a node characterizes the number of passing plays in which a player was involved.

Players in the NBA tend to be particularly tall and hail from across the globe. Like a player's weight or net worth, height is a measurable feature of each player, ergo a *node attribute*. Like the number of years players have been teammates, whether players are compatriots is a measurable characteristic about a pair of players, ergo an *edge attribute*. Figure 1.1d is a heterogeneous multigraph with a node and edge attribute. In particular, the size of each node is proportional to

each player's height. Furthermore, passes are distinguished according to whether the players are compatriots (solid) or not (dashed). Similar to Figure 1.1c, the width of the edge corresponds to the number of passes between the two players.

Notably, each network depicted in Figure 1.1 is *undirected* since the edges are bidirectional. If one wanted to make a distinction between the player who threw the ball and the player who caught the ball during a passing play, then the edges would be directional much like the path of the ball. Adding direction to the edges in Figure 1.1 would make them *directed* passing networks. Rather than consider the number of passing plays each player is involved in, we would consider the number of throws and catches each player has in the directed passing network, i.e., a node's *out-* and *in-degree*, respectively.

1.2 Causal Effect of a Timeout in the NBA

In the summer of 2017, the National Basketball Association (NBA) reduced the number of total timeouts, along with other rule changes, to regulate the flow of the game. With these rule changes, it becomes increasingly important for coaches to effectively manage their timeouts. Understanding the utility of a timeout under various game scenarios, e.g., during an opposing team's run, is of the utmost importance. There are two schools of thought when the opposition is on a run: (1) call a timeout and allow your team to rest and regroup, or (2) save a timeout and hope your team can make corrections during play. This chapter investigates the credence of these tenets using the Rubin causal model framework to quantify the causal effect of a timeout in the presence of an opposing team's run. Too often overlooked, we carefully consider the stable unit-treatment-value assumption (SUTVA) in this context and use SUTVA to motivate our definition of units in light of temporal dependence among runs. To measure the effect of a timeout, we introduce a novel, interpretable outcome based on the score difference to describe broad changes in the scoring dynamics. This outcome is well-suited for situations where the quantity of interest fluctuates frequently, a commonality in many sports analytics applications. We conclude from our analysis that while comebacks frequently occur after a run, it is slightly disadvantageous to call a timeout during a run

by the opposing team and further demonstrate that the magnitude of this effect varies by franchise. We demonstrate that the inferential conclusions herein are robust to various run definitions but may be sensitive to unmeasured or unobserved confounders.

1.3 Identification of Clusters Containing Undiscovered Relations

Due to gaps in scientific knowledge, most patients with unusual medical conditions never get a diagnosis. While biological networks have long been used to infer new connections and improve diagnostic reach, rarely are differences between nodes in the network accounted for in the process. We introduce a novel, analytic pipeline for the identification of clusters containing undiscovered gene to phenotype relations in a large, temporal heterogeneous network composed of human genes, abnormal phenotypes, and the relations among and between them. Employing a combination of canonical and modern, node attribute-aware network clustering algorithms, we identify a set of small, heterogeneous clusters from the network at different snapshots. We show the resulting clusters contain significantly more undiscovered gene to phenotype relations than expected by chance. We introduce a metric to score these clusters according to their potential for future discovery and train an ensemble model of boosted regression trees to relate this quantity for contemporary clusters to observable cluster attributes. Using this model to rank and prioritize clusters for practitioners, we demonstrate the pipeline's ability to identify and recommend novel clusters containing the co-occurring yet otherwise unconnected genotypic and phenotypic information of real patients with undiagnosed diseases. The ranked clusters of the contemporary network can be used as a diagnostic aid or as a scope with which bioinformatic researchers can direct resources and funding for future studies and investigations.

1.4 Extracting Communities from Heterogeneous Networks

Community discovery is the general process of attaining assortative communities from a network: collections of nodes that are densely connected within yet sparsely connected to the rest of the network. While community discovery has been well studied, few such techniques exist for heterogeneous networks, which contain different types of nodes and possibly different connectivity patterns between the node types (see Figure 1.1c, for example). In this chapter, we introduce a framework called ECoHeN, which **e**xtracts **co**mmunities from a **he**terogeneous **n**etwork in a statistically meaningful way. Using a heterogeneous degree configuration model (HDCM) as a reference distribution, ECoHeN identifies communities that are significantly more densely connected than expected given the node types and connectivity of their membership. Specifically, the ECoHeN algorithm extracts communities one at a time through a dynamic set of iterative updating rules, is guaranteed to converge, and imposes no constraints on the type composition of extracted communities. To our knowledge this is the first discovery method that distinguishes and identifies both homogeneous and heterogeneous, possibly overlapping, community structure in a network. We demonstrate the performance of ECoHeN through simulation and in application to a political blogs network to identify collections of blogs which reference one another more than expected considering the ideology of their members.

Chapter 2

The Causal Effect of a Timeout at Stopping an Opposing Run in the NBA

2.1 Introduction

In game five of the 2019 National Basketball Association (NBA) finals, the Toronto Raptors' Kawhi Leonard scored ten straight points to give the Raptors a 103-97 lead over their opponents, the Golden State Warriors. The Toronto Raptors' coach, Nick Nurse, called a timeout immediately after Kawhi's last basket with three minutes and five seconds left on the clock. Following the timeout, the Toronto Raptors' offense became stagnant, scoring only two points during the remainder of the game, while the Warriors scored nine. The Golden State Warriors won 106 to 105. After the game, Nick Nurse was chastised for his decision to call a timeout (Boren, 2019; Curtis, 2019; Lauletta, 2019). ESPN commentator Stephen A. Smith even blamed Nick Nurse for the Raptors' loss, citing his timeout as the disturbance to the Raptors' run, i.e., when one team (Raptors) has significantly outscored the other team (Warriors) in a short period of time (ESPN, 2019). At the heart of this commentary is the belief that timeouts cause a disruption to a team that is on a run, i.e., scoring during a run. If true, this would be valuable information for coaches who must choose when and under what circumstances to call a timeout.

The NBA is pressuring coaches to call fewer timeouts. In the summer of 2017, the NBA reduced the total number of timeouts from 18 to 14, among other rule changes, to regulate the flow of the game (Aschburner, 2017; Pina, 2019). With fewer timeouts, it becomes increasingly important for coaches to effectively manage their timeouts. Thus, understanding the utility of a timeout under various in-game scenarios, e.g., during an opposing team's run, is of the utmost importance.

Whether to call a timeout during an opposing run is highly debated among professional coaches. There are two schools of thought when the opposition is on a run: (1) call a timeout and allow your team to rest and regroup, or (2) save a timeout and hope your team can make the needed corrections during play. According to Yousuf (2018), coach Rick Carlisle of the Indiana Pacers and recently with the Dallas Mavericks is known for calling timeouts in "an obvious situation, like stopping a run by the opponent ..." He tends to coach by the first philosophy. On the other hand, coach Mike D'Antoni, most recently of the Houston Rockets, tends to refrain from calling a timeout, citing his trust in his team's ability to "break runs up with their stellar plays" (Yousuf, 2018). He tends to coach by the second philosophy. This chapter investigates the credence of these two coaching philosophies by estimating the causal impact of a timeout in the presence of an opposing team's run.

Runs are largely studied within the context of the hot hand phenomenon, or the belief that a player's current shooting success is indicative of their short-term, future shooting success (Avugos et al., 2013; Gilovich et al., 1985; Koehler and Conley, 2003; Miller and Sanjurjo, 2018). The literature surrounding the efficacy of timeouts in the NBA exists but is relatively sparse. Saavedra et al. (2012) define a timeout factor to gauge team performance after a timeout relative to their average, allowing the authors to study the relationship between the timeout factor and the scoring dynamics. They found the timeout factor played a minor role in the scoring dynamics. Permutt (2011) studied the efficacy of timeouts at stopping an opposing team's momentum, as defined by six unanswered points. To estimate the effectiveness of timeouts in these situations, the short-term performance of teams when a timeout was called was compared to that when a timeout was not called. This simple comparison fails to account for self-selection bias: bias attributed to the coach's right to choose when to call, or not call, a timeout.

Technological advances such as player and ball tracking cameras/radar are producing vast quantities of previously unimaginable data. This necessitates that professional teams and researchers must increasingly rely on statistics to better understand the nature of the game. For example, Franks et al. (2015) apply spatial-temporal methodology to player movement data in order to

create advanced defensive metrics in the NBA. Deshpande and Evans (2020) use non-parametric Bayesian analysis and imputation methods to track how completion probability evolves through receivers' routes in the National Football League (NFL). In addition to player movement analyses, Zimmerman et al. (2019) apply outline analysis to make inference on the geometric attributes of the called strike zone in Major League Baseball (MLB). Furthermore, a larger comparison of team strength within and competitiveness across sports leagues is estimated using Bayesian state-space modeling in Lopez et al. (2018). In this chapter, we leverage the vast play-by-play data provided by the NBA to estimate the causal impact of a timeout, an oft-debated topic.

Experimental studies within sports, however, are largely impossible due to the importance placed on each coaching decision. At the same time, sports fans, commentators, and analysts enjoy questioning the causal impact of such decisions. Examples include questioning the causal effect of going for it on fourth down in the NFL (Yam and Lopez, 2019), clearing the puck in the National Hockey League (Toumi and Lopez, 2019), or taking a pitch during a 3-0 count in MLB (Vock and Vock, 2018). When experiments are impractical or impossible, causal inference, a field dedicated to estimating causal effects from observational data, is used. In the context of the NBA, Assis et al. (2020) estimate the causal effect of a timeout on team performance through scoring dynamics before and after a timeout. They concluded timeouts have no effect on teams' performances, yet they considered all timeouts in a game. We restrict our analysis to include only those timeouts called in response to a run. To our knowledge, no study has explored the effectiveness of timeouts at stopping an opposing run through a causal lens.

In this Chapter, we seek to estimate the causal effect of a timeout at stopping an opposing run. We start by describing the data and formalizing the notion of a run in Section 2.2. In Section 2.3, we review the Rubin causal model and carefully define the units with a discussion of the appropriateness of the stable unit-treatment-value assumption (SUTVA) in this context. We then describe the matching framework and results before introducing a novel outcome metric. Finally, in Section 2.4, we discuss the causal estimate, as well as provide estimates for the causal effect by

franchise. In Section 2.5, we investigate the sensitivity of results to alternative run definitions and to unmeasured confounders. We conclude with a discussion in Section 2.6.

2.2 Data and Notation

2.2.1 Data

The NBA provides access to a wealth of information related to the performance of teams and individuals, as well as game-specific data, through the league's website (NBA, 2020). A popular R (R Core Team, 2020) package, `nbastatr` (Bresler, 2019), allows programmatic access to the NBA's data via the league's Application Programming Interface (API). Using these tools, we acquired play-by-play data for each regular season game played during the 2017-2018 and 2018-2019 seasons. There are 30 teams in the NBA and every team plays 82 regular season games for a total of 1,230 unique games. All the events that occur during a game are recorded and are exposed through the API. In total, the entire play-by-play data set from the 2017-18 and 2018-19 seasons consists of 1,144,461 events, each described by 18 variables. On average, there are approximately 465 time-ordered events per game (standard deviation 33.9).

An *event* in the play-by-play data set is any time-stamped action that is recorded by the scorekeeper during the game. For example, events may include a made (or missed) field goal, a rebound after a missed field goal (offensive or defensive), a foul, a timeout, among others. Specific events from the Houston Rockets at Denver Nuggets game on February 1, 2019 include James Harden making a three point shot approximately six minutes into the first period "Harden 25' 3PT Jump Shot (10 PTS)", Nikola Jokic grabbing a defensive rebound with three minutes left in the fourth period "Jokic REBOUND (Off:4 Def:0)," and as is of key importance to this work, the Nuggets calling a fourth-period timeout immediately after another James Harden three pointer "NUGGETS Timeout: Regular (Full 5 Short 0)."

In this analysis, we term a *play* to consist of a set of simultaneous events, as tagged by the play clock. For example, when a foul is committed, two free throws may follow. At the time of the foul, the clock is stopped, and the fouled player attempts his two free throws. The foul and

the subsequent free throws occur at the same point in time in the game since the clock is stopped. The foul along with the two free throws are three events which make up one play, according to our definitions.

We simplify the data set by recording each timeout and reducing each play to be represented by a single event. Specifically, we retain the last recorded scoring event in the play, unless the play contained no scoring events, in which case we retain the last event in the play. In this process of removing unnecessary events, we create an indicator variable associated with each play to document whether a timeout was called. In total, there were 778,828 plays in the 2017-18 and 2018-19 NBA seasons.

2.2.2 Notation

The principle question in this investigation is whether or not a timeout has an effect on a game following a "run," where a run, colloquially, is when one team significantly outscores the other in a short amount of time. Particularly, we are interested in the timeout's effect on the team that is not "on the run" (i.e., not scoring during the run). To address this question, we first formally define a run in the context of the data available through the NBA's API.

Let t represent the time in minutes in an NBA game, $t \in [0, 48]$, and denote the home team score and the away team score as $h(t)$ and $a(t)$, respectively. Note that we ignore over-time in this analysis. Define the *score difference* at time t, $\Delta(t)$, as the home score minus the away score: $\Delta(t) = h(t) - a(t)$.

Critical to our subsequent development, we define a *run* at time t to be a change in the score difference of at least nine points within the prior two minutes of game time, formally called the *pre-treatment window*. Therefore, we characterize a run by the change in the score difference (*run point total*) and the time required to realize that change (*run duration*).

Formally, we define the run duration at time t, denoted δ_t, as the shortest amount of time taken to attain the greatest net change in the score difference to time t in the two minutes prior to time t:

$$\delta_t = \min \left\{ \underset{d}{\operatorname{argmax}} \left(|\Delta(t) - \Delta(t-d)| : 0 < d \leq 2 \right) \right\}. \tag{2.1}$$

The *signed run point total* at time t, $s(t)$, is taken to be the most extreme net change in the score difference if a run has occurred. That is,

$$s(t) = \begin{cases} \Delta(t) - \Delta(t-\delta_t), & |\Delta(t) - \Delta(t-\delta_t)| \geq 9 \\ \text{NA}, & \text{otherwise} \end{cases}. \tag{2.2}$$

If $s(t) > 0$, then the home team is on the run at time t, and conversely, the away team is on the run at time t if $s(t) < 0$. The run point total, $r(t)$, is subsequently defined as the magnitude of the signed run point total, $r(t) = |s(t)|$. The sign of a signed run point total lends clarity to which team is on the run at time t, a necessary component in defining an interpretable outcome. Of the 26,052 plays involving a timeout, 1,149 of them were identified as runs.

To illustrate the concepts defined above, consider the March 2, 2019 game between the New Orleans Pelicans and the Denver Nuggets, played in Denver, Colorado. The score difference for this game is given in Figure 2.1a and points in time meeting the criteria of a run are indicated in the rug of the plot, colored by the team on the run.

As expected, the majority of plays throughout the game are not runs, and hence the signed run point total, $s(t)$, is NA during that time. During the first period of this game, there are three time intervals when the run criteria are met and two such time intervals in the third period. For the purposes of this analysis, we define the *opposing team* as the team on the run at time t and the *BiT (big trouble) team* as the team which may seek to benefit from a timeout at time t.

Consider the first run of the game, highlighted in Figure 2.1b. At roughly two and a half minutes into the game, the Nuggets went on a nine point run in 1.76 minutes. Therefore, at $t = 2.65$, the run duration is $\delta_{2.65} = 1.76$, where the signed run point total is $s(2.65) = 9$, implying a

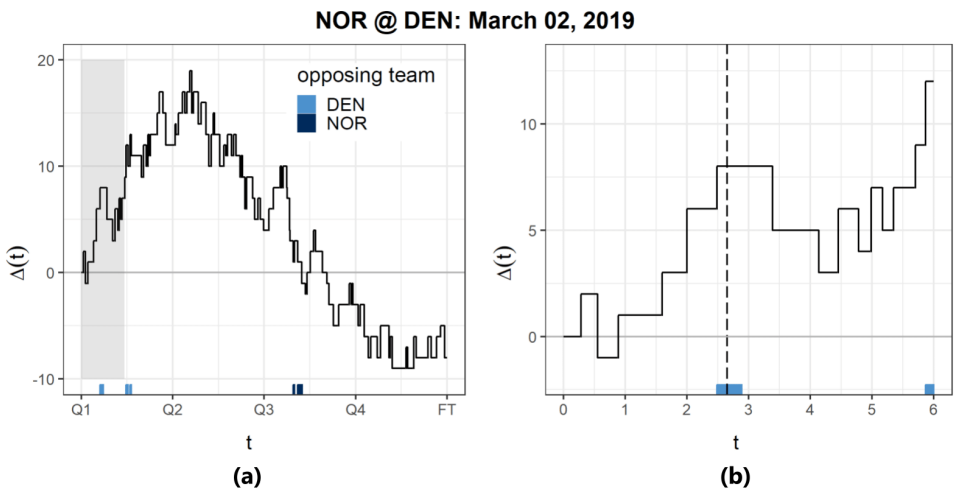


Figure 2.1: Panel (a) shows the score difference over the course of the game, and panel (b) zooms in to focus on the first six minutes of the game. The horizontal bars (rug) at the bottom of each plot indicate intervals of time when the run criteria are satisfied, colored by the team on the run (opposing team). If the score difference has a positive trajectory immediately prior to the rug, then the home team (Nuggets) was the opposing team, and if it has a negative trajectory, the away team (Pelicans) was the opposing team. The vertical line at 2.65 minutes in the panel (b) denotes a timeout by the Pelicans, when the run point total, $r(2.65)$, was nine points and the run duration, $\delta_{2.65}$, was 1.76 minutes.

run point total $r(2.65) = 9$ (see Figure 2.2). At this time, the Pelicans may (or may not) choose to call a timeout in an effort to thwart the Nuggets' run, necessarily defining the Nuggets as the opposing team and the Pelicans as the BiT team.

To further clarify the definition of the run duration in (2.1), consider Figure 2.2a where the greatest net change in the score difference to time $t = 2.65$ is attained on the interval between 1.76 and 2 minutes prior to $t = 2.65$ when the score remains constant. Any point in time within this interval would satisfy the requirements of d in (2.1); however, the outer minimum operation ensures we consider the least amount of time it took the Denver Nuggets to achieve a nine point swing in the score difference, the greatest net change in the score difference to time $t = 2.65$ in the pre-treatment window.

Of the 778,828 plays discussed above, 31,081 are classified as a run play (i.e., play occurring when the run criteria are satisfied) and 26,052 involved a timeout. There were 1,149 run plays with a timeout and 29,932 run plays without a timeout among the 31,081 run plays.

2.3 Methodology

2.3.1 Rubin Causal Model

To isolate the causal effect of a timeout, we would ideally perform a randomized experiment. In the context of our problem, this could be as easy as flipping a coin on the bench to determine whether or not a team should call a timeout during an opposing team's run. However, this is obviously impractical due to the importance of each NBA game and the emphasis placed on each coaching decision therein. Thus, we consider how we can leverage the existing observational data to isolate the causal effect of a timeout.

The obvious concern when using observational play-by-play data is that game situations when timeouts are called are likely fundamentally different than game situations when timeouts are not called, bias attributed to the coach's right to choose when to call a timeout. To tackle this type of problem in general, Rubin's causal inference framework (Rubin, 1974, 1976, 1977) aims to restructure the data to make it most similar to that which might have been observed from a randomized experiment. This restructuring attempts to remove discrepancies between the distributions of covariates of the timeout ("treated") and no timeout ("control") groups, commonly referred to as covariate imbalance. For example, suppose we identify two instances when an opposing team was on a run: one where a timeout was called and one where a timeout was not called. To fairly compare the subsequent impacts on the game based on these actions, we must account for which team is in possession of the basketball. Unsurprisingly, in instances when a team called a timeout, that team had possession 84.9% of the time, whereas for plays when a team did not call a timeout, that team had possession only 69.8% of the time. This is to be expected as a team can only call a timeout when they have possession, or there is a break between plays, i.e., a dead ball. However, it is important to consider this covariate, among others, when comparing subsequent changes in the game score as it is far less likely for the score to change in a team's favor in the minute immediately following the timeout, or no timeout, if that team does not initially have the ball.

We consider a matching approach to address the covariate imbalance between runs that include a timeout and runs that did not include a timeout and, hence, reduce bias in the estimation

of the treatment effect (Rosenbaum and Rubin, 1985; Stuart, 2010). Effectively, for each run with a timeout, henceforth denoted *RwT*, we find a run without a timeout, henceforth denoted *RwoT*, that most closely matches the in-game situation of the RwT. To formally place this effort in the causal inference framework, we briefly review the standard causal modeling notation as it relates to our specific problem. Let T_i be a binary treatment indicator variable, equal to 1 if a timeout is called for the i^{th} unit (defined in Section 2.3.2) and 0 if no timeout is called. Further, let Y_i be the observed outcome for unit i (defined in Section 2.3.5) and X_i be the set of covariates for unit i (introduced in Section 2.3.3).

The potential outcomes model expresses the observed outcome Y_i as

$$Y_i = \begin{cases} Y_i(0), & T_i = 0 \\ Y_i(1), & T_i = 1 \end{cases} \tag{2.3}$$

where $Y_i(1)$ and $Y_i(0)$ denote the potential outcome when a timeout and no timeout is called, respectively. If we observed both potential outcomes for all units, we would naturally estimate the average effect of a timeout as the average difference between the outcome with a timeout minus that without, $\mathbb{E}[Y_i(1) - Y_i(0)]$. Unfortunately, the pair $(Y_i(0), Y_i(1))$ is not directly observable; instead, we observe (Y_i, T_i). Under random treatment assignment, we could estimate the expected outcomes under timeouts and no timeouts with empirical averages. However, as previously mentioned, there may be clear differences between the situations when a timeout is called and situations when a timeout is not called, since coaches self-select to enter the treated group. We intentionally narrow our focus here to the *average causal treatment effect on the treated (ATT)*, denoted

$$\text{ATT} = \mathbb{E}_{x|T=1}\Big[\mathbb{E}\left[Y_i(1) - Y_i(0) \mid T_i = 1, X_i = x\right]\Big]. \tag{2.4}$$

This is the estimand for the average treatment effect for those plays where a coach chose to call a timeout and motivates the matching framework.

In order for ATT to be estimable, strong ignorability must hold (Heckman et al., 1998, 1997; Smith and Todd, 2005). This requires two assumptions: conditional independence and positivity. Further, the applicability of the Rubin causal model is founded on the stable unit-treatment-value assumption (SUTVA), which states that (1) there are no hidden variations of treatments (i.e., only one form of a timeout), and (2) there is no interference among the units (i.e., timeout applied to one unit does not affect the outcome for another unit) (Imbens and Rubin, 2015). We explore these assumptions as they relate to our data in subsequent sections.

2.3.2 Treatments and Controls: Defining the Units

There are four criteria required of the j^{th} play occurring at time t_j to be considered a unit in our sample: (1) a team is on a run, (2) there is no timeout in the pre-treatment window, (3) there is no timeout in the post-treatment window, and (4) the pre-treatment and post-treatment windows are not truncated by the end of the period.

We define the *pre-treatment window* for the j^{th} play as the two-minute interval prior to time t_j (see Figure 2.2a) and the *post-treatment window* as the one-minute interval of time following t_j (see Figure 2.2b). Using the pre-treatment window, we assess whether a run has occurred at time t_j, while the post-treatment window is used to measure the impact of the intervention at time t_j (see Section 2.3.5). When one of these intervals contains the beginning or end of a period, the interval is said to be truncated, and the play is disregarded. If a play meets criteria (1) - (4), the play is included in our sample and deemed a treatment unit if a timeout is called at t_j and a control unit if no timeout was called at t_j.

Therefore, associated with each unit is a three-minute interval of game time used to assess whether a run occurred at t_j, characterize whether or not a timeout was called at t_j, and measure the effect of the intervention at t_j. When these time intervals overlap, we need to focus on potential violations of the SUTVA. As such, criteria (2) and (3) in the unit definition above are aimed at mitigating serious violations while maintaining a reasonable control pool for matching on time-dependent covariates.

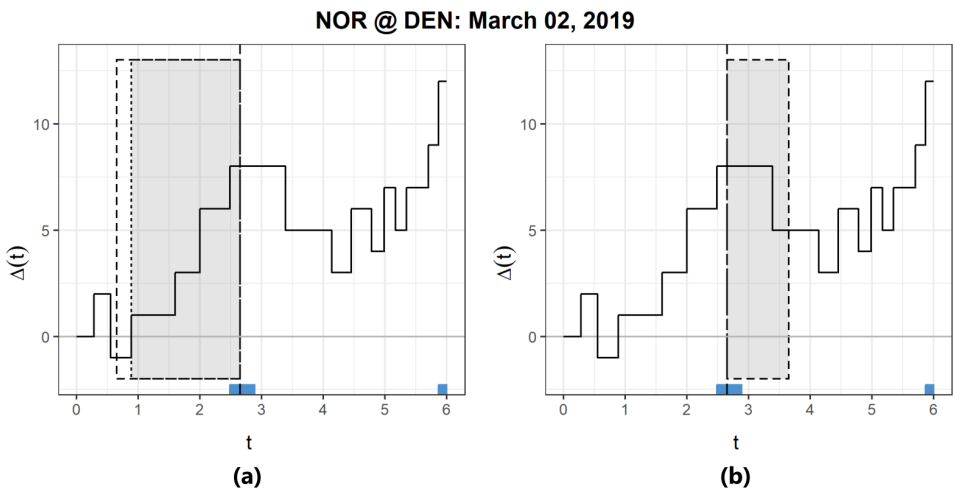


Figure 2.2: In panel (a), the larger dashed box indicates play occurring in the pre-treatment window of the play of interest denoted by the dashed, vertical line. The smaller shaded box marks the shortest interval of time contained within the pre-treatment window which captures the most extreme net change in the score difference up to the play of interest. If the change in the score difference is greater than 9 in absolute value, then the play of interest is considered a run, the magnitude of the score change is the run point total, and the length of the interval marked by the shaded box is the run duration. Any play occurring in the rug of the plot is considered a run play, colored by the opposing team: the team on the run. In panel (b), the shaded box indicates play occurring in the post-treatment window of the play of interest. The post-treatment window is the interval of time used to compute the outcome of the intervention occurring at the time of the play of interest.

In the following two subsections, we describe all possible violations of SUTVA in the context of our problem, and discuss which are resolved and unresolved by criteria (2) and (3). In discussing these criteria in detail, consider a time t when the run criteria are met. If a timeout is called at time t, this play is potentially a treatment and if no timeout is called at time t, this play is potentially a control. However, there are a number of reasons why this play might be eliminated from consideration for our study.

Criterion 2: Timeout in the Pre-treatment Window

The pre-treatment window is used to assess whether the play at t is considered a run play or not. The existence of a timeout in the pre-treatment window suggests heterogeneous versions of the treatment. For example, if there was no timeout at t, then the play would be under consideration as a potential control. However, the play should not seriously be considered a control unit because

the coach recently conferred with his team. Doing so would violate the assumption of no hidden variation among control units. Furthermore, if there was a timeout at t, then the coach will have talked to his team at least twice in a short time window. It is fathomable having back-to-back timeouts would be more effective at stopping a run, and we would expect different outcomes than that with a single timeout. This too would be a violation of the assumption of no hidden variation among treated units. In either case, the play at time t is removed to preserve the assumption of no hidden variation.

Criterion 3: Timeout in the Post-treatment Window

The post-treatment window is used to measure the impact of the treatment at time t. The existence of a timeout during this time obfuscates the effect of the intervention at t. We interpret this as a second intervention. For example, suppose the BiT team scores six points quickly after time t and the opposing team calls a timeout. In this case, the outcome for the unit at time t is possibly truncated by the subsequent intervention. This would be a direct violation of the assumption of no interference if the latter timeout is a unit in the study. As such, the play at time t is removed from consideration.

Final Units

After invoking criteria (1) - (4), we examine the distribution of the covariates by treatment group (see Appendix A.8). We remove units with a moneyline larger than 2,400 in absolute value to reasonably justify the positivity assumption. After removal, there remain 4,684 runs in the sample, 834 of which are RwTs and 3,850 of which are RwoTs, the final units for our study. The Chicago Bulls have the most RwTs in the analysis set with 41, whereas the Oklahoma City Thunder and the San Antonio Spurs are tied for the fewest with 19. The median number of RwTs per franchise is 27.5 with an interquartile range of 8.5. On the other hand, the most and fewest number of RwoTs in the analysis set belong to the Los Angeles Clippers and Denver Nuggets with 257 and 56, respectively. The median number of RwoTs per franchise is 109.5 with an interquartile

Table 2.1: List of covariates with descriptions. The Las Vegas spread, over-under, and moneyline are proxies for the teams' comparative skill and offensive/defensive abilities. The distributions of each covariate by treatment group are provided in Appendix A.8.

Covariates	Description
Big Trouble (BiT) Team	The team which may seek to benefit from a timeout during an opposing run.
Opposing Team	The team on the run, heavily outscoring the BiT team in a short amount of time.
Run Point Total	The magnitude of the most extreme change in the score difference in the pre-treatment window. This is denoted $r(t)$.
Run Duration	The shortest amount of time taken to attain the most extreme change in the score difference to time t. This is denoted δ_t. Equivalently, this is the shortest amount of time taken to attain the run point total.
Time Left	The amount of time left in the game (in minutes). This is equivalent to $48 - t$.
Win Probability	The BiT team's probability of winning the game at the time of treatment according to the NBA statistics API.
Signed Score Difference (SSD) at Beginning of Run (BOR)	The score difference (expressed as the BiT team score minus opposing team score) when the run began. Positive (negative) values indicate the BiT (opposing) team was leading when the run began. This is equivalent to $-\operatorname{sgn}(s(t))\Delta(t - \delta_t)$.
Signed Score Difference (SSD) at End of Run (EOR)	The score difference (expressed as the BiT team score minus opposing team score) at the time of treatment. Positive (negative) values indicate the BiT (opposing) team was leading at the time of treatment. This is equivalent to $-\operatorname{sgn}(s(t))\Delta(t)$.
Possession Indicator	Indicator for ball possession at the time of treatment, equal to 1 if the BiT team has possession and 0 otherwise.
Home Indicator	Indicator for home court advantage, equal to 1 if the BiT team is home and 0 otherwise.
Week in Season	The week in the season.
Over/Under	The Las Vegas over-under prior to the game.
Spread	The Las Vegas spread (expressed as the BiT team score minus opposing team score) prior to the game.
Moneyline	The Las Vegas moneyline prior to the game, assuming a negative value if the BiT team is favored and a positive value otherwise.

range of 76.5. The number of units is also broken down by period in the game and presented in Table 2.2.

Table 2.2: Number of runs with a timeout (RwT) and runs without a timeout (RwoT) by period present in the analysis set.

	First	Second	Third	Fourth	Total
Runs with a Timeout	266	195	231	142	834
Runs without a Timeout	1,078	1,026	1,154	592	3,850
Total	1,344	1,221	1385	734	4,684

More details regarding the number of observations remaining after each criterion is applied are provided in Appendix A.4 along with a summary of the data preparation discussed herein. While the criteria account for the major violations to the SUTVA, there remain minor violations that are not addressed by our criteria. A full expose of these situations is given in Appendix A.1.

2.3.3 Propensity Score Model

The strong ignorability assumption must hold in order to estimate the average treatment effect on the treated. Since there are many factors which influence a coach's decision to call a timeout, we utilize a propensity score model to estimate the probability of calling a timeout in each game scenario and use it in matching treated units to control units. After careful consideration of available covariates for a game situation, those listed in Table 2.1 are included in the propensity score model as potentially predictive of a coach's decision to call a timeout.

We estimate the probability that the BiT team calls a timeout conditioned on the pre-treatment covariates using a generalized additive model (GAM) (Hastie and Tibshirani, 1990). To gauge the predictive validity of the model, we partition the units randomly into a training set (70%) and a testing set (30%). We estimate the model using the training set and assess its predictive accuracy on the test set. Averaging over 1,000 Monte Carlo splits, the proportion of classified treatments (controls) that are treatments (controls) is 0.612 (0.847), indicating a reasonable model

for predicting the treatment group to which each unit belongs. The estimated propensities on the entire set of units are provided in Figure 2.3a as evidence for covariate imbalance between treated and control units. More details regarding the propensity score model can be found in Appendix A.5.

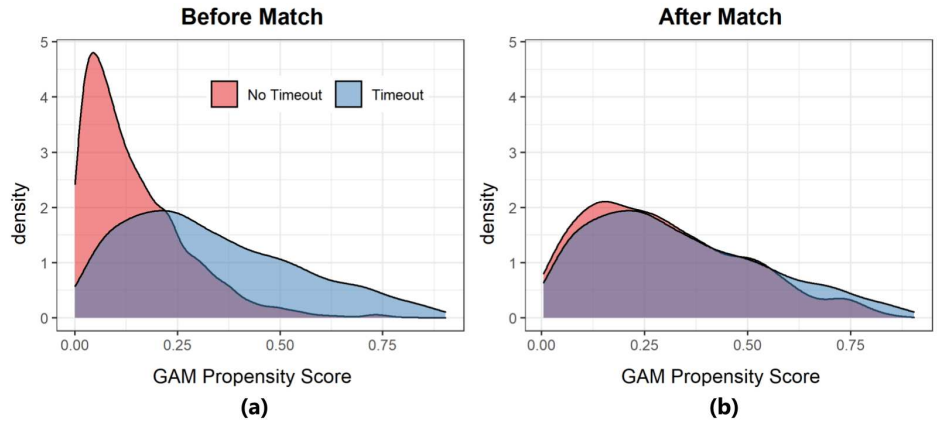


Figure 2.3: Panel (a) shows that the propensity to call a timeout, estimated using the pre-treatment covariates in Table 2.1, differs between the treated (timeout) and control (no timeout) groups. This signifies significant covariate imbalance between the groups, which impedes the ability to compare them directly. Panel (b) shows the distribution of propensity scores after matching are quite similar.

2.3.4 Matching

In seeking to calculate the treatment effect, the naïve approach would be to develop an outcome of interest and calculate the difference in means between the treated units and the control units identified in Section 2.3.2. As seen in Figure 2.3a and discussed in Section 2.3.1, game situations when timeouts are called are fundamentally different from when timeouts are not called. To mitigate the ill effects of self-selection bias, we employ a matching procedure with propensity scores (Lopez and Gutman, 2017; Rosenbaum and Rubin, 1983; Stuart, 2010).

Two common approaches are generally used to balance covariates: matching on the distance between the covariate vectors, such as Mahalanobis distance, or matching based on the estimated propensity scores. Here we employ a hybrid approach introduced by Diamond and Sekhon (2013) that matches based on the distance between the covariate vectors and the propensity scores through

minimization of a generalized version of Mahalanobis distance (GMD). In this approach, the GMD has a weight parameter, and the weights of the covariates and propensity score are chosen to minimize the largest individual discrepancy using p-values from Kolmogorov-Smirnov tests and paired t-tests. This estimation procedure is implemented in the `Matching` package in R using a genetic search algorithm (Sekhon, 2011). Details regarding the function arguments used can be found in Appendix A.3.

In practice, every observation in the treated group is matched to an observation in the group of potential controls in a one-to-many fashion. That is, a potential control can feasibly serve as the control for more than one treatment. As an example, the identified match for the treated unit in Figure 2.1 is a play from February 06, 2019 when the Washington Wizards played the Milwaukee Bucks in Milwaukee. The identified match featured a run of nine points attained in 1.73 minutes. At the time of treatment, there were 38.08 minutes left in the game. In both situations, the BiT team was away and did not have possession at the time of treatment. The probability of calling a timeout (estimated from the GAM) was 0.32 for the RwT versus 0.28 for the matched RwoT. Thus, we see that in both scenarios the covariates were largely comparable. For more detail regarding covariate balance, including the distribution of the covariates by treatment group after matching, see Appendix A.8.

According to the Love plot (Zhang et al., 2019), presented here in Figure 2.4, matching improves the balance among the covariates, except for variables week in season and the signed score difference before and after the run. However, for these three variables, the standardized bias was already near zero, and the change is arguably negligible. Most notably, we observe a marked improvement in the distribution of the propensity scores after matching (see Figure 2.4 and Figure 2.3b). Further, the standardized bias for each variable is less than 0.2 in absolute value, suggesting no covariate imbalance in the matched cohort (Stuart, 2010).

To rigorously assess the balance of the matched cohorts, hypothesis testing is used to check for discrepancy in each of the covariates listed in Table 2.1 as well as the estimated propensity score. For discrete and continuous variables, bootstrapped Kolmogorov-Smirnov tests were ap-

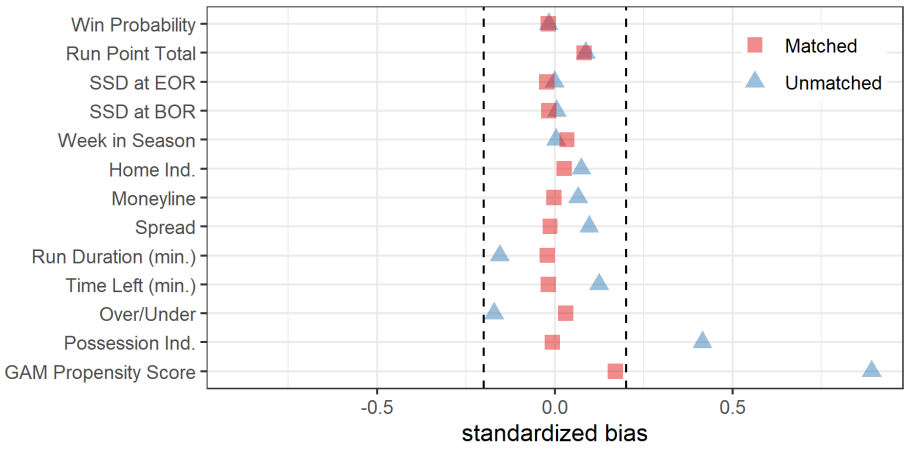


Figure 2.4: Standardized bias is often used to gauge covariate balance before and after matching or other trimming procedures. The propensity to call a timeout is largely imbalanced before implementing the matching algorithm, well past the recommended threshold of ± 0.2. After matching, no covariate exceeds the recommended threshold.

plied, shown to have correct coverage in Abadie (2002). Multiple comparison correction is performed to control the false discovery rate at 0.05 (Benjamini and Hochberg, 1995). For binary variables, t-tests are used, and chi-squared tests are used for categorical variables. Before matching, a discrepancy between the distribution of covariates associated with treatments and controls was identified in all covariates listed in Table 2.1 except for week in season (see Appendix A.8). A distributional discrepancy was also identified in the estimated propensity score. After matching, there was no evidence of a discrepancy in covariate distributions for any of the listed covariates or the estimated propensity score. Hence, genetic matching appears to yield a matched cohort similar in both the covariates and the propensities.

2.3.5 Outcome

The goal of the outcome measure is to quantify the BiT team's response to a run following a timeout (or lack thereof). Naturally, we first considered the change in the score difference from the time of treatment to the end of the post-treatment window. This approach, however, ignores any mid-window scoring, potentially treating two fundamentally different responses the same simply because they share the same start/end points. For example, suppose the BiT team scores five points after the time of treatment followed by five points by the opposing team. This would correspond

to no change in the score difference, and is thus equivalent to a situation when neither team scored during the post-treatment window.

To address this concern, we subsequently considered the most extreme change in the score difference occurring in the post-treatment window. Unfortunately, this method suffers from a similar problem as described above. For example, suppose the opposing team continued their run in the post-treatment window, scoring five points before the BiT team answered with a three-point shot. This response is treated the same as if the opposing team scored five unanswered points in the post-treatment window and the BiT team scored zero. Finally, we considered using the change in the BiT team's win probability during the post-treatment window (Yam and Lopez, 2019), but such an approach relies on the unknown model used to produce this measure, is time variant over the course of a game, and is heavily dependent on the score difference at the time of the treatment.

The clear drawbacks of the initially proposed outcome measures suggested a new method for quantifying a team's response during the post-treatment window was needed. Our goal is to capture how the score difference holistically changes in the entirety of the post-treatment window without punishing (or rewarding) the BiT team for the score difference at the time of treatment. To this end, we develop the outcome for the i^{th} unit, denoted y_i, as the integrated, centered-score difference, defined by

$$y_i = -\operatorname{sgn}\big(s(t_i)\big) \int_{t_i}^{t_i+1} \Big[\Delta(x) - \Delta(t_i)\Big] dx, \tag{2.5}$$

where t_i is the time of the treatment for the i^{th} unit, $s(t_i)$ is the signed run point total at time t_i, and sgn is the sign function. Notice that the integrand is the score difference centered by that at the time of treatment. This ensures we do not penalize (or reward) the BiT team for play occurring before the time of treatment. This outcome measures the BiT team's response to the opposing run following an intervention (either a timeout or no timeout), such that positive values indicate evidence of stopping the run, zero indicates a scoreless response or an even exchange in scoring, and negative values indicate evidence of a continued run. Figure 2.5 shows the outcome for three different units on the same score difference curve.

If the home team is on the run at time t_i then the score difference had a positive trajectory in the pre-treatment window. In this case, a continuation of this increasing trend in the post-treatment window signifies a negative response to the treatment (see Figure 2.5a). Conversely, if the score difference started decreasing after the time of treatment, this would indicate a positive response by the BiT team to the treatment (see Figure 2.5c). The negative sign in the outcome definition in (2.5) ensures the measure aligns with intuition: the first scenario has a negative outcome and the second scenario has a positive outcome.

The sign of the signed run point total is used to create an interpretable outcome, regardless of which team is on the run at time t_i. Contrary to the example just given, if the away team is on the run at time t_i, then the score difference is trending negatively in the pre-treatment window and the signed run point total would be negative, $s(t_i) < 0$. The sign function cancels the negative sign in the front of (2.5), which is desirable as positive trajectories in the score difference in the post-treatment window then result in a positive outcome. If the outcome is zero, there is either no scoring in the post-treatment window or the BiT team exchanged points with the opposing team at an even rate (see Figure 2.5b).

The outcome for the example highlighted in Figure 2.1 is 0.80 and is shown again in Figure 2.6a. Similarly, the outcome for its selected match is -1.93 and is shown in Figure 2.6b. The key desirable property of this outcome is its ability to classify the response to the timeout on a spectrum where negative values indicate failure in stopping the run, zero indicates no change of score or an even exchange of scoring, and positive values indicate success in reversing the run. The larger the magnitude of the outcome, the more extreme the response to the run.

2.4 Results

With a balanced, matched cohort of treated and control units, we turn to estimating the causal effect of a timeout. Histograms for the outcomes in the treatment and control groups are given in Figure 2.7. First, note that a large portion of the outcome distribution for both the treated and control cohorts is greater than zero, indicating that the BiT team often has some level of a

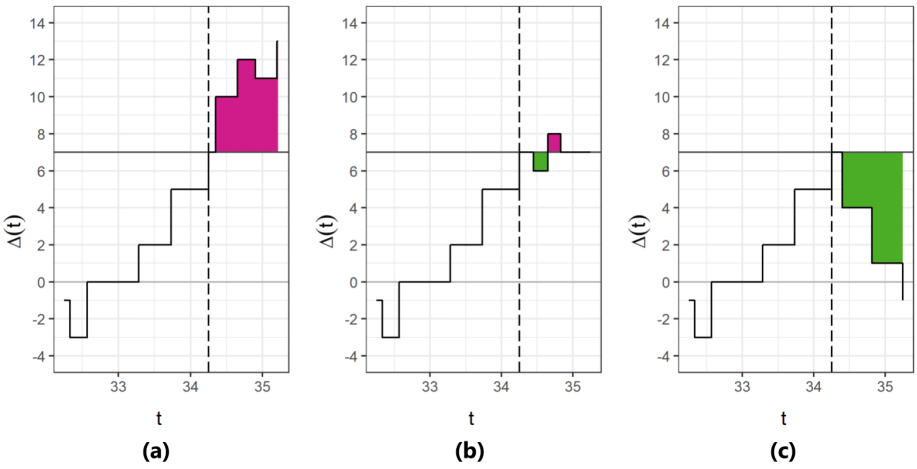


Figure 2.5: The shaded area represents the integral in (2.5). Panel (a) shows the score difference changed at a rate of 3.47 points per minute for the opposing team during the post-treatment window, signifying a negative response to the run following the timeout. Panel (c) shows a score difference that changed at a rate of 4.08 points per minute for the BiT team, a positive response to the run following the timeout. In the panel (b), the opposing team and the BiT team swapped points at a near even rate, neither a negative nor a positive response to the run following a timeout.

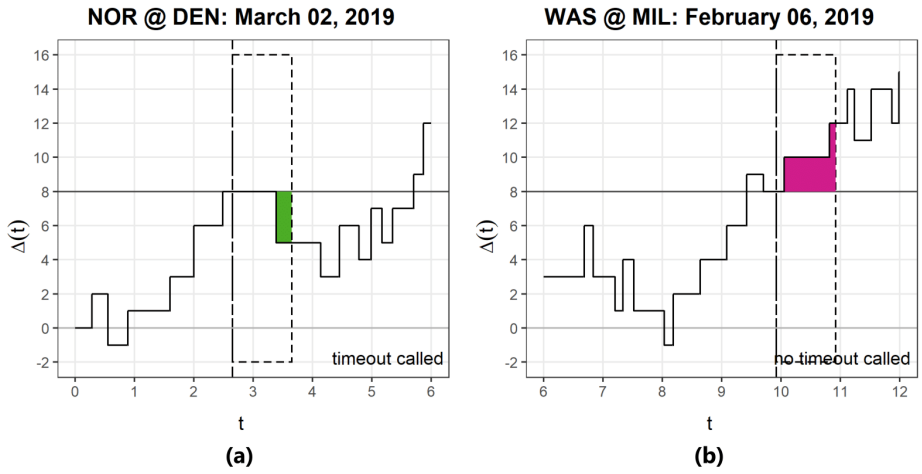


Figure 2.6: The outcome for the timeout highlighted in Figure 2.1 is shown in panel (a), and the matched control is shown in panel (b). In panel (a), the New Orleans Pelicans positively responded to the Denver Nuggets run when a timeout was called; however, within the selected match, the Washington Wizards negatively responded to the Milwaukee Bucks' run when no timeout was called. The dashed regions outline the post-treatment window for each unit.

comeback, regardless of whether there is stoppage in play. We interpret this as evidence that momentum shifts are common, and thus should be expected.

The estimate of the average treatment effect on the treated is visualized by the difference between the mean of the treated group and the mean of the control group of the matched cohort (shown by the dashed lines in Figure 2.7). The estimated average treatment effect on the treated is -0.35 with an Abadie-Imbens standard error (Abadie et al., 2004) of 0.07 and an associated p-value of $p < 0.001$. Hence, on average, it appears slightly disadvantageous to call a timeout in the presence of an opposing run. While these results may seem counterintuitive, the negative estimate aligns with the unmatched, naïve estimate of -0.08. This measured effect, however, is insignificant until employing formal causal methodology to account for the inherent self-selection bias associated with observational studies.

Due to the non-deterministic nature of the matching algorithm, we re-ran the matching step in our analysis to study the variability in the estimated ATT due to different matched cohorts. Twenty realizations of the estimated ATT are presented in Appendix A.6, indicating that results were consistent across different matched cohorts.

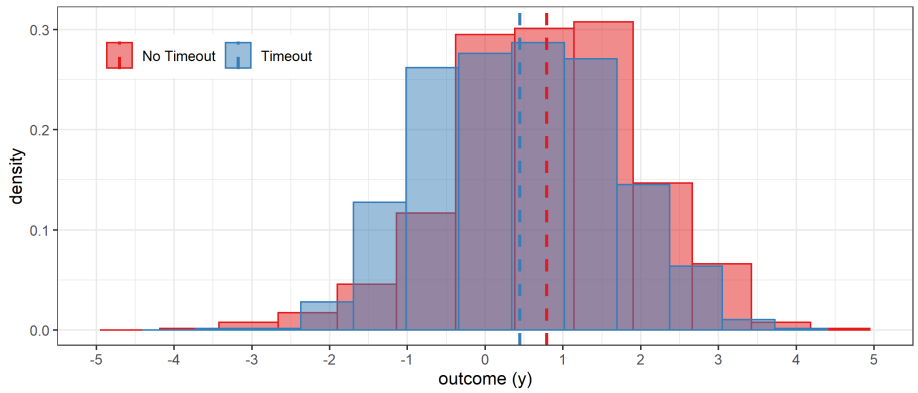


Figure 2.7: The distribution of the outcome for each intervention group within the matched cohort. The estimated average treatment effect of -0.35 is given by the mean of the treated group (blue vertical line) minus the mean of the control group (red vertical line).

Acknowledging that the effectiveness of a timeout in stopping an opposing run may vary across teams due to, say, the maturity of the players or the ability of the coach to strategize a comeback, we estimated the ATT for each franchise individually. The average treatment effect on

the treated for franchise f, ATT_f, is defined

$$\text{ATT}_f = \mathbb{E}\left[Y_i(1) - Y_i(0) \mid T_i = 1, \mathcal{B}_i = 1\right], \tag{2.6}$$

where $\mathcal{B}_i = \mathbb{I}_{\{f \text{ is the BiT team for the } i^{th} \text{ unit}\}}$. The matching procedure does not guarantee covariate balance within franchise. Therefore, we employ a hypothesis testing approach to assess whether or not the covariates are sufficiently balanced when conditioning on the BiT team's identity (see Appendix A.8). After controlling the false discovery rate at 0.05 to account for multiple comparisons (Benjamini and Hochberg, 1995), there are no significant findings, indicating it is reasonable to assume the treatment is ignorable when conditioning on the BiT team's identity.

To estimate ATT_f for each franchise, the matched treated and control units were partitioned based on the BiT franchise of the treated unit and the average within-matched-set mean differences in outcome was computed. Some franchises obviously were associated with more treated units than others (see Appendix A.7), so bootstrapped samples were created to quantify the variability of the causal estimator in (2.6), as in Yam and Lopez (2019). The results are given in Figure 2.8.

Immediately, we notice that twenty of the thirty franchises exhibit negative point estimates for their franchise average treatment effect on the treated. For these franchises, the negative estimated effect suggests, on average, the opposing team scores at a faster rate when a timeout is called than when it is not called during an opposing run. To assess statistical significance of these estimates and address the multiplicity problem, paired permutation tests are conducted for each of the franchises, and the two-sided p-values are recorded. After computing the unadjusted p-values, we employ the strategy suggested in Benjamini and Hochberg (1995) to control the false discovery rate at 0.05.

After accounting for multiple comparisons, there are two significant, negative franchise treatment effects on the treated: those corresponding to the Indiana Pacers and the Utah Jazz. Relatively small sample sizes may have contributed to the lack of significant findings after accounting for multiple testing. None of these positive effects withstand statistical significance. Data and code to reproduce results are provided in Appendix A.9.

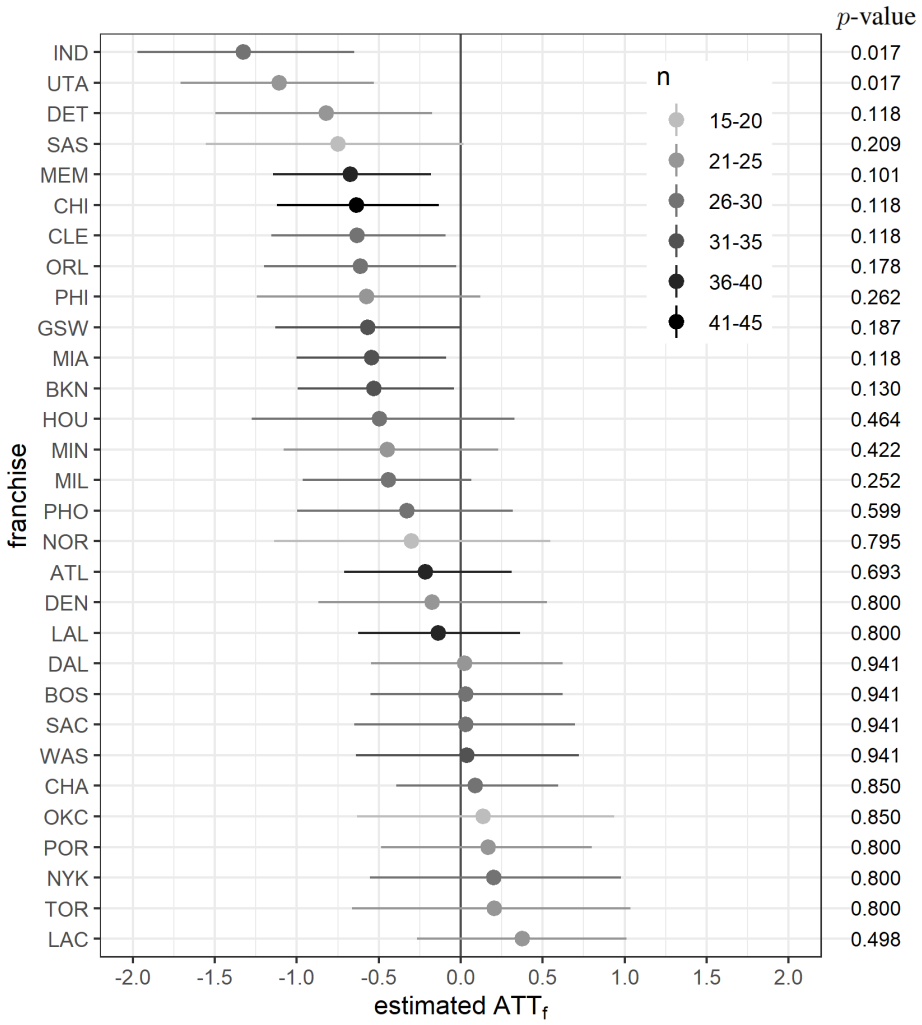


Figure 2.8: Estimated average treatment effect for the treated for each franchise with a 95% confidence interval based on a non-parametric bootstrap. Ignoring the multiplicity problem, there are ten significant results at a significance level of 0.05. Computing the unadjusted p-values with paired permutation tests and controlling the false discovery rate at 0.05, the adjusted p-values are provided on the right margin. After accounting for multiple testing, the Indiana Pacers and the Utah Jazz have significant, negative average treatment effects on the treated. For these franchises, the opposing team scores at a significantly faster rate concluding a timeout than when a timeout is not called during an opposing run.

2.5 Sensitivity Analysis

A sensitivity analysis is employed to assess the robustness of the results (1) with regard to the specification of a run, and (2) in the presence of unobserved confounding. In this work, a run is characterized not only by the magnitude of the change in the score difference (run point total) but also the time taken by the opposing team to attain said change (run duration). While few would argue the importance of these two attributes in defining a run, there is no universal agreement on what values constitute a run. We define a play to be a run if it features a nine-point change in the score difference attained in the prior two minutes. Considering the average NBA possession is conservatively 15 seconds (Beuoy, 2021), a two-minute window allows for approximately eight possessions, or four possessions per team. The current definition was motivated by the simple scenario: four defensive stops and at least three three-point shots made yields a drastic change in the game in a short amount of time. We acknowledge that this definition is arbitrary and should be explored more carefully. Therefore, we examine the sensitivity of our results relative to different combinations of run point total thresholds (7, 8, 9, and 10 points) and limits to the run duration (1.5, 2, 2.5, 3 minutes). For each combination, we replicate the analysis and report the findings in Table A.1 of Appendix A.2. As can be seen, the estimated effect is negative and significant for each combination, indicating robustness to alternative run definitions.

Aside from the run specification, it is important to consider the robustness of the perceived effects to unmeasured or unobserved confounders. While matching methods can adjust for observed confounding (assessed through covariate balance before and after matching), the impact of unobserved confounding on the estimated effect must be explored through a sensitivity analysis (Rosenbaum, 2007). Following the recommendations of Rosenbaum (2013), we let Γ represent the magnitude of the bias from nonrandom treatment assignment and estimate the ATT and confidence interval for the ATT as in (2.4) at various Γ. The results are illustrated in Figure 2.9. When $\Gamma \approx 1.50$, the confidence interval for the treatment effect includes 0, indicating a materialistic change in the inferential conclusions of this study. While these results imply sensitivity to unobserved confounding, it is important to note that the 95% confidence interval is known to be

conservative (Rosenbaum, 2015). As stated by Liu et al. (2013), this method assumes a situation in which “the unobserved confounder perfectly predicts the outcome of interest," an unrealistic assumption in practice. As a result, sensitivity to unobserved confounders is likely overstated.

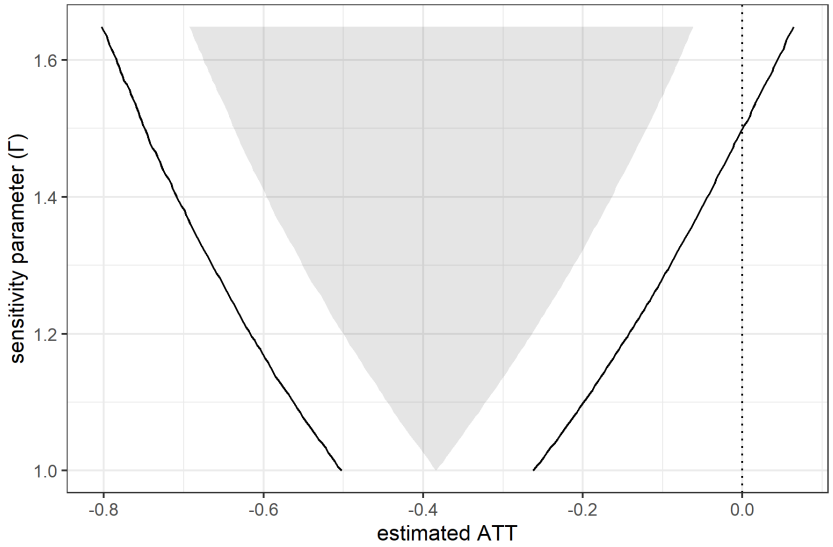


Figure 2.9: Provided a bias of Γ in the treatment assignment, the shaded region illustrates the interval of point estimates possible, and the solid black lines illustrate the (conservative) 95% confidence interval. The confidence interval includes 0, indicating no effect, at $\Gamma \approx 1.50$, the magnitude of bias from nonrandom assignment necessary to alter the conclusions herein. While these results indicate the study may be sensitive to unobserved confounding, the degree to which is likely overstated (Rosenbaum, 2015).

2.6 Discussion

While the idea of a “run” is commonly used within the context of basketball, there is no formal mathematical definition. Part of the novelty of this work is formalizing the colloquial understanding of a run as it pertains to professional basketball, while developing an interpretable outcome which captures the relative performance of each team in a game where the score changes frequently. After proposing a framework with which to study runs, we employed causal methods to estimate the potential gain (or loss) attributed to a timeout during a team’s run. We find that, on average, calling a timeout worsened the non-run team’s short-term performance compared to if no timeout was taken during an opposing run. In particular, the Indiana Pacers and the Utah Jazz

short-term performance significantly declines from a timeout compared to if no timeout was taken, on average. No teams, on average, exhibit a significant gain in their short-term performance from a timeout compared to if no timeout was taken.

One important variable which is not considered within this analysis is substitutions that occur when a timeout is called. These are reasonably assumed to have a positive impact on the outcome but cannot be included within the matching procedure since these substitutions occur after or simultaneously with the treatment. That said, more substitutions actually exist within the group of treated units, when a timeout is called, than in the group of control units, when a timeout is not called. This suggests the effect of a timeout may actually be less than that estimated in this analysis.

There are choices made in defining units which could be addressed by generalizing the causal methods used. For example, plays occurring in the last minute of a period are excluded from this analysis. Since the outcome requires a succeeding one minute of game time after intervention, any play occurring in the last minute of a period is akin to a censored observation and is thus omitted. Currently, for a given play at time t, if there are timeouts in the pre-treatment window (two minute interval of time prior to t), then that play is excluded from the analysis on the basis of multiple variations. The number of timeouts and the time between timeouts could be used to create a more general, non-dichotomous treatment regime and is an area of future work.

Causal inference is becoming a popular tool for sports analysts due to the observational nature of sporting events. However, despite the rise in popularity, too often little emphasis is placed on closely examining whether the stable unit-treatment-value assumption (SUTVA) is reasonable. In this work, we prioritize this discussion by not only defining units but describing how their definition was motivated by adherence to SUTVA. Aside from the honest exposition surrounding SUTVA, we also take great care in defining an interpretable outcome which measures relative short-term performance well. This outcome is flexible and well suited for applications where the quantity to measure is susceptible to short term fluctuations, but the interest is in the average

change. Other phenomena that might benefit from the modeling tools introduced here include stock prices, approval ratings, and of course, score differences in professional sports.

Chapter 3

The Identification of Network Clusters Containing Undiscovered Gene to Phenotype Relations

3.1 Introduction

Since the Hippocratic Corpus, scholars have continually searched for the causes of disease, and in many cases, have succeeded. In 1989, a group of investigators led by Dr. Lap-Chee Tsui identified a mutation to a gene that mediates the transport of chloride and sodium ions across the membranes of cells controlling mucus, sweat and digestive juices as the cause of cystic fibrosis (CF) (Busch, 1990). This discovery marked the first disease causing gene to be identified for any disease. Since then, thousands of such diseases known as *genetic disorders* have been identified (e.g., Crohn's disease, Huntington's disease, and sickle cell disease) (Cleveland Clinic, 2021) with many more undoubtedly undiscovered.

Genetic disorders are caused by mutations to the genome contained in the deoxyribonucleic acid (DNA) of chromosomes. The Human Genome Project estimates there are between 20,000 to 25,000 genes in the human genome, many of which are responsible for the synthesis of a protein (International Human Genome Sequencing Consortium, 2004). Since proteins direct cellular function, alterations to the DNA of a gene (known as *genetic variants*) can have severe consequences on one's function and health in the form of abnormal *phenotypes*. As with other diseases, a genetic disorder is often characterized in the context of its abnormal phenotypes: observable characteristics or traits associated with a disease. For example, some abnormal phenotypes associated with CF include elevated sweat chloride, exocrine pancreatic insufficiency, and asthma, each reasonably associated with the underlying mechanism for the disease. While a disease's phenotypes need not imply the mechanism of the disease, understanding the relationships among and between one's genotypic and phenotypic information can help practitioners improve diagnosis and management

for patients with undiagnosed conditions. In this chapter, we present a new method for the identification of collections of genes and phenotypes ripe for future discovery.

The current state of knowledge about phenotypes and the human genome are summarized by various *biological ontologies*, described in Bard and Rhee (2004) as "formal way[s] of representing knowledge in which concepts are described both by their meaning and their relationship to each other." Ontologies provide medical practitioners with a common vocabulary with which to diagnose and treat patients. Two examples are the Human Phenotype Ontology (HPO) (Köhler et al., 2017) and the Search Tool for Retrieval of Interacting Genes/Proteins (STRING) (Szklarczyk et al., 2016). The genetic variant that causes cycstic fibrosis—the CF transmembrane conductance regulator—is denoted as CFTR in STRING, and some common cystic fibrosis phenotypes—elevated sweat chloride, exocrine pancreatic insufficiency, and asthma—are denoted as HP:0012236, HP:0001738, and HP:0002099 in HPO, respectively.

Aside from standardizing medical terminology, ontologies also provide researchers with a tractable way to represent and study biological processes. The known relationships among and between phenotypes and the human genome can be integrated into a heterogeneous network (i.e., graph), which consists of the biological entities (i.e., nodes), connections between them (i.e., edges), and known characteristics about the biological entities and the connections (i.e., node and edge attributes). The network representation is deemed heterogeneous since the biological entities are distinguished by their identity in the network (i.e., node types)—some representing genes and others representing phenotypes. By combining the data of various biological ontologies into a single, heterogeneous network, we are able to potentially exploit the mesoscopic topology of the network (i.e., the intermediate topological scale of organization between nodes) to improve diagnostic reach. In particular, nodes naturally tend to link together or "cluster" into groups which are densely connected within group yet sparsely connected to the rest of the network. These latent clusters are often related to the functionality of the group and may aid a practitioner in diagnosis when their patients' medical information is seemingly unconnected.

Many methods exist to identify missing or undiscovered links in real, biological networks but rely on local or global measures of topology. Classically, the potential or propensity for an unconnected node pair to form a connection is scored via local measures of connectivity, such as the number of shared neighbors, the Jaccard coefficient of neighbor sets, or the preferential attachment (i.e., the degree product) between pairs of nodes (Liben-Nowell and Kleinberg, 2003). Similarly, another assortment of methods score absent links via global measures of connectivity, such as the shortest path length, random walk (with restart), or the Katz Index (Martínez et al., 2016). A recent class of methods uses community structure as a proxy for the mesoscopic topology of a network, either scoring absent links according to a partition of a network into communities or identifying links as missing if their proposed existence improves measures of community structure (Ghasemian et al., 2020).

In this chapter, we propose a novel analytic pipeline for the **i**dentification of **c**lusters **c**ontaining **u**ndiscovered gene to phenotype **r**elations (ICCUR) in a very large, temporal heterogeneous biological network. In particular, ICCUR identifies and ranks subgraphs (clusters) of a network according to their predicted potential for future discovery using insights drawn from the clusters of the prior snapshot of the network. ICCUR can be considered a new approach to link prediction using community structure with mechanisms to score clusters rather than individual node pairs. We demonstrate ICCUR's ability to identify clusters with co-occurring genes and phenotypes linked together in the later snapshot of the network more than expected by chance. Importantly, the results of the pipeline can be potentially used as a diagnostic aid by clinicians as it will return a collection of ranked clusters pertaining to a patient's known genetic variants and abnormal phenotypes. We verify ICCUR's utility as a diagnostic aid by applying it to data from MyGene2, a web-based platform which connects persons with rare, undiagnosed diseases to clinicians or researchers working with individuals with similar medical profiles. We demonstrate ICCUR's ability to identify and recommend novel clusters containing unconnected phenotypic and genotypic information of patients with undiagnosed disease.

The organization of this chapter is as follows. In Section 3.2, we discuss the temporal biological network of interest and its construction. In Section 3.3, we introduce the ICCUR pipeline, including (1) how to identify dense, reasonably sized heterogeneous clusters, (2) how to characterize clusters' undiscovered relations and score their potential for future discovery, (3) how to predict clusters' potential for future discovery using observable cluster features, and (4) how to rank clusters according to model predictions. We fit ICCUR pipeline to the temporal biological network and validate the ICCUR pipeline in Section 3.4. In Section 3.5, we apply the results of the ICCUR pipeline to real, undiagnosed patients with rare disease, identifying clusters containing patients' co-occurring yet otherwise unconnected genotypic and phenotypic information, some connections which have since been validated by human curation. We conclude with a discussion in Section 3.6.

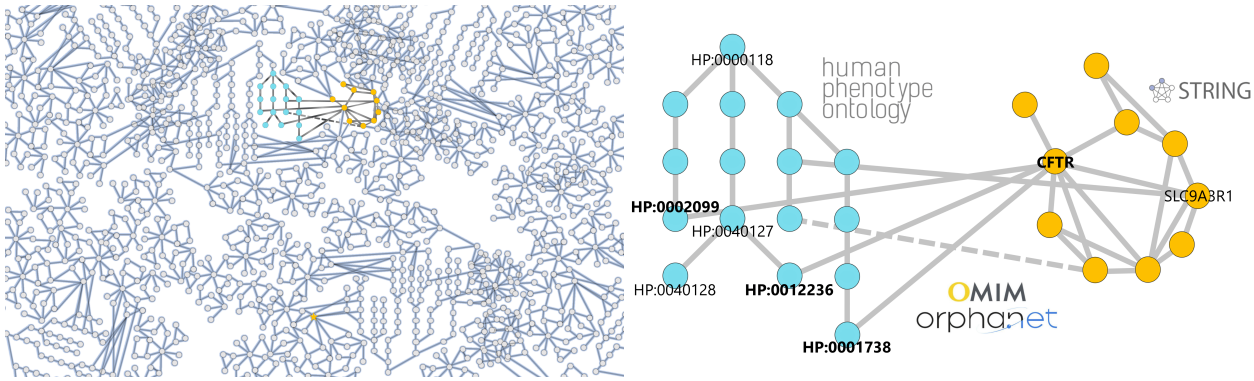
3.2 Data and Notation

A genetic disorder is often characterized by the abnormal phenotypes and genetic variants associated with the disease. Known relationships between and among phenotypic and genotypic information are provided by biological ontologies. In this section, we discuss the two biological ontologies utilized in this work, outline how these ontologies are combined into a large, temporal heterogeneous biological network, and provide formal notation. A mockup of a subset of the network is provided in Figure 3.1 to illustrate the structure of the network and underlying data sources.

3.2.1 Biological Ontologies

Human Phenotype Ontology (HPO)

The Human Phenotype Ontology (HPO) (Köhler et al., 2017) integrates phenotypic information across scientific fields and databases, providing resources for the analysis of human disease. By design, HPO is a directed acyclic graph (DAG), where each node is a phenotype and edges indicate that the receiver node (child) is a subclass of sender node (parent). Each child is a more specific instance of its parent phenotype. For example, patients presenting with elevated sweat chloride and



(a) Mockup of $\mathcal{G}_{t-1}$ **(b)** Subgraph of $\mathcal{G}_{t-1}$

Figure 3.1: Panel (a) is a mockup used to represent the biological network at time $t-1$. A small subgraph of the mockup is colored and provided in panel (b) to illustrate the data sources and characteristic features of the real network. Dashed lines are used to represent edges absent from the network at time $t-1$ subsequently added to the network at time t; that is, undiscovered relations at time $t-1$. Phenotype (in blue) to phenotype (P2P) relations are derived from HPO, and gene (in orange) to gene (G2G) relations are derived from STRING. Gene to phenotype (G2P) relations are derived from HPO using information from OMIM and Orphanet.

exocrine pancreatic insufficiency may be noted for phenotypes HP:0012236 and HP:0001738 in HPO, respectively. As illustrated in Figure 3.1b, elevated sweat chloride is the child of abnormal sweat homeostasis (HP:0040127), which is also the parent of abnormal sweat electrolytes (HP:0040128). Elevated sweat choloride is distantly related to exocrine pancreatic insufficiency; the shortest path between the two phenotypes traverses through the root node—phenotypic abnormality (HP:0000118)–with a shortest path length of nine. As of 2021, HPO documented 20,279 relationships between 16,041 phenotypic abnormalities. There is one connected component, meaning there exists a path from any node in the network to any other node in the network. The average shortest path length between pairs of phenotypic abnormalities is 10.2, and the ratio of relationships present to total number of possible relationships (i.e., edge density) is less than 0.001.

Search Tool for Retrieval of Interacting Genes/Proteins (STRING)

The Search Tool for Retrieval of Interacting Genes/Proteins (STRING) (Szklarczyk et al., 2016) integrates publicly available knowledge on protein-protein interactions, providing a large scale model for cellular processes in humans. By design, STRING is graph where nodes are genes and an edge exists between two genes if they encode interacting proteins. All protein-

protein interactions are determined based on experimental data, computational prediction methods, or public text collection. For example, CFTR (the CF causing gene) is connected to SLC9A3R1 (another protein coding gene) in STRING (see Figure 3.1b). This connection in part stems from the work of Castellani et al. (2012) which demonstrated that SLC9A3R1 codes for a protein "involved in PKA-dependent activation of CFTR." As of 2021, STRING documented nearly six million relationships between 19,384 genes. There are two connected components, which means that there are two separate subgraphs or portions of the network that have no connection or link to each other. In other words, there is no way to reach a node in one component from a node in the other component through the edges or connections in the network. The average shortest path length between pairs of genes is 2.041, and the edge density is less than 0.031. When computing the average shortest path length, we do not take into account the connections between nodes that belong to different components. In other words, we only consider the node pairs within the same component to calculate the average shortest path length.

Online Mendelian Inheritance in Man (OMIM) and Orphanet

In this chapter, connections between genotypic and phenotypic information are derived from annotation files produced by HPO. HPO makes use of the Online Mendelian Inheritance in Man (OMIM) (Hamosh et al., 2005) and Orphanet (Weinreich et al., 2008) databases, each providing information on disease, genes, and genetics information. OMIM summarizes a genetically determined phenotype and provides hyperlinks to other databases, such as DNA and protein sequences, PubMed references, and many others. Orphanet provides information on rare diseases, such as the related genetic variants and phenotypic information associated with each disease. As illustrated in Figure 3.1b, CFTR is linked to exocrine elevated sweat chloride, exocrine pancreatic insufficiency, and asthma—those phenotypic abnormalities commonly reported by patients diagnosed with cystic fibrosis; however, in total, CFTR is linked to 83 phenotypic abnormalities. As of 2021, HPO documented nearly 197,926 relationships between 8,753 phenotypic abnormalities and 4,702 genes using information from OMIM and Orphanet. There are ten connected components. The average shortest path length between a phenotypic abnormality is 3.428, and the edge density is 0.004.

Similarly to STRING, we only consider the node pairs within the same component to calculate the average shortest path length.

3.2.2 Network Construction

Ontologies are updated periodically to reflect current understanding of biology. We create a heterogeneous network for each year 2019, 2020, and 2021 by merging the labeled nodes and edges of each biological ontology discussed in Section 3.2.1 at each snapshot. Note, we have three snapshots to demonstrate and validate the ICCUR pipeline. Each network is composed of genes (e.g., CFTR), phenotypes (e.g., HP:0012236), and the relationships both among and between them, reflecting the state of knowledge at the end of 2019, 2020, and 2021. Gene to gene (G2G) relations are constructed from STRING. Phenotype to phenotype (P2P) relations are constructed from HPO, and gene to phenotype (G2P) relations are constructed from HPO using information from OMIM and Orphanet, as shown in Figure 3.1b. The edges are encoded as binary and undirected; that is, any relationship weight or direction is subsequently ignored.

The temporal network is large, consisting of roughly 35,000 nodes and 6,000,000 edges at each snapshot. Earlier generations of the network tend to have fewer nodes and edges than their successors, and thus the average internal degree is negligibly decreasing with time. The network at each snapshot has two connected components. Additional information about the network at each snapshot, including the access dates for each ontology, is provided in Table 3.1.

Table 3.1: Information about the temporal network, $\mathcal{G}_\tau$, including the number of nodes ($|\mathcal{V}_\tau|$), number of edges ($|\mathcal{E}_\tau|$), and average degree (Δ_τ) at each snapshot $\tau \in \{19, 20, 21\}$ denoting the year. Each network is composed of data collected from HPO, OMIM/Orphanet, and STRING with the listed dates of access.

					Access Date for Data Sources		
Snapshot	Network	$\lvert\mathcal{V}_\tau\rvert$	$\lvert\mathcal{E}_\tau\rvert$	Δ_τ	HPO	OMIM/Orphanet	STRING
2019	$\mathcal{G}_{19}$	33.9K	5.9M	350.35	2019-11-08	2019-09-02	2020-10-17
2020	$\mathcal{G}_{20}$	34.9K	6.1M	347.92	2020-12-07	2020-08-25	2021-08-12
2021	$\mathcal{G}_{21}$	35.7K	6.2M	346.61	2021-10-10	2021-10-10	2022-09-30

3.2.3 Notation

Let $\mathcal{G}_\tau = (\mathcal{V}_\tau, \mathcal{E}_\tau)$ denote the undirected, simple, binary biological network at snapshot τ. The node set at time τ is $\mathcal{V}_\tau = (1_\tau, 2_\tau, \ldots, n_\tau)$, where n_τ is the total number of nodes at time τ. Each node is either a phenotype or a gene. The identity of each node in the network (i.e., a phenotype or a gene) is referred to as its node type. We let $\mathcal{T}_\tau$ denote an n_τ-dimensional vector of node types at time τ where the u_τ^{th} element of $\mathcal{T}_\tau$ is a 1 if node u_τ is a phenotype; otherwise, the u_τ^{th} element of $\mathcal{T}_\tau$ is a 2. When necessary, the node type of an arbitrary node u_τ is denoted in bracketed superscript, such that $u_\tau^{[1]}$ implies that node u_τ is a phenotype and $u_\tau^{[2]}$ implies that node u_τ is a gene.

Since each node is either a gene or a phenotype, the node set can be partitioned as $\mathcal{V}_\tau = V_\tau^{[1]} \cup V_\tau^{[2]}$ where $V_\tau^{[1]}$ denotes the node set containing $|V_\tau^{[1]}|$ phenotypes, $V_\tau^{[2]}$ denotes the node set containing $|V_\tau^{[2]}|$ genes, and $V_\tau^{[1]} \cap V_\tau^{[2]} = \emptyset$. The network $\mathcal{G}_\tau$ is a simple, binary network meaning that there is either precisely zero or one edge between each pair of nodes. The edge set of $\mathcal{G}_\tau$ can be partitioned according to adjacent nodes' types: $\mathcal{E}_\tau = \cup_{1 \le k \le l \le 2} E_\tau^{[kl]}$ where $E_\tau^{[kl]}$ contains the links between nodes of type k and nodes of type l. If $v_\tau^{[k]}$ is adjacent to $u_\tau^{[l]}$ at time τ, then $\{v_\tau^{[k]}, u_\tau^{[l]}\} \in E_\tau^{[kl]}$. Since $\mathcal{G}_\tau$ is undirected $E_\tau^{[kl]} = E_\tau^{[lk]}$.

3.3 Methodology

In this section, we introduce the analytic pipeline to **i**dentify **c**lusters **c**ontaining **u**ndiscovered gene to phenotype **r**elations (ICCUR). Presented in Algorithm 3.1, the ICCUR pipeline takes as input two snapshots of a temporal, biological network, $(\mathcal{G}_{t-1}, \mathcal{G}_t)$, and returns as output a ranked collection of densely connected subgraphs of $\mathcal{G}_t$, where the ranking is according to the potential for future discovery of the subgraphs based on insights drawn from the identified subgraphs of $\mathcal{G}_{t-1}$. The resulting ranked collection of subgraphs can be filtered according to patients' phenotypic and genetic information, serving to aid clinicians in diagnosis when patients' medical information is seemingly unconnected.

The ICCUR pipeline is composed of four operations: **Discover**, **Score**, **Train**, and **Rank**, each illustrated in Figure 3.3.1. To describe these operations, we consider an arbitrary biological

ICCUR Pipeline

Inputs: Graphs $\mathcal{G}_{t-1} = (\mathcal{V}_{t-1}, \mathcal{E}_{t-1})$ and $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$.
Return: Ranked collection of heterogeneous clusters $\boldsymbol{\mathcal{C}}^t_{\text{ord}}$.

$\boldsymbol{\mathcal{C}}^{t-1} = \{\mathcal{C}^{t-1}_h\}_{h\in[H]} \leftarrow$ **Discover**$(\mathcal{G}_{t-1})$.
For each $h \in [H]$ **do :**
  $\tilde{\pi}^{t-1}_h \leftarrow$ **Score**$(\mathcal{G}_{t-1}, \mathcal{C}^{t-1}_h, E^{[12]}_t - E^{[12]}_{t-1})$.
Let $\tilde{\boldsymbol{\pi}}^{t-1} = \{\tilde{\pi}^{t-1}_h\}_{h\in[H]}$.
$f^{t-1} \leftarrow$ **Train**$(\boldsymbol{\mathcal{C}}^{t-1}, \tilde{\boldsymbol{\pi}}^{t-1})$.
$\boldsymbol{\mathcal{C}}^t = \{\mathcal{C}^t_w\}_{w\in[W]} \leftarrow$ **Discover**$(\mathcal{G}_t)$.
$\boldsymbol{\mathcal{C}}^t_{\text{ord}} = \{\mathcal{C}^t_{w(i)}\}_{i\in[W]} \leftarrow$ **Rank**$(\boldsymbol{\mathcal{C}}^t, f^{t-1})$.

Algorithm 3.1: The ICCUR pipeline identifies a collection of heterogeneous clusters from a biological network $\mathcal{G}_t$ ranked according to their potential for undiscovered gene to phenotype relations using insights drawn from the previous snapshot of the biological network, $\mathcal{G}_{t-1}$. The operations Discover, Score, Train, and Rank are illustrated in Figure 3.3.1 and discussed in Section **??**.

network composed of genes and phenotypes, denoted $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Let $\boldsymbol{\mathcal{B}}$ denote an arbitrary collection of sets of nodes; that is, $\boldsymbol{\mathcal{B}} = \{\mathcal{B}_w\}_{w\in[W]}$ such that $\mathcal{B}_w \subseteq \mathcal{V}$ and $z_w := |\mathcal{B}_w|$ for all $w \in [W] := \{1, 2, \ldots, W\}$. Furthermore, let $\mathcal{R}$ denote a subset of the collection of unobserved edges in $\mathcal{G}$, excluding possible self-loops; that is, $\mathcal{R} \subseteq \mathcal{V} \times \mathcal{V} - \mathcal{E}$.

We describe each operation of ICCUR in Section **??** before detailing the pipeline in Section 3.3.2. First, we describe how to identify small, densely connected clusters composed of genes and phenotypes from $\mathcal{G}$ (see Figure 3.2a (Discover)). Second, we introduce a metric called the *potential for future discovery*, used to compare the number of undiscovered relations, $\mathcal{R}$, between nodes in $\mathcal{B}_w$ for all $w \in [W]$ (see Figure 3.2b (Score)). Third, we detail how to train an ensemble model of boosted regression trees to predict $\mathcal{B}_w$'s potential for future discovery using observable features about $\mathcal{B}_w$ for all $w \in [W]$, particularly useful when the potential for future discovery is unobservable (Figure 3.2c (Train)). Finally, we detail how to rank sets of nodes in $\boldsymbol{\mathcal{B}}$ according to model predictions (see Figure 3.2d (Rank)).

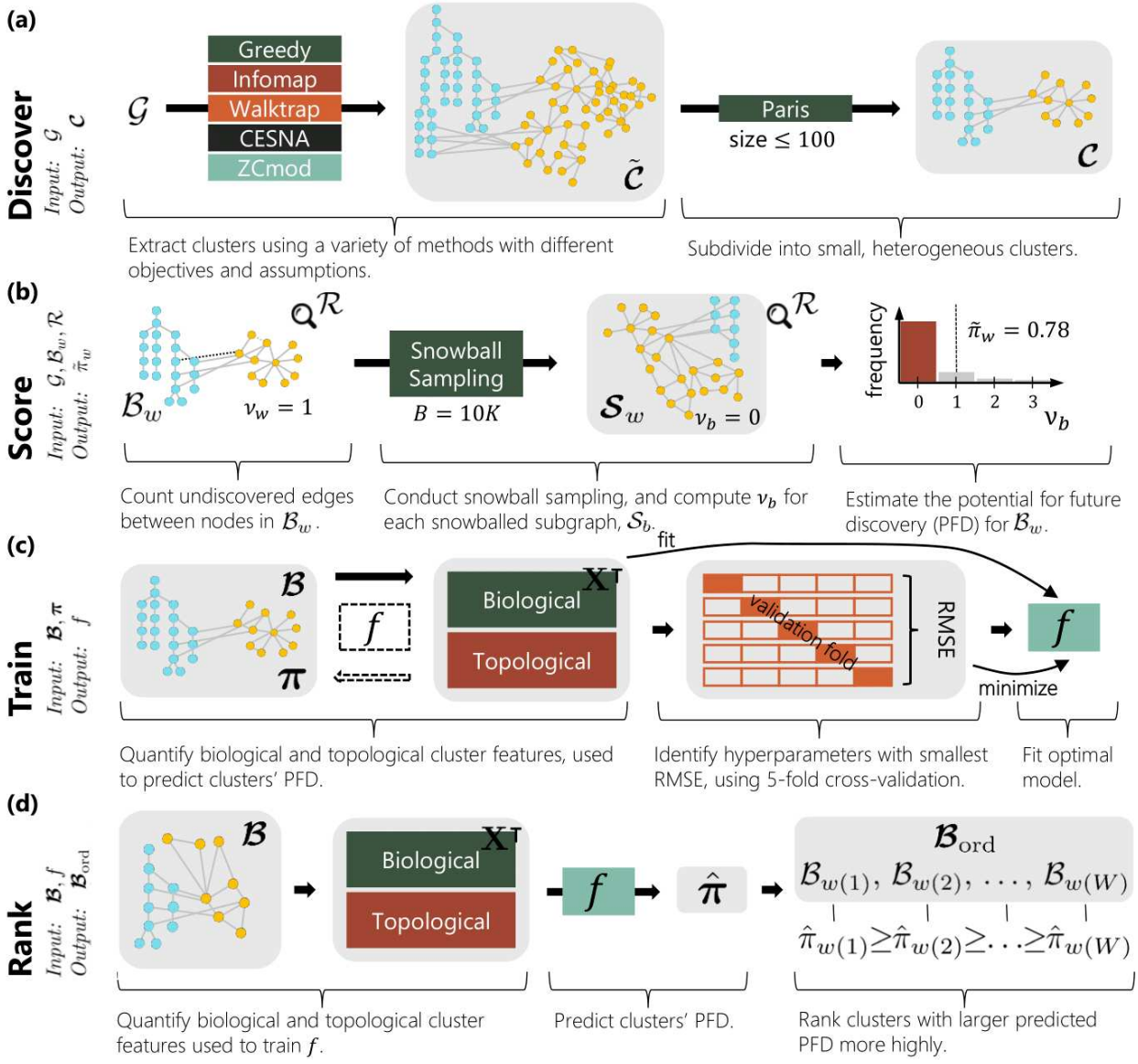


Figure 3.2: Four operations used in the ICCUR pipeline. **Discover** in panel (a) identifies a collection of small, densely connected heterogeneous clusters from a biological network, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. **Score** in panel (b) counts the number of undiscovered relations, $\mathcal{R}$, between nodes in $\mathcal{B}_w$, denoted ν_w, and quantifies the rarity of ν_w with a metric called the potential for future discovery, denoted π_w. **Train** in panel (c) constructs a flexible regression model to relate π_w for each $\mathcal{B}_w$ to observable features about $\mathcal{B}_w$. Lastly, **Rank** in panel (d) ranks each $\mathcal{B}_w$ with predictions made from a model f.

3.3.1 ICCUR Toolpack

Discover Dense, Heterogeneous Clusters

Provided a biological network $\mathcal{G}$, the Discover operation returns a collection of small, densely connected heterogeneous clusters identified from $\mathcal{G}$. The collection of clusters is denoted $\boldsymbol{\mathcal{C}} = \{\mathcal{C}_h\}_{h\in[H]}$ such that $\mathcal{C}_h \subseteq \mathcal{V}$, $\mathcal{C}_h \cap V^{[k]} \neq \emptyset$ for $k \in \{1, 2\}$, and $\mathcal{G}[\mathcal{C}_h]$ (i.e., the induced subgraph) is particularly dense for all $h \in [H]$. To attain such a collection of sets, we rely on existing *community discovery* methods. Community discovery (i.e., clustering) is the general process of attaining communities (i.e., clusters) from a network: collections of nodes that are densely connected within yet sparsely connected to the rest of the network.

There are many existing methods to identify communities from simple, homogeneous networks (see Fortunato (2010); Lancichinetti and Fortunato (2009) for a review). By comparison, there are relatively few methods for simple, heterogeneous networks, designed to account for differences in nodes and the resulting differences in the connectivity patterns between pairs of node types (e.g., Yang et al. (2013); Zhang and Chen (2020)). Since no single community discovery algorithm is universally optimal (Peel et al., 2017), we consider a collection of communities identified from several well-known community discovery methods: Greedy (Clauset et al., 2004), Walktrap (Pons and Latapy, 2006), Infomap (Rosvall and Bergstrom, 2007b), CESNA (Yang et al., 2013), and ZCmod (Zhang and Chen, 2020).

The community detection methods considered for this study carry different objectives and capture a broad range of communities. Greedy, Walktrap, and Infomap are designed for homogeneous networks, so node types on $\mathcal{V}$ are ignored when applying these methods. By comparison, CESNA and ZCmod are both "node type aware," each accounting for node type with different assumptions and constraints about the type composition of the community structure. A summary of each method is provided in Table 3.2 and a detailed account of each method is provided in Appendix B.1.

For our purposes, clusters need to be reasonably small to allow for further investigation by practitioners. Choobdar et al. (2019) recommends at least three but no more than 100 nodes per

Table 3.2: Clustering and subclustering methods considered in this study. The complexity and method objectives are provided, where n denotes the number nodes and m denotes the number of edges. Methods designed to account for differences in nodes are deemed "node type aware" (NTA). References and further details are provided in the descriptions of Appendix B.1.

Class	NTA	Method	Complexity	Objective
Clustering		Greedy	$\mathcal{O}(n \log^2 n)$	Modularity
		Walktrap	$\mathcal{O}(n^2 \log n)$	Distance based on random walks
		Infomap	$\mathcal{O}(m)$	Map equation
	✓	CESNA	$\mathcal{O}(m+n)$	Latent space likelihood
	✓	ZCmod	$\mathcal{O}(m^2)$	Heterogeneous modularity
Subclustering		Paris	$\mathcal{O}(m)$	Distance based on node pair sampling

cluster when attempting to identify disease-relevant modules in a variety of gene and protein networks. Considering the size of $\mathcal{G}$ in our application, the clustering methods used are largely network partitioning methods—the most scalable of network clustering methods. However, network partitioning methods often fail to find reasonably small communities in large networks (Fortunato and Barthelemy, 2007; Leskovec et al., 2010). Clusters in real networks often exhibit hierarchical structure, such that larger clusters are composed of smaller clusters which can be further divided. Leveraging this fact, we use an agglomerative, hierarchical clustering method called Paris (Bonald et al., 2018) to subdivide clusters attained from the clustering methods in Table 3.2 into "subclusters" meeting specified size constraints and retain only those subclusters containing both genes and phenotypes. These H heterogeneous subclusters of $\mathcal{G}$ are denoted $\boldsymbol{\mathcal{C}} = \{\mathcal{C}_h\}_{h \in [H]}$.

Formally, the Discover step first identifies a collection of clusters from $\mathcal{G}$ using a variety of network clustering methods (Table 3.2). The resulting collections of clusters are denoted $\tilde{\boldsymbol{\mathcal{C}}} = \{\tilde{\boldsymbol{\mathcal{C}}}(m)\}_{m \in M}$, where M denotes the set of clustering methods considered and $\tilde{\boldsymbol{\mathcal{C}}}(m)$ denotes the collection of clusters identified from $\mathcal{G}$ with method m. Using Paris, each cluster $\tilde{\mathcal{C}}(m) \in \tilde{\boldsymbol{\mathcal{C}}}$ is subdivided into densely connected "subclusters" with no more than 100 nodes each. We let $\boldsymbol{\mathcal{C}} = \{\mathcal{C}_h\}_{h \in [H]}$ denote the collection of H subclusters (henceforth referred to as clusters) of $\mathcal{G}$ containing both genes and phenotypes.

Score Sets' Undiscovered Relations

Provided a biological network $\mathcal{G}$, a set of nodes $\mathcal{B}_w$, and a set of newly discovered relations $\mathcal{R}$, the Score operation counts the number of undiscovered relations between nodes in $\mathcal{B}_w$, denoted ν_w, and quantifies the rarity of ν_w with a metric called the *potential for future discovery*, denoted π_w. In particular, π_w compares ν_w to what is expected from random, dense subgraphs of equal size. To proceed, we formally define ν_w, introduce a null model based on snowball sampling, and define π_w—the result of the Score operation.

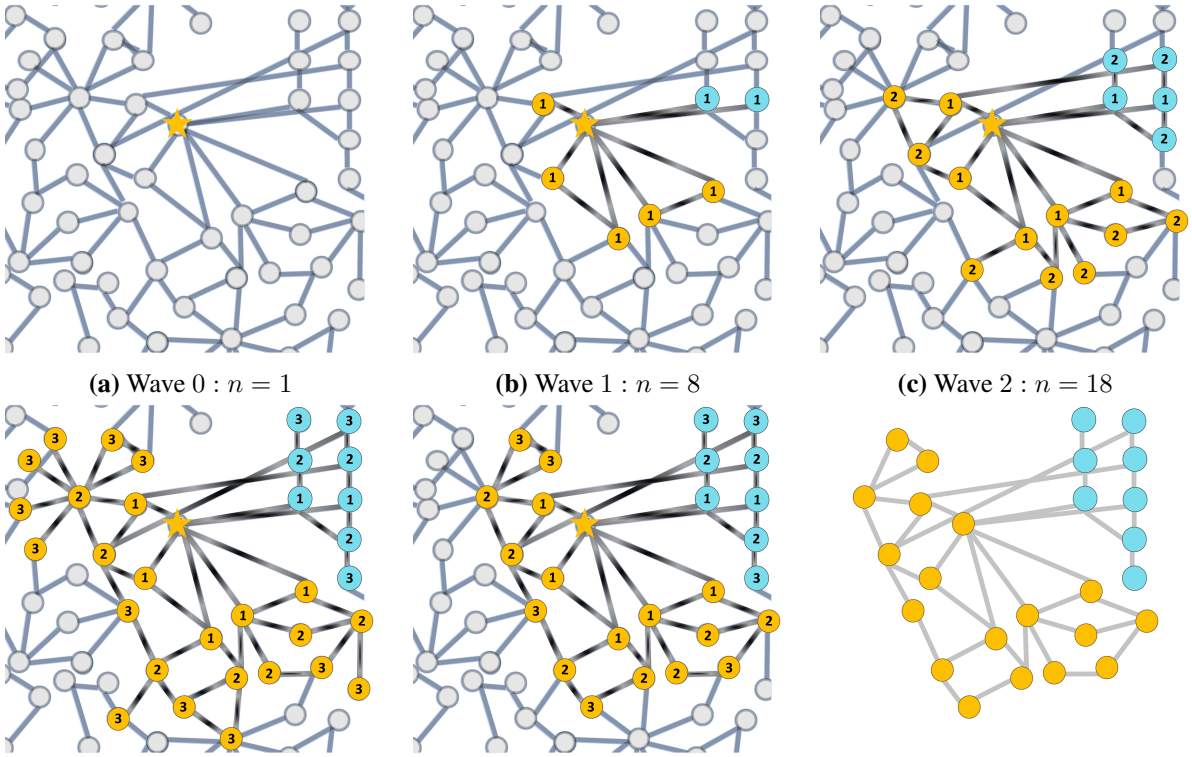
Undiscovered Relations Between Nodes in $\mathcal{B}_w$ The number of undiscovered relations between nodes in $\mathcal{B}_w$ is denoted ν_w, where

$$\nu_w = |\{\{u, v\} \in \mathcal{R} : u, v \in \mathcal{B}_w\}|. \tag{3.1}$$

The set of undiscovered relations, $\mathcal{R}$, are provided as dashed edges in the mockup in Figure 3.1; hence, the corresponding subgraph in Figure 3.1b would have a corresponding $\nu = 1$ to reflect the single undiscovered relation. To assess the rarity of such an observation given the size of the subgraph, we introduce a null model based on snowball sampling.

Snowball Sampling Null Model When interested in the number of undiscovered relations between nodes in $\mathcal{B}_w$, an appropriate null model should account for both the number of nodes in $\mathcal{B}_w$ and the relative density of $\mathcal{B}_w$. For example, suppose $\mathcal{B}_w$ is the set of 26 densely connected nodes provided in Figure 3.1b. To assess whether one undiscovered relation is noteworthy, it would be inadequate to compare against what might be expected from a set of 100 nodes, since more nodes harbor more opportunity for connection. Furthermore, it would be inadequate to compare against a set of nodes which are topologically distant from one another or otherwise unconnected, since unconnected nodes of a well-connected module tend to have a higher likelihood of being functionally related in biological networks (Gillis and Pavlidis, 2012; Wolfe et al., 2005). Therefore, we pro-

pose gauging the relative extremity of ν_w by comparing ν_w to a reference distribution constructed over the finite collection of snowballed subgraphs of $\mathcal{G}$.



(a) Wave 0 : $n = 1$ **(b)** Wave 1 : $n = 8$ **(c)** Wave 2 : $n = 18$

(d) Wave 3 : $n = 33$ **(e)** Paring down : $n = 26$ **(f)** Snowballed subgraph

Figure 3.3: Snowball sampling a subgraph of size $z_w := |\mathcal{B}_w| = 26$ starting at a randomly selected node, starred in panel (a). Waves of neighbors are selected until the node count exceeds 26. Following, nodes are removed uniformly at random until a subgraph of size z_w is attained. Panels (b) through (d) demonstrate three waves of selecting neighbors followed by removing seven nodes selected uniformly at random in panel (e). Nodes are labeled according to the wave they were added. Provided in panel (f), the snowballed subgraph is dense and maintains 26 nodes, as desired.

Traditionally used to recruit and study hidden populations (Parker et al., 2019), *snowball sampling* is a mechanism for attaining subgraphs of specified size through network ties—referred to here as *snowballed subgraphs*. One can construct a random, snowballed subgraph of size z_w by picking a node uniformly at random from the network and then selecting "waves" of neighbors until a total of z_w nodes are selected. If the final wave breeds more nodes than necessary, excess nodes are removed from the final wave uniformly at random to reach a total of z_w nodes. This

process is illustrated in Figure 3.3 for $z_w = 26$, corresponding to the size of the subgraph provided in Figure 3.1b.

First, a node is selected uniformly at random from the network, illustrated by the gold star in Figure 3.3a. The neighbors of the starred node comprise the first wave of the snowball sample, each marked by a 1 in Figure 3.3b. Since there are fewer than 26 nodes, neighbors of any node selected in the first wave comprise the second wave of the snowball sample, each marked by a 2 in Figure 3.3c. Since there are still fewer than 26 nodes, a third wave is sampled, each marked by a 3 in Figure 3.3d. With 33 nodes having surpassed the target of 26, seven nodes from the third wave are selected uniformly at random and discarded as in Figure 3.3e. The final subgraph, illustrated in Figure 3.3f, is the induced subgraph from the nodes selected through snowball sampling. By construction, the subgraph is well-connected and maintains exactly 26 nodes, as desired.

Potential for Future Discovery (PFD) Metric To characterize the extremity of ν_w, we introduce a scoring metric for $\mathcal{B}_w$ called the *potential for future discovery* (PFD):

$$\pi_w := \pi(\nu_w; z_w, \mathcal{G}, \mathcal{R}) = \int_{\mathbb{N}_0} \mathbb{I}(\nu < \nu_w)\, dF(\nu; z_w, \mathcal{G}, \mathcal{R}), \tag{3.2}$$

where F is the distribution of ν over the finite collection of snowballed subgraphs of $\mathcal{G}$ and $\mathbb{N}_0$ is the set of non-negative integers. The PFD is interpreted as the proportion of snowballed subgraphs with fewer undiscovered relations between internal nodes and is comparable across different sized sets of nodes. Specifically, if $\pi_w > \pi_{w'}$, then $\mathcal{B}_w$ harbors more undiscovered relations than $\mathcal{B}_{w'}$ when compared to what is expected from respectively sized clusters. In this sense, sets with a larger PFD are arguably more valuable for practitioners trying to better understand the underlying relationship among otherwise unconnected phenotypic and genotypic information.

The distribution F is difficult to attain since the relative likelihood of subgraph being generated through snowball sampling is unknown. Hence, rather than compute (3.2) exactly, we estimate the quantity by sampling snowballed subgraphs using the generative process just described (see Figure 3.3) and computing ν for each sampled snowballed subgraph. Denote the collection of

sampled snowballed subgraphs of $\mathcal{G}$ with size z_w as $\boldsymbol{\mathcal{S}}_w = \{\mathcal{S}_b\}_{b\in[B]}$. Then, the potential for future discovery, π_w, can be estimated as

$$\tilde{\pi}_w := \tilde{\pi}(\nu_w; z_w, \mathcal{G}, \mathcal{R}) = \frac{1}{B}\sum_{b\in[B]} \mathbb{I}(\nu_b < \nu_w), \tag{3.3}$$

where ν_b is the number of undiscovered relations between nodes in $\mathcal{S}_b$.

Train a Model to Predict the PFD Metric

While the PFD is a useful metric to compare the number of undiscovered relations in different sized sets of nodes, it can only be calculated when a set of undiscovered relations $\mathcal{R}$ is given—newly discovered edges, so to speak. Provided a collection of sets of nodes $\boldsymbol{\mathcal{B}}$ and their respective PFDs, similarly denoted $\boldsymbol{\pi} = \{\pi_w\}_{w\in[W]}$, the Train operation constructs a flexible ensemble model of regression trees to relate π_w for any $\mathcal{B}_w$ to observable features of $\mathcal{B}_w$ (see Table 3.3). Features can include both node meta-data and meta-data on $\mathcal{B}_w$.

In what follows, we discuss the features and describe the chosen model. We then discuss the training regime and optimal model specification in terms of cross-validated loss. Once specified, the model is used to predict the PFD for a set of nodes when the PFD for such set is unobservable.

Features In many networks, members of a community have both structural and functional similarities (Clauset et al., 2004), which vary drastically between communities. Here, we attempt to predict a set of node's potential for future discovery using structural and functional properties of the set. All features considered are provided in Table 3.3 and include both biologically inspired metrics (e.g., disease and tissue specificity) and network topology metrics (e.g., Newman-Girvan modularity and hub dominance).

Table 3.3: Features used as input to a DART model trained to predict the PFD for $\mathcal{B}_w$. The features describe both biological (Bio) and network topological (Net) aspects of $\mathcal{B}_w$. A brief description for each is provided, along with a range of values and the mean absolute SHAP value (Lundberg and Lee, 2017) for variable importance. Larger mean absolute SHAP values indicate the feature has relatively more influence on the model predictions. The range and mean absolute SHAP value correspond to when the pipeline is fit to $(\mathcal{G}_{19}, \mathcal{G}_{20})$ in Section 3.4.1.

Type	Feature	Description	Range	SHAP
Bio	Diseases (D)	Number of diseases for which cluster genes are enriched.	0 - 33	0.12
	Sig. GO	Number of significant GO enrichment terms after a Bonferroni correction (Gene Ontology Consortium, 2019; Mi et al., 2019).	0 - 1358	0.08
	CT specificity	Proportion of genes participating in cell type enrichment.	0.00 - 1.00	0.06
	D specificity	Proportion of genes participating in disease enrichment.	0.00 - 1.00	0.23
	PLoF[a]	Number of predicted loss of function (PLoF) variants associated with a gene in the cluster (Karczewski et al., 2020).	0 - 4440	0.06
	Gene ratio	Number of genes in a cluster divided by cluster size.	0.01 - 0.99	0.40
Net	Size	Number of nodes in cluster.	3 - 100	0.24
	TPR	Triangle participation ratio, defined as the fraction of community nodes that belong to a triad.	0.00 - 1.00	0.14
	AID	Average internal degree, defined as the average degree of all nodes in cluster.	1.33 - 58.33	0.05
	HubD	Hub dominance, defined as the ratio of the internal degree of the cluster's most connected node with respect to the theoretically maximal degree within the community	0.10 - 1.00	0.39
	AE	Average embeddedness of all nodes in cluster where the embeddedness of a node is its internal degree with respect to its overall degree.	0.00 - 0.96	0.15
	Edges inside	Number of edges internal to the community.	2 - 2711	0.03
	Surprise	Quality metric assuming that edges between nodes emerge randomly according to a hypergeometric distribution (Traag et al., 2015).	1.85 - 9.04[b]	0.02
	IED	Internal edge density, defined as the number of internal edges divided by the number of possible internal edges.	0.02 - 0.96	0.09
	Significance	Estimates how likely a partition of dense communities appear assuming a random graph (Traag et al., 2015).	3.33 - 10.53[b]	0.03

Expansion	Number of edges per community node that point outside the cluster.	-0.92 - 7.90[b]	0.00
Cut	Defined as the fraction of existing edges leaving the community.	0.00 - 0.08	0.04
Conductance	Time required for a random walk on the cluster to achieve its stationary distribution.	0.17 - 1.00	0.08
NG modularity	The number of internal edges minus the expected number of internal edges where edges are lain randomly to preserve the degree of each node (Newman, 2006).	0.99 - 1.00[c]	0.06

[a] Summarized with a max, median, mean, standard deviation, and sum of values. The sum yields the largest mean absolute SHAP value of 0.058, others can be found in Figure B.2.
[b] Log transformation applied.
[c] Exponential transformation applied.

Many of the biologically inspired features are derived from a Gene Ontology (GO) term enrichment analysis (Ashburner et al., 2000; Gene Ontology Consortium, 2019). GO enrichment is a popular technique in bioinformatics, allowing researchers to profile a set of genes (i.e., those in $\mathcal{B}_w$) based on the functional characteristics of each gene. In particular, a set of genes are assigned (possibly multiple) GO terms to describe the molecular function, biological process, and cellular component of the set—including terms like estrogen binding, protein mannosylation, and postsynapse, respectively. Using PANTHER (Mi et al., 2019), we conduct the enrichment tests on the subset of genes in $\mathcal{B}_w$ using the Gene Ontology classification system. We retain the number of significant GO terms, the number of tissues and diseases associated with the gene set, and the proportion of the gene set participating in enrichment. The number of predicted loss of function (PLoF) variants associated with each gene in $\mathcal{B}_w$ is computed with gnomAD (Karczewski et al., 2020).

Network topologically inspired features are meant to characterize the edge structure of $\mathcal{B}_w$ relative to the rest of the network, and discriminate sets in $\boldsymbol{\mathcal{B}}$. Some features are relatively simple, including the number of genes and phenotypes, total number of edges, and average number of connections in $\mathcal{B}_w$. Others are more specialized and measure things like how well-knit or cohesive $\mathcal{B}_w$ is (e.g., average embeddedness, conductance, and triangle participation ratio), the degree of hub-and-spoke topology in $\mathcal{B}_w$ (e.g., hub dominance), and the propensity for nodes in $\mathcal{B}_w$ to connect

to external nodes (e.g., expansion, cut, and normalized cut). Measures of topology are computed using Python (Van Rossum and Drake, 2009) packages `NetworkX` (Hagberg et al., 2008) and `CDlib` (Rossetti et al., 2019).

Model Description We train a multiple additive regression trees with dropout (DART) model (Vinayak and Gilad-Bachrach, 2015) to model $\mathcal{B}_w$'s potential for future discovery, denoted $\tilde{\pi}_w$, as a function of the observable features of $\mathcal{B}_w$, denoted $\mathbf{x}_w^\intercal$. Implemented in the R (R Core Team, 2022) package `XGBoost` (Chen and Guestrin, 2016) for scalability, the DART model is a generalization of a multiple additive regression trees (MART) model (Friedman and Meulman, 2003) with an added "dropout" rate parameter to overcome issues of model over-specialization. The dropout rate controls the rate at which trees are dropped or "muted" during boosting and is shown to reduce over-specialization and overfitting.

In general, tree based models require no parametric assumptions, allowing for complex relationships among the features and the response. Furthermore, boosted tree models are robust to collinearity (Chen et al., 2018). They are widely used, and when compared to other ensemble methods, provide exemplary results for logistic objectives (Carreras and Marquez, 2001; Li, 2012; Opitz and Maclin, 1999).

Model Training and Fit While incredibly flexible, DART models (like other gradient boosting models) include a host of hyperparameters that ought to be tuned to minimize loss and avoid a suboptimal fit. For a comprehensive list of hyperparameters, see Chen et al. (2022). The hyperparameters we tune are provided in Table 3.4 along with the description, range, and calculated optimal value for each according to the cross-validated loss specified.

To decide the optimal value of each hyperparameter, we consider a grid of 100 parameter combinations. To reasonably cover the parameter space with 100 points (i.e., parameter combinations), we use the principle of maximum entropy and maximize the entropy of the distribution of

Table 3.4: Hyperparameters to the DART model used to estimate each cluster's PFD. A description and range of values for each parameter is provided along with the optimal values attained via a five-fold cross-validation. The "optimal" values minimize the cross-validated RMSE when the pipeline is fitted to $(\mathcal{G}_{19}, \mathcal{G}_{20})$ in Section 3.4.1.

Parameter	Alias	Description	Range	Optimal
learning rate	`eta`	Shrinkage factor for the feature weights; used to prevent overfitting	0 - 0.1	0.043
minimum loss reduction	`gamma`	Minimum reduction in loss to continue partitioning a leaf node on a tree	0 - 30	< 0.001
maximum number of trees	`nrounds`	Maximum number of boosting iterations to consider	2 - 2000	141
maximum tree depth	`max_depth`	Maximum depth of a tree	1 - 15	12
subsample ratio	`subsample`	Subsample ratio of the training instances	0.1 - 1	0.38
dropout rate	`rate_drop`	Fraction of previous trees to drop during dropout	0 - 1	0.11
skip rate	`skip_drop`	Probability of skipping the dropout procedure during a boosting iteration	0 - 1	0.86

points. Implemented in the R package `dials` (Kuhn and Frick, 2022), we choose the 100 points such that the determinant of the spatial correlation matrix between points is maximized—shown equivalent to maximizing the entropy of the distribution of points in Johnson et al. (1990). Using five-fold cross-validation, we compute the average root mean squared error (RMSE) for each parameter combination. In a given fold, if the RMSE does not improve for 50 boosted iterations, then training will stop. The parameter combination resulting in the smallest, cross-validated average RMSE is deemed optimal.

Once the optimal parameter combination is identified, the model is trained on the entire matrix of features $\mathbf{X}^\intercal$ such that the w^{th} row of $\mathbf{x}_w^\intercal$. The optimal DART model, f, takes a vector of cluster features of $\mathcal{B}_*$, denoted $\mathbf{x}_*^\intercal$, as input and outputs $\mathcal{B}_*$'s predicted potential for future discovery, denoted $\hat{\pi}_* \in (0, 1)$; that is, $f : \mathbf{x}_*^\intercal \to (0, 1)$. In full generality, one could train a variety of models to predict a number of scoring metrics using a plethora of different features. With our application

in mind, we focus on training a flexible model to predict the potential for future discovery from the set of features provided in Table 3.3.

Rank Sets According to Model Predictions

Provided a collection of sets of nodes $\boldsymbol{\mathcal{B}} = \{\mathcal{B}_w\}_{w\in[W]}$ and a model $f : \mathbf{x}_w^\intercal \rightarrow (0, 1)$ where $\mathbf{x}_w^\intercal$ is a row vector of features of $\mathcal{B}_w$ listed in Table 3.3, the Rank operation ranks the collection of sets $\boldsymbol{\mathcal{B}}$ according to the collection of predictions $\hat{\boldsymbol{\pi}} = \{\hat{\pi}_w\}_{w\in[W]}$ where $\hat{\pi}_w = f(\mathbf{x}_w^\intercal)$ for $w \in [W]$.

To this end, we first compute the features of $\mathcal{B}_w$ listed in Table 3.3 for $w \in [W]$. Following, we compute the set of predicted PFDs: $\hat{\boldsymbol{\pi}} = \{\hat{\pi}_w\}_{w\in[W]}$. Let $w(i)$ be the rank index of the i^{th} largest value of $\hat{\boldsymbol{\pi}}$, breaking possible ties uniformly at random. Then, the collection of ranked sets is denoted $\boldsymbol{\mathcal{B}}_{\text{ord}} = \{\mathcal{B}_{w(i)}\}_{i\in[W]}$, where $\mathcal{B}_{w(i)}$ is the set of nodes with the i^{th} largest predicted PFD.

3.3.2 ICCUR Pipeline

Provided two ordered snapshots of a temporal, biological network, $\mathcal{G}_{t-1}$ and $\mathcal{G}_t$, the ICCUR pipeline returns a collection of densely connected clusters of $\mathcal{G}_t$, ranked according to their potential for future gene to phenotype relations using insights drawn from the identified clusters of $\mathcal{G}_{t-1}$ (see Algorithm 3.1). To identify and rank clusters at time t according to their potential for future gene to phenotype relations, we first identify a collection of dense, heterogeneous clusters from $\mathcal{G}_{t-1}$, denoted $\boldsymbol{\mathcal{C}}^{t-1} = \{\mathcal{C}_h^{t-1}\}_{h\in[H]}$, using **Discover** on $\mathcal{G}_{t-1}$.

We then characterize the degree to which each cluster, $\mathcal{C}_h^{t-1}$, harbors gene to phenotype relations subsequently introduced at time t. The collection of newly discovered G2P relations is given by the set difference between the G2P relations at time t and the G2P relations at time $t-1$: $E_t^{[12]} - E_{t-1}^{[12]}$. Using **Score** on $\mathcal{G}_{t-1}$, $\mathcal{C}_h^{t-1}$, and $E_t^{[12]} - E_{t-1}^{[12]}$ for all $h \in [H]$, we calculate each cluster's potential for future discovery, denoted $\tilde{\pi}_h^{t-1}$. If $\tilde{\pi}_h^{t-1} > \tilde{\pi}_{h'}^{t-1}$ for $h, h' \in [H]$, then $\mathcal{C}_h^{t-1}$ contains more newly discovered gene to phenotype relations than $\mathcal{C}_{h'}^{t-1}$ compared to what is expected from respectively sized clusters. Denote the collection of clusters' estimated PFD as $\tilde{\boldsymbol{\pi}}^{t-1} = \{\tilde{\pi}_h^{t-1}\}_{h\in[H]}$.

While the PFD is a useful way for practitioners to compare clusters, it cannot be calculated for clusters at time t if $t+1$ has yet to occur. Hence, using **Train** on $\boldsymbol{\mathcal{C}}^{t-1}$ and $\tilde{\boldsymbol{\pi}}^{t-1}$, we fit a flexible regression model to relate a cluster's potential for future discovery to observable cluster features. The resulting model is denoted as f^{t-1} to emphasize that it is trained on the potential for future discovery and features of clusters identified from $\mathcal{G}_{t-1}$, for which both quantities are observable.

At this point, we have used operations from Section **??** to better understand how clusters identified at time $t-1$ capture undiscovered gene to phenotype relations subsequently introduced at time t. To identify and rank a collection of clusters of $\mathcal{G}_t$ according to their potential for future discovery, we use **Discover** on $\mathcal{G}_t$ to identify $\boldsymbol{\mathcal{C}}^t = \{\mathcal{C}_w^t\}_{w\in[W]}$—a collection of dense, heterogeneous clusters of $\mathcal{G}_t$. Following, we use **Rank** on $\boldsymbol{\mathcal{C}}^t$ to predict the potential for future discovery for each cluster identified from $\mathcal{G}_t$ and rank them accordingly. The ranked collection of clusters is denoted $\boldsymbol{\mathcal{C}}^t_{\text{ord}} = \{\mathcal{C}^t_{w(i)}\}_{i\in[W]}$ where $\mathcal{C}^t_{w(i)}$ has the i^{th} largest predicted potential for future discovery.

3.4 Results

3.4.1 Pipeline Fitting

We fit the ICCUR pipeline to the pairs of biological networks $(\mathcal{G}_{19}, \mathcal{G}_{20})$ and $(\mathcal{G}_{20}, \mathcal{G}_{21})$. The total number of clusters identified by the Discover steps of the ICCUR pipeline is provided in Table 3.5, along with some basic information about the clusters. Aside from 2019, the majority of genes and phenotypes are assigned to a cluster each year. The clusters tend to be reasonably small, with a median size of about 22 nodes and an interquartile range of about 35 nodes each year. For each cluster, we compute the *ratio of densities* (RatD), which measures how dense a cluster is internally relative to the rest of the network. The clusters tend to have a ratio of densites of about 23 with a an IQR of about 21. For each year, the ratios of densities are much larger than one, indicating that each cluster is largely more dense internally than to the rest of the network.

When fitting the ICCUR pipeline to $(\mathcal{G}_{19}, \mathcal{G}_{20})$, clusters identified from $\mathcal{G}_{19}$ are scored according to newly discovered gene to phenotype relations subsequently introduced in 2020. Similarly, clusters identified from $\mathcal{G}_{20}$ are scored according to newly discovered gene to phenotype relations

Table 3.5: The number of clusters (N) identified by the Discover steps of the ICCUR pipeline in each year, along with the number of genes and phenotypes assigned to a cluster, the median and IQR of cluster size, and the median and IQR of cluster ratio of densities (RatD). As expected, each cluster is reasonably small and more dense internally than to the rest of the network.

		Nodes in a Cluster		Size		RatD	
Year	N	Genes (%)	Phenotypes (%)	Median	IQR	Median	IQR
2019	1474	13,810 (71)	5640 (39)	22	43	22.92	20.68
2020	3412	17,410 (89)	12,231 (80)	21	30	23.38	21.11
2021	3656	17,864 (91)	12,821 (80)	23	35	25.03	21.66

introduced in 2021 when fitting the ICCUR pipeline to $(\mathcal{G}_{20}, \mathcal{G}_{21})$. The distributions of the PFDs for 2019 and 2020 clusters are provided in Figure B.2, along with the distributions of cluster features described in Table 3.3 for 2019, 2020, and 2021 clusters. The PFD is highly varied over the range of zero and one with a low density mode near zero and a high density mode near one. The high density mode near one indicates that the clusters identified in 2019 and 2020 largely capture more undiscovered gene to phenotype relations than expected from dense, respectively sized clusters randomly attained through snowball sampling. The cluster features appear fairly stable over time, even though the identified clusters are different each year.

The hyperparameters provided in Table 3.4 minimized the average, cross-validated loss when fitting the ICCUR pipeline to both $(\mathcal{G}_{19}, \mathcal{G}_{20})$ (with an RMSE of 0.213) and $(\mathcal{G}_{20}, \mathcal{G}_{21})$ (with an RMSE of 0.191). Hence, the fitted models, f, are an ensemble of relatively few, complex trees. Each tree is complex in terms of an incredibly small minimum loss reduction and large maximal depth of each tree; however, the moderately large learning rate and relatively few boosting iterations combat the complexity of each individual tree.

When the clusters used to create the training data, i.e., $\mathbf{X}^{\intercal}$, overlap significantly with each other, it is reasonable to be concerned about independence in the training data and in the folds used for cross-validation. We demonstrate in Appendix B.2 that clusters are largely more dissimilar than similar; that is, there is little substantial pairwise overlap between the clusters used to create the training data. Furthermore, we demonstrate that the optimal model's complexity remains consistent

after controlling the pairwise overlap among the clusters. Hence, the complexity of the optimal DART model is a result of better out-of-sample performance and not a manifestation of highly dependent units present in the training and validation folds.

To better understand what each feature contributes to model predictions when fitting the ICCUR pipeline to $(\mathcal{G}_{19}, \mathcal{G}_{20})$, we leverage SHAP (SHapley Additive ExPlanations) values (Lundberg and Lee, 2017). The mean absolute SHAP value is provided in Table 3.3 as an aggregate measure of feature importance where relatively larger values indicate the feature has a relatively larger influence on model predictions. We found that gene ratio and hub dominance were most influential among the biologically and network topologically inspired features, respectively. A detailed account on SHAP values is provided in Appendix B.3, along with a summary of each cluster feature's marginal impact on model predictions.

3.4.2 Pipeline Validation

When the ICCUR pipeline is fit to $(\mathcal{G}_{19}, \mathcal{G}_{20})$, the clusters attained from $\mathcal{G}_{20}$ are ranked according to their estimated potential for future discovery using insights drawn from clusters attained from $\mathcal{G}_{19}$. Nevertheless, having observed the 2021 biological network, $\mathcal{G}_{21}$, we also have the observed potential for future discovery for each of the clusters attained from $\mathcal{G}_{20}$. With both the estimated and observed PFD for each of the 2020 clusters, we conduct an out-of-sample validation of the ICCUR pipeline. We verify that (1) the clusters identified by the ICCUR pipeline contain co-occurring yet otherwise unconnected genes and phenotypes more than expected by chance, and (2) the rankings induced by the estimated PFD are largely concordant with those induced by the observed PFD.

Clusters Contain Newly Discovered Gene to Phenotype Relations Ideally, clusters identified by the ICCUR pipeline should contain newly discovered relations between genes and phenotypes more than expected if undiscovered G2P relations were lain uniformly at random. Between 2019 and 2020, a total of 77,227 relations were added between a pair of unconnected genes and phenotypes in $\mathcal{G}_{19}$. Of the 77,227 new G2P edges, precisely 844 are internal G2P edges, meaning they

are incident to a gene and phenotype found co-occurring in a cluster identified from $\mathcal{G}_{19}$ by the ICCUR pipeline. While the fraction of new, internal edges is relatively small, we investigate the significance of 844 new, internal edges.

In total, there are approximately 280 million ways to construct a new G2P edge and approximately 196 thousand ways to construct a new, internal G2P edge in $\mathcal{G}_{19}$. If the new G2P edges were lain uniformly at random between pairs of unconnected genes and phenotypes in $\mathcal{G}_{19}$, then the random number of new, internal G2P edges would be hypergeometrically distributed for which the probability of observing at least 844 new, internal G2P edges would be less than 2.2×10^{-16}. A similar result holds for those clusters identified from $\mathcal{G}_{20}$. Hence, the clusters identified by the ICCUR pipeline contain more undiscovered G2P relations than expected by chance.

Estimated Rankings are Concordant with the Observed When the ICCUR pipeline is applied to $(\mathcal{G}_{19}, \mathcal{G}_{20})$, the estimated rankings should align with the rankings associated with the observed potential for future discovery. To verify this fact, we compute Kendall's coefficient of concordance W (Kendall, 1948), a measure of interrater reliability where $W = 0$ suggests no agreement between raters and $W = 1$ suggests perfect agreement between raters. We attain a Kendall's W of 0.83, which suggests the estimated rank largely aligns with that observed. Therefore, we conclude the DART model is a reliable judge of clusters' relative potential for future discovery. The claim of a non-zero W withstands statistical significance with $p < 2.2 \times 10^{-16}$, correcting for ties.

Notably, a collection of 870 clusters from 2020 are tied with the largest observed PFD > 0.9999. Each of these clusters harbored more new, internal G2P relations than all of the 10,000 snowball samples. To gauge the DART model's ability to rank highly the observed collection of clusters ripe for discovery, we identify the 870 clusters with the largest predicted PFD and compute the overlap between the predicted collection and the observed collection. Precisely 510 of the 870 clusters (about 59%) tied for the largest PFD were ranked in the top 870 clusters according to their predicted PFD, again providing evidence the predicted rankings provided by ICCUR are informative.

3.5 Application to Rare Disease Diagnosis

The network $\mathcal{G}_{21}$ reflects our modern understanding of biology. When the ICCUR pipeline is fit to $(\mathcal{G}_{20}, \mathcal{G}_{21})$, the clusters attained from $\mathcal{G}_{21}$ represent the "contemporary clusters" whose potential for future discovery is unobserved. To demonstrate the utility of the ICCUR pipeline as a diagnostic aid, we collect real, anonomyzed patient information from MyGene2 and identify clusters containing patients' otherwise unconnected genotypic and phenotypic information, some relations which have since been validated by human curation.

MyGene2 (Chong et al., 2016) is a web-based platform that connects persons with rare, undiagnosed diseases to clinicians or researchers working with individuals with similar phenotypic and genotypic information. MyGene2 allows users to upload and publicize patient background information, genotypic information, and phenotypic information, where the amount of information and degree of privacy is determined by the users. Some users publicly provide patient photos, background, entire gene sequence variations, and a long list of abnormal phenotypes, while other users choose to upload a single suspected genetic variant and a few abnormal phenotypes accessible only to those who have contributed to MyGene2, such as other families, clinicians or researchers. We scrape all publicly available MyGene2 profiles, 912 in total, and find 115 of these profiles contain no direct connections among the list of genetic variants and abnormal phenotypes. We posit that these are patients with an undiagnosed, rare disease.

Patient 1930, referred to here as Jane Doe, is an eleven year old female with gait ataxia, seizures, pancreatits, and weight loss, among 13 other documented phenotypic abnormalities. Jane has two variants in genes SSPO and NBEA. Despite being listed as "candidate genes" (meaning either of the variants could be responsible for her condition), there are no established relationships between any of Jane's phenotypes and her genetic variants. While her medical information is seemingly unconnected, the ICCUR pipeline identifies a cluster containing gene NBEA and phenotype HP:0001824 (weight loss), provided in Figure 3.4. The cluster identified by the ICCUR pipeline containing part of Jane Doe's medical profile may be used by practitioners and clinicians for hy-

pothesis generation, potentially establishing a relationship between Jane Doe's genetic variant to NBEA and her struggle with weight loss.

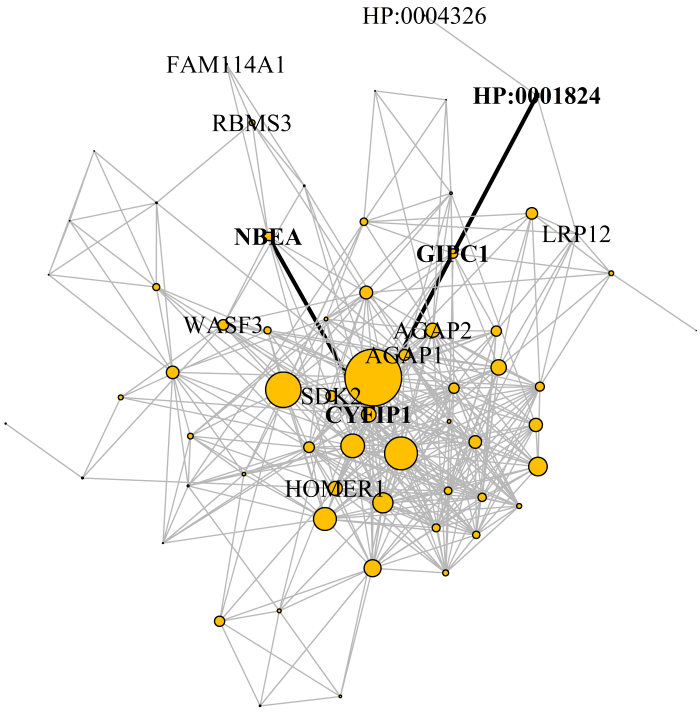


Figure 3.4: A cluster identified by the ICCUR pipeline pertaining to Jane Doe's phenotypic and genotypic information. Each node in the cluster is sized proportional to its eigenvector centrality in $\mathcal{G}_{21}$. The resulting cluster suggests a relationship between NBEA and weight loss (HP:0001824) via the shortest path (highlighted in black) of length of three: NBEA $\longleftrightarrow$ CYFIP1 $\longleftrightarrow$ GIPC1 $\longleftrightarrow$ HP:0001824. Each node along the shortest path is bolded, and all labeled nodes are direct neighbors to either NBEA or weight loss.

While generally associated with seizures, autism, and other neurological disorders (Kushima et al., 2018; Mulhern et al., 2018), NBEA has been shown to contribute to feed intake in mice and associated with body mass index in humans (Olszewski et al., 2012), despite no current direct relationship having been established in the biological ontologies for humans. Nevertheless, variants to NBEA are associated with weight loss in more patients than Jane Doe (Cantwell et al., 2021). Although a direct relationship between variants to NBEA and weight loss are not universally recognized, clinicians and researchers associated with Harvard's Undiagnosed Diseases Network (Ramoni et al., 2017) identified Jane's genetic change to NBEA as "causing the participant's symptoms," (UDN, 2022) further validating our findings.

Like Jane Doe, for each of the 115 patients with undiagnosed conditions, we test the ICCUR pipeline's ability to identify and rank clusters pertaining to these patients' otherwise unrelated phenotypic and genotypic information. For 14 MyGene2 patients, we find 17 gene to phenotype pairs co-occurring in 19 different clusters attained from ICCUR, each provided in Table 3.6 along with the clusters' predicted PFD. Except Infomap, each clustering method identified at least one relevant gene to phenotype relation. CESNA, the only overlapping method considered, identified the largest number gene to phenotype connections relevant to the MyGene2 patients. Some relevant gene to phenotype pairs were identified in multiple clusters; for example, gene SCN2A and phenotype seizures were co-occurring in four clusters identified by CESNA. Except one, all of the gene to phenotype pairs are separated by a path length of two. The only exception is that of Jane Doe and the identified relationship between NBEA and weight loss (HP:0001824). The terms are separated with a path length of three through two intermediate genes: NBEA $\longleftrightarrow$ CYFIP1 $\longleftrightarrow$ GIPC1 $\longleftrightarrow$ HP:0001824, as seen in Figure 3.4.

3.6 Discussion

We propose a novel, analytic pipeline for the identification of clusters containing undiscovered gene to phenotype relations (ICCUR) in a large, temporal heterogeneous biological network composed of genes, abnormal phenotypes, and the established relationships among them. The network is constantly evolving to reflect our most modern understanding of protein-protein interactions (via G2G relations), the level of classification in abnormal phenotypes (via P2P relations), and importantly, disease (via G2P relations). Often, patients with seemingly unrelated genetic variants and phenotypes never receive a diagnosis. ICCUR offers these patients an avenue for diagnosis by allowing practitioners to identify clusters of the network ripe for future discovery and pertaining to their patients' otherwise unconnected genetic variants and abnormal phenotypes. ICCUR is a new approach to link prediction using community structure with a mechanism to score clusters, rather than individual node pairs.

Table 3.6: Gene to phenotype pairs provided by patients' with rare, undiagnosed disease found co-occurring in a cluster identified by the ICCUR pipeline. The MyGene2 patient ID and cluster ID are provided followed by the cluster's estimated potential for future discovery.

Patient Information			Cluster Information		
ID	Gene	Phenotype	ID	Method	$\hat{\pi}$
27	SCN2A	HP:0001250 (Seizure)	15778	CESNA	0.99
27	SCN2A	HP:0001250 (Seizure)	21373	CESNA	0.99
27	SCN2A	HP:0001250 (Seizure)	24273	CESNA	0.99
27	SCN2A	HP:0001250 (Seizure)	1940	CESNA	0.82
75	BGN	HP:0010503 (Fibular duplication)	585	ZCmod	0.33
1292	TNNT3	HP:0002804 (Arthrogryposis multiplex congenita)	1321	Walktrap	0.97
1292	TNNT3	HP:0002804 (Arthrogryposis multiplex congenita)	968	Greedy	0.97
1635	SPTBN5	HP:0001250 (Seizure)	22018	CESNA	0.94
1922	FAT2	HP:0001310 (Dysmetria)	15553	CESNA	0.95
1930	NBEA	HP:0001824 (Weight loss)	84	Walktrap	0.54
1932	HUWE1	HP:0000739 (Anxiety)	9175	CESNA	0.99
1943	SHANK2	HP:0011968 (Feeding difficulties)	3199	CESNA	0.98
1948	CHD2	HP:0001250 (Seizure)	9175	CESNA	0.99
1950	SATB2	HP:0001999 (Abnormal facial shape)	14105	CESNA	0.99
1950	SATB2	HP:0000202 (Oral cleft)	5816	CESNA	0.95
2197	C17orf62	HP:0006532 (Recurrent pneumonia)	6379	CESNA	0.99
2197	C17orf62	HP:0001744 (Splenomegaly)	6379	CESNA	0.99
2234	DROSHA	HP:0000252 (Microcephaly)	24074	CESNA	0.98
2527	TRAPPC5	HP:0000011 (Neurogenic) bladder	745	Walktrap	0.80
2527	TRAPPC5	HP:0000011 (Neurogenic) bladder	420	Greedy	0.80
2584	PYROXD1	HP:0003715 (Myofibrillar myopathy)	24123	CESNA	0.99
2584	PYROXD1	HP:0003198 (Myopathy)	24123	CESNA	0.99

The novelty of the ICCUR pipeline is its ability to rank and prioritize clusters for practitioners according to their potential for future discovery using insights drawn from the identified clusters of the previous year's biological network. Using a generalization of a multiple additive regression trees model, the ICCUR pipeline is trained to predict a cluster's potential for future discovery using biologically and network topologically inspired features about the clusters, nodes, and edges. Using these estimates, ICCUR ranks clusters accordingly. The ICCUR pipeline is shown to iden-

tify clusters containing future G2P relations more than expected by chance. Furthermore, using principles of cross-validation, predicted rankings align with the rankings associated with observed potential for future discovery.

When the results of the ICCUR pipeline are applied to real, anonymized patient information, the ICCUR pipeline offers meaningful clusters for 14 of 115 patients with rare, undiagnosed conditions. The resulting clusters can be used for hypothesis generation and further investigation by clinicians to aid in diagnosis and management. These clusters have been demonstrated useful and validated by research in at least one case, that of Jane Doe (patient 1930). While we have demonstrated the utility of ICCUR for clinical diagnosis, the ranked clusters of the modern biological network can also act as a scope with which bioinformatic researchers can direct resources and funding for future studies and investigations.

3.6.1 Strengths, Limitations, and Future Directions

The ICCUR pipeline is a scalable solution for the identification of small, dense heterogeneous clusters in very large, heterogeneous networks. The resulting ranked clusters are stored and accessed using the developmental software available on GitHub (Martin, 2022). The application allows practitioners to access, filter, and sort clusters by the number of internal genes and phenotypes matching their patient's medical profile and/or the cluster's potential for future discovery. The application also visualizes other biologically useful information pertaining to the clusters, such as the internal genes' GO tissue enrichment and predicted loss of function.

A computationally burdensome step in the ICCUR pipeline involves the approximation of clusters' potential for future discovery. If the likelihood of a subgraph being generated through snowball sampling were known, then we could analytically compute the PFD in (3.2) and avoid the computationally burdensome approximation in (3.3). Hence, it stands to formally characterize the distribution of subgraphs generated through snowball sampling, an area of future work. Furthermore, since the clusters identified from the network are heterogeneous, a more appropriate null model in this context would control for the number of genes and the number of phenotypes in the

cluster, rather than simply the number of genes and phenotypes. Another area of future work is to generalize the ICCUR pipeline to either compute or approximate the PFD in (3.2) where F is the distribution of ν over the finite collection of snowballed subgraphs with prespecified number of nodes of each node type.

Finally, while there has been notable progress in the development of methodology to identify communities (i.e., clusters) from homogeneous networks, relatively few methods exist for a more general heterogeneous network. Existing methods which take into account differences between nodes and their interconnections tend to make assumptions about the community structure of heterogeneous networks. For example, CESNA (ZCmod) assumes communities in a heterogeneous network should have relatively similar (different) node types. An area of future work is the development of a community discovery method which accounts for existing differences in the connectivity patterns between pairs of node types without making assumptions or imposing constraints on the type composition of the resulting community structure.

Chapter 4

ECoHeN: A Hypothesis Testing Framework for Extracting Communities from Heterogeneous Networks

4.1 Introduction

Complex phenomena, from biological systems (Nacu et al., 2007) to world trade patterns (García-Algarra et al., 2019), are often modeled as networks (graphs) which consist of entities (nodes), connections between them (edges), and known covariates about the entities and the connections (node and edge attributes). Many disciplines have devoted significant efforts to the analysis and application of network models including statistics, physics, computer science, biology, and the social sciences. One focus that has garnished much attention is *community discovery*: the general process of assigning nodes to collections whose members are densely connected within the collection yet sparsely connected to the rest of the network, calling these relatively dense subsets of nodes *communities* (Girvan and Newman, 2002). After community members were shown to embody structural and functional similarities (Newman and Girvan, 2004), a flurry of algorithms designed to partition a graph to create this disparity in connectedness within and between partitions were proposed, e.g., (Radicchi et al., 2004; Wu and Huberman, 2004). The process of partitioning a network is generally referred to as *community detection*.

The rapid growth of literature in community detection prompted several comparative studies in the late 2000s (Fortunato, 2010; Lancichinetti and Fortunato, 2009) and generalizations to community detection in the 2010s (Psorakis et al., 2011; Traag and Bruggeman, 2009; Yang et al., 2013). One generalization of interest is a shift from community detection, a graph partitioning problem, to *community extraction*, a set-identification problem. Community extraction methods seek to discover communities from a network one at a time, allowing for arbitrary structure in the

rest of the network. Unlike community detection methods, community extraction methods readily identify *background nodes*, defined as nodes that are not preferentially attached to any well-defined community, and overlapping community structures, where nodes may belong to more than one community. While some extraction methods seek to optimize an extraction criterion (Wilson et al., 2017; Zhao et al., 2011), others seek to define the statistical significance of a community's connections under a global null model (Lancichinetti et al., 2011; Wilson et al., 2014), a strategy we follow here.

There has been notable progress in the development of methodology for extracting communities from homogeneous networks; however, few methods exist for a more general heterogeneous network. Formally defined in Section 4.2, a *heterogeneous network* is a network with different types of nodes. Most networks representing real systems are in fact heterogeneous (Shi et al., 2016). For example, large-scale biological systems are often represented as heterogeneous networks (Alshahrani et al., 2017; Piñero et al., 2016), where the biological entities, i.e., nodes, are distinguished by their biological function, i.e., node types. These networks comprise node types like proteins, diseases, phenotypes, and genetic variants (Callahan et al., 2020). Often extreme interest is placed on understanding fundamental relationships both *among* and *between* these biological entities. While community discovery is a common tool in this setting (Choobdar et al., 2018), few methods are designed to identify communities which are densely connected considering the node types of the community members. Thus, practitioners are forced to either ignore node type altogether, treating the network as homogeneous, or adapt standard community discovery methods to analyze subgraphs of the heterogeneous network separately. In either case, any information gleaned from differences in the rates of connectivity between nodes of different type are subsequently ignored.

Knowing, for example, that a link between a gene and a phenotype is relatively rare compared to a link between two genes or a link between two phenotypes is valuable information when determining whether a set of nodes composed of both genes and phenotypes should be deemed an assortative community. Relatively few connections between the genes and the phenotypes of this

set might be deemed substantial when considering the general propensity for nodes of these types to share a connection in the network. However, this is an unattainable conclusion if communities in a heterogeneous network are not determined according to the topology of the network as it relates to the node types of the community membership.

A disparity in connectedness related to node type is not unique to biological networks. McPherson et al. (2001) describe the tendency for kindred individuals to connect as *homophily*, where homophily can occur on categories such as gender, class, or political ideology. Conditioning, for example, on political ideology and accounting for the relative propensity for political actors to connect can provide rich information about the underlying community structure, including communities of mixed political ideology which are undetectable using contemporary methods. Methods designed to account for the heterogeneity in the node types and different connectivity patterns between pairs of node types tend to assume communities should be densely connected with (a) similar node types among the community members, e.g., (Li et al., 2018; Liu et al., 2014; Sengupta and Chen, 2015; Smith et al., 2016), or (b) dissimilar node types among the members, e.g., (Zhang and Chen, 2020). As a result, existing methods facilitate the discovery of homogeneous or heterogeneous community structure, but not both.

In this chapter, we introduce ECoHeN: an algorithm designed to **e**xtract **co**mmunities from **he**terogeneous **n**etworks. The significance of connectivity between a node and a set of nodes is measured using a p-value derived from the reference distribution under a heterogeneous degree configuration model. Using these p-values, ECoHeN extracts communities one at a time through a dynamic set of iterative updating rules which are guaranteed to converge. ECoHeN is a generalization and refinement of an extraction method ESSC which stands for **e**xtraction of **s**tatistically **s**ignificant **c**ommunities (Wilson et al., 2014). Unlike ESSC, ECoHeN accounts for existing differences in the connectivity patterns between pairs of node types to identify communities that are more densely connected than expected given the node types and connectivity of its membership. Existing community discovery methods for heterogeneous networks have treated the discovery of homogeneous and heterogeneous community structure as two separate objectives requiring sepa-

rate algorithms. In comparison, ECoHeN makes no assumption and places no constraint on the resulting type composition of each community, allowing ECoHeN to distinguish and identify both homogeneous and heterogeneous community structures that may overlap and can identify nodes that are not preferentially attached to any community.

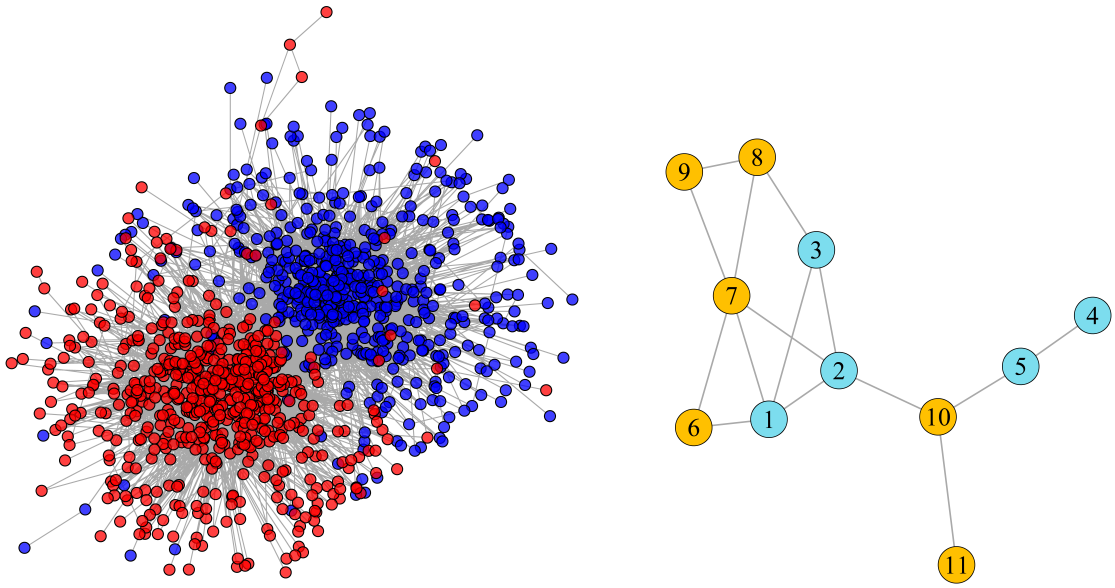
We start by formally defining a heterogeneous graph and outlining notation in Section 4.2. In Section 4.3.1, we introduce the heterogeneous degree configuration model (HDCM), a null model for studying heterogeneous networks. In Section 4.3.2, we provide the theoretical foundation for ECoHeN before outlining the algorithm in Section 4.3.3 and discussing the algorithm's parameter choices in Section 4.3.4. We illustrate the performance of ECoHeN relative to other methods in Section 4.4 before applying ECoHeN to a well-known political blogs data set in Section 4.5. Finally, we conclude with a discussion in Section 4.6.

4.2 Heterogeneous Networks

Heterogeneous networks are a special case of a more general class of networks known as *node-attributed networks*. Consistent with the terminology of Wasserman et al. (1994), node-attributed networks consist of a *structural dimension* with nodes and interactions among them, a *compositional dimension* containing the attributes (also called features or meta-data) for the nodes, and an *affiliation dimension* describing the group memberships. Since known affiliations can be expressed as node attributes, we interpret affiliation as being some latent membership to be learned through community discovery. We use the term heterogeneous network to mean a node-attributed network whose nodes are distinguished by a categorical feature called *node types*. Informally, a heterogeneous network is a node-colored network where each unique color represents a unique node type.

Heterogeneous networks are common in the social and biological sciences. For example, the political blogs network introduced by Adamic and Glance (2005) is a heterogeneous network in which political blogs, represented as nodes, are distinguished by their political ideology, represented as node types (liberals in blue and conservatives in red, as in Figure 4.1a). Used to rep-

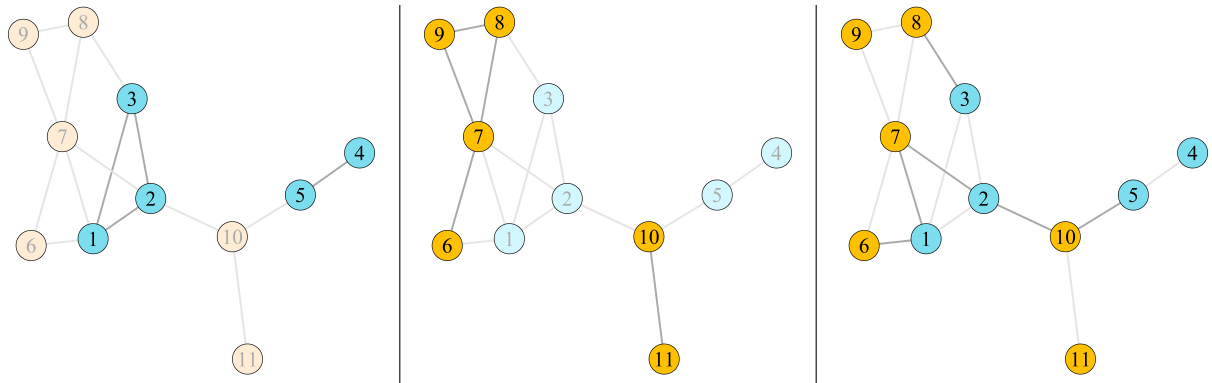
resent large-scale biological processes, knowledge graphs are another example of heterogeneous networks where biological actors, represented as nodes, are differentiated by their biological function in the network, represented as node types. Common node types in a knowledge graph include genetic variants, proteins, and phenotypes (Callahan et al., 2020). A smaller, contrived example of a heterogeneous network is provided in Figure 4.1b to help solidify notation and concepts to come. This network has two node types, depicted in blue and orange. If multiple categorical features are present, we encode each possible combination of features as a different type.



(a) Political blogs network **(b)** Toy heterogeneous network

Figure 4.1: Panel (a) is a force-directed layout of the political blogs network: a network of political blogs and the hyperlinks between them. Each blog, represented as a node, is colored according to the political ideology, i.e., its node type, where red is used to indicate conservative ideology and blue is used to indicated liberal ideology. Panel (b) is a toy example of a heterogeneous network $\mathcal{G}$ with 11 nodes and 15 edges, implying $|\mathcal{V}| = 11$ and $|\mathcal{E}| = 15$. There are five type one nodes (colored in blue) and six type two nodes (colored in orange), implying $|V^{[1]}| = 5$ and $|V^{[2]}| = 6$. The edge set consists of nine within-type edges (four among type one nodes and five among type two) and six between-type edges, implying $|E^{[11]}| = 4$, $|E^{[22]}| = 5$, and $|E^{[12]}| = 6$.

Formally, let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote an observed, undirected heterogeneous network with n nodes labeled $1, \ldots, n$ and K node types labeled $1, \ldots, K$. The node set $\mathcal{V} = \bigcup_{k=1}^{K} V^{[k]}$ where $V^{[k]}$ denotes the node set containing $|V^{[k]}|$ nodes of type k and $V^{[k]} \cap V^{[l]} = \emptyset$. When necessary, the



(a) $G^{[11]} = (V^{[1]}, E^{[11]})$ **(b)** $G^{[22]} = (V^{[2]}, E^{[22]})$ **(c)** $G^{[12]} = (V^{[1]} \cup V^{[2]}, E^{[12]})$

Figure 4.2: Decomposition of the toy network example provided in Figure 4.1b into two unipartite subgraphs ($G^{[11]}, G^{[22]}$) and a bipartite subgraph ($G^{[12]}$). Each subgraph $G^{[kl]}$ contains nodes of type k and l and the observed links between them highlighted. The degree of $3^{[1]}$, $d(3^{[1]})$, is three. The type 1 degree of $3^{[1]}$, $d^{[1]}(3^{[1]})$, is two, and the heterogeneous degree sequence of node $3^{[1]}$ is $\boldsymbol{d}(3^{[1]}) = (2, 1)$.

node type of an arbitrary node u is denoted in bracketed superscript such that $u^{[l]}$ implies that node u is of type $l \in \{1, \ldots, K\}$. Let $\mathcal{T}$ denote an n-dimensional vector of node types where the u^{th} element of $\mathcal{T}$ provides the node type of node u. We allow the heterogeneous network $\mathcal{G}$ to be a *multigraph*, meaning there could be self-loops or multi-edges; if there are no self-loops or multi-edges, the heterogeneous network is deemed *simple*. Figure 4.1b, for example, is a simple, heterogeneous network with two node types. The edge multiset of $\mathcal{G}$ can be partitioned according to adjacent nodes' types: $\mathcal{E} = \cup_{1 \leq k \leq l \leq K} E^{[kl]}$ where $E^{[kl]}$ contains the links between nodes of type k and nodes of type l. If $v^{[k]}$ is adjacent to $u^{[l]}$, then $\{v^{[k]}, u^{[l]}\} \in E^{[kl]}$. Since $\mathcal{G}$ is undirected $E^{[kl]} = E^{[lk]}$.

Partitioning the node and edge sets motivates the fact that $\mathcal{G}$ is the collection of K unipartite and $K(K-1)/2$ bipartite subgraphs. Figure 4.2 highlights the two unipartite subgraphs and one bipartite subgraph that when augmented together yield the heterogeneous network depicted in Figure 4.1b. Let $G^{[kl]} = (V^{[k]} \cup V^{[l]}, E^{[kl]})$ denote the subgraph composed of type k and type l nodes and the connections between them. If $k = l$, then $G^{[kk]}$ is a unipartite subgraph; otherwise, $G^{[kl]}$ is a bipartite subgraph. Since edges are assumed bidirectional, $G^{[kl]}$ is equal to $G^{[lk]}$.

Important for subsequent development, the *degree* of $u^{[l]}$, denoted $d(u^{[l]})$, is the number of edges incident to $u^{[l]}$. At times, it will be useful to consider the number of edges incident to $u^{[l]}$

also incident to a type k node; we refer to this quantity as the *type k degree* of $u^{[l]}$, denoted as $d^{[k]}(u^{[l]})$[1]. We let

$$\boldsymbol{d}(u^{[l]}) = \left(d^{[1]}(u^{[l]}), \ldots, d^{[K]}(u^{[l]})\right) \tag{4.1}$$

denote the *heterogeneous degree sequence* of $u^{[l]}$, providing the number of type k nodes adjacent to $u^{[l]}$ for all types $k \in \{1, \ldots, K\}$. Note that $d^{[k]}(u^{[l]})$ represents degree of $u^{[l]}$ in $G^{[kl]}$, such that $d(u^{[l]}) = \sum_k d^{[k]}(u^{[l]})$ represents the degree of $u^{[l]}$ in $\mathcal{G}$.

4.3 Methodology

We now outline a method to extract communities from heterogeneous networks which we refer to as ECoHeN. ECoHeN is designed to account for differences in connectivity patterns between nodes of various types. At its core, ECoHeN evaluates the significance of connectivity between a node and a set of nodes using a p-value derived from the reference distribution arising from a heterogeneous degree configuration model (HDCM). Using these p-values, ECoHeN extracts communities one at a time through a dynamic set of iterative updating rules which are guaranteed to converge. Identified communities are more densely connected than expected given the node types and connectivity of its membership, allowing for the discovery of both homogeneous and heterogeneous community structure. Resulting communities may overlap and exclude nodes altogether which are not preferentially attached to any community.

We start by introducing the HDCM in Section 4.3.1. In Section 4.3.2, we define a p-value for the hypothesis that an arbitrary node is well-connected to an arbitrary set of nodes and provide a theorem for the asymptotic distribution of a random variable under the HDCM which serves as the foundation for a reasonable approximation of the p-value. In Section 4.3.3, we introduce the ECoHeN algorithm and discuss initialization, extraction, and convergence. Furthermore, we

[1]Allowing for self-loops and multi-edges can complicate the interpretation and discussion surrounding $d^{[k]}(u^{[l]})$. In an undirected, simple setting, there is no distinction between counting the number of edges incident to $u^{[l]}$ also incident to a type k node (i.e., $d^{[k]}(u^{[l]})$) and counting the number of type k nodes adjacent to $u^{[l]}$, so we treat them as the same objective here. When discussing the number of type k nodes adjacent to $u^{[l]}$, we would count, for example, a type k node as twice adjacent to $u^{[l]}$ should there exist two edges between them.

discuss a procedure for paring down the set of discovered communities for practitioners. Finally, in Section 4.3.4, we discuss the algorithm's parameter choices.

4.3.1 Heterogeneous Degree Configuration Model

The degree configuration model (DCM) is a classic random network null model defined as the uniform distribution over the space of networks maintaining a given degree sequence. In the heterogeneous network setting, simply fixing the degree of each node ignores the differences among the nodes and the resulting differences in the types of connections they form. Motivated by Zhang and Chen (2020), we define the *heterogeneous degree configuration model* (HDCM) as the uniform distribution over the space of networks with node types $\mathcal{T}$ maintaining a given collection of heterogeneous degree sequences, $\boldsymbol{\mathcal{D}}$, where

$$\boldsymbol{\mathcal{D}} = \{\boldsymbol{d}(u) : u \in \mathcal{V}\} = \left\{\left(d^{[1]}(u), \ldots, d^{[K]}(u)\right) : u \in \mathcal{V}\right\}. \tag{4.2}$$

The model, denoted HDCM$(\mathcal{T}, \boldsymbol{\mathcal{D}})$, assumes all edge configurations adhering to the given collection of heterogeneous degree sequences are equally likely.

One can construct a random, heterogeneous network with node types $\mathcal{T}$ and degree collection $\boldsymbol{\mathcal{D}}$ through a generative process informally known as "stub matching." Figure 4.3 provides an illustration of this process for the toy network example in Figure 4.1b. Initially, each node type is assigned a corresponding color, e.g., node type one (two) is assigned blue (orange). Starting with n isolate nodes, each node is assigned a color according to its respective node type provided in $\mathcal{T}$, e.g., node $3^{[1]}$ is assigned blue. Colored stubs, which act as half-edges, are attached to each node according to each node's heterogeneous degree sequence $\boldsymbol{\mathcal{D}}$, e.g., node $3^{[1]}$ is assigned two blue stubs and one orange stub since $\boldsymbol{d}(3^{[1]}) = (d^{[1]}(3^{[1]}), d^{[2]}(3^{[1]})) = (2, 1)$. Finally, stubs are attached uniformly at random within constraints dictated by the color of the stubs and nodes. At each stage, a stub is selected uniformly at random from the set of available stubs. The color of the stub indicates the color of the node to which it must connect. If, for example, the first stub selected is an orange stub emanating from a blue node, then it is matched randomly with

an available blue stub emanating from an orange node. This process is repeated until no stubs remain. See Figure 4.3 for an illustration of this process. Appendix C.1 provides further discussion on the similarities and differences between the DCM and the HDCM, and an efficient means for generating a heterogeneous network from the model through a constrained permutation of the node labels in the edge multiset $\mathcal{E}$.

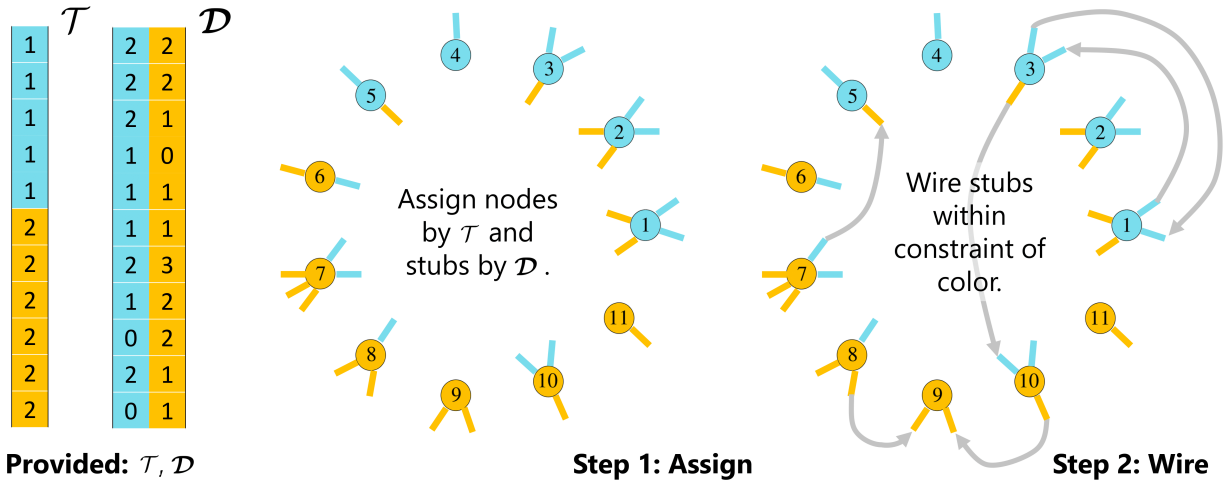


Figure 4.3: A schematic of the heterogeneous degree configuration model: a model of a random network maintaining the collection of heterogeneous degree sequences, $\mathcal{D}$. A generative form of the model begins with n isolate nodes with node type corresponding to $\mathcal{T}$. Colored stubs, which act as half-edges, are assigned according to each node's heterogeneous degree sequence and the adjacent nodes' type, provided by $\mathcal{D}$. A stub is selected where the color of the stub selected indicates the color of the node to which it must be matched. The partnering stub is selected uniformly at random from the set of available partners, and the process is repeated until no stubs remain.

4.3.2 Defining Significance of Connectivity

We evaluate the significance of the connectivity between an arbitrary node of type l, $u^{[l]} \in V^{[l]}$, and an arbitrary subset of the node set, $\mathcal{B} \subseteq \mathcal{V}$, using a p-value derived from the reference distribution under the heterogeneous degree configuration model. Since each node maintains exactly one node type, $\mathcal{B}$ can be partitioned into K subsets: $\mathcal{B} = \cup_{k=1}^{K} B^{[k]}$ where $B^{[k]}$ denotes the set of type k nodes in $\mathcal{B}$. To assess the significance of connectivity between $u^{[l]}$ and the set $\mathcal{B}$, we compare the observed number of connections between $u^{[l]}$ and nodes in $B^{[k]}$ for all $k \in \{1, \ldots, K\}$ to the

distribution under the HDCM. Let $x^{[k]}(u^{[l]} : \mathcal{B})$ denote the observed number of type k nodes in $\mathcal{B}$ (equivalently, in $B^{[k]}$) adjacent to $u^{[l]}$ for $k \in \{1, \ldots, K\}$. For comparison, let $X^{[k]}(u^{[l]} : \mathcal{B})$ represent the random variable for the number of type k nodes in $\mathcal{B}$ adjacent to $u^{[l]}$ in a random network contrived from the HDCM for $k \in \{1, \ldots, K\}$.

Under the HDCM, if the joint probability of attaining at least as many nodes in $B^{[k]}$ adjacent to $u^{[l]}$ as observed for all $k \in \{1, \ldots, K\}$ is sufficiently small, then we say $u^{[l]}$ is *well-connected* to the set $\mathcal{B}$ where the joint probability is expressed as

$$p_{\mathcal{B}}(u^{[l]}) := \mathbb{P}\left(\bigcap_{k=1}^{K} \left\{X^{[k]}(u^{[l]} : \mathcal{B}) \geq x^{[k]}(u^{[l]} : \mathcal{B})\right\}\right) \tag{4.3}$$

$$= \prod_{k=1}^{K} \mathbb{P}\left(X^{[k]}(u^{[l]} : \mathcal{B}) \geq x^{[k]}(u^{[l]} : \mathcal{B})\right). \tag{4.4}$$

Small values of $p_{\mathcal{B}}(u^{[l]})$ are unlikely under the assumption that edges were randomly constructed under the HDCM, providing evidence that $u^{[l]}$ is well-connected to $\mathcal{B}$. In this way, $p_{\mathcal{B}}(u^{[l]})$ is reminiscent of a p-value for testing the hypothesis that $u^{[l]}$ is well-connected to nodes in $\mathcal{B}$, and if $p_{\mathcal{B}}(u^{[l]})$ is less than some prespecified $\alpha \in (0, 1)$, then we conclude that $u^{[l]}$ belongs to $\mathcal{B}$. Since the connections between different pairs of node types are independent in the HDCM, (4.3) is the product of K simpler probability statements quantifying the connectivity between $u^{[l]}$ and each node type, as shown in (4.4).

To compute $p_{\mathcal{B}}(u^{[l]})$, we must characterize the distribution of $X^{[k]}(u^{[l]} : \mathcal{B})$: the random number nodes in $B^{[k]}$ adjacent to $u^{[l]}$ under the HDCM. The exact distribution for $X^{[k]}(u^{[l]} : \mathcal{B})$ is difficult to attain since enumerating every network in the sample space is infeasible even for networks on the order of 100 nodes. It is possible to approximate the distribution by sampling networks via HDCM$(\mathcal{T}, \boldsymbol{\mathcal{D}})$, but this process can be memory intensive. Instead, we analytically characterize the random generative process for the HDCM$(\mathcal{T}, \boldsymbol{\mathcal{D}})$ and provide Theorem 4.3.1, which describes the asymptotic behavior of $X^{[k]}(u^{[l]} : \mathcal{B})$ and provides the theoretical foundation for a reasonable approximation of $p_{\mathcal{B}}(u^{[l]})$.

Setup. *Let $\{\mathcal{G}_n\}_{n\geq 1}$ denote a sequence of observed, heterogeneous networks where $\mathcal{G}_n = (\mathcal{V}_n, \mathcal{E}_n)$ and $\mathcal{V}_n = \{1, \ldots, n\}$ with node types $\mathcal{T}_n$. The collection of heterogeneous degree sequences of $\mathcal{G}_n$ is denoted $\boldsymbol{\mathcal{D}}_n$. Let $\{\mathcal{H}_n\}_{n\geq 1}$ denote a sequence of random, heterogeneous networks where $\mathcal{H}_n$ is constructed via HDCM$(\mathcal{T}_n, \boldsymbol{\mathcal{D}}_n)$. Let $\{\mathcal{B}_n\}_{n\geq 1}$ denote a sequence of sets of nodes where $\mathcal{B}_n \subseteq \mathcal{V}_n$, and let $c \geq 1$ be fixed. For each $n \geq 1$, let $V_n^{[k]} = \{v \in \mathcal{V}_n : v \text{ is of type } k\}$ and $V^{[k]} \cap V^{[l]} = \emptyset$ for all $k \neq l$ such that $\mathcal{V}_n = \cup_{k=1}^K V_n^{[k]}$. Furthermore, let $B_n^{[k]} = \{v \in \mathcal{B}_n : v \in V_n^{[k]}\}$ such that $\mathcal{B}_n = \cup_{k=1}^K B_n^{[k]}$. Let $u_n^{[l]} \in V_n^{[l]}$ denote a type l node with a type k degree of c, i.e., $d^{[k]}(u_n^{[l]}) = c$, and let $F_{l,n}^{[k]}$ denote the empirical distribution of type k degrees of type l nodes in $\mathcal{V}_n$, i.e., the empirical distribution of $\{d^{[k]}(v) : v \in V^{[l]}\}$.*

Here we introduce two necessary assumptions. Assumption 4.3.1 states that the limiting proportion of type k nodes is non-zero for $k \in \{1, \ldots, K\}$. In particular, no node type vanishes as the network grows. Assumption 4.3.2 posits the existence of a limiting distribution of the type k degrees of type l nodes, for which there exists a finite, limiting mean type k degree of type l nodes for all $k, l \in \{1, \ldots, K\}$. Informally speaking, the general affinity for nodes of type k and l to connect is limiting for all $k, l \in \{1, \ldots, K\}$.

Assumption 4.3.1. *$|V_n^{[k]}|/|\mathcal{V}_n| \to \gamma^{[k]}$ as $n \to \infty$ for all $k \in \{1, \ldots, K\}$ such that $\gamma^{[k]} \in (0, 1)$ and $\sum_{k=1}^K \gamma^{[k]} = 1$.*

Assumption 4.3.2. *There exists a cumulative distribution function $F_l^{[k]}$ on $[0, \infty)$ with*

$$0 \leq \mu_l^{[k]} := \int_{\mathbb{R}^+} x \, dF_l^{[k]}(x) < \infty,$$

such that $F_{l,n}^{[k]} \xrightarrow{d} F_l^{[k]}$ and $\mu_{l,n}^{[k]} := \int_{\mathbb{R}^+} x \, dF_{l,n}^{[k]}(x) \to \mu_l^{[k]}$ as $n \to \infty$ for all $k, l \in \{1, \ldots, K\}$.

Theorem 4.3.1. *Under Assumptions 4.3.1 and 4.3.2, if $\{X_n^{[k]}(u_n^{[l]} : \mathcal{B}_n)\}_{n\geq 1}$ denotes the sequence of random variables of the number of type k nodes in $\mathcal{B}_n$ adjacent to $u_n^{[l]}$ in $\mathcal{H}_n$, then*

$d_{TV}\left(X_n^{[k]}(u_n^{[l]},\mathcal{B}_n), Y_{l,n}^{[k]}(\mathcal{B}_n)\right) \to 0$ *as* $n \to \infty$, *where* $Y_{l,n}^{[k]}(\mathcal{B}_n) \sim$ *Binom*$(c, p_{l,n}^{[k]}(\mathcal{B}_n))$ *and*

$$p_{l,n}^{[k]}(\mathcal{B}_n) = \frac{\sum_{w \in B_n^{[k]}} d^{[l]}(w)}{2^{\mathbb{I}(k=l)}|E_n^{[kl]}|}. \tag{4.5}$$

This theorem states that the total variation distance, d_{TV}, between the distribution of the number of type k nodes in a subset of the node set adjacent to an arbitrary node of type l and the distribution of a binomial random variable tends to zero as the network grows. Given this result, one can reasonably approximate the significance of connectivity between a node, $u^{[l]}$, and a set of nodes, $\mathcal{B}$, as $\prod_{k=1}^{K} \mathbb{P}\left(Y_l^{[k]}(\mathcal{B}) \geq x^{[k]}(u^{[l]} : \mathcal{B})\right)$. However, the parameter in (4.5) depends on whether $u^{[l]}$ is an element of $\mathcal{B}$, capturing the possibility of $u^{[l]}$ connecting with itself when $u^{[l]}$ is in $\mathcal{B}$. This is unsatisfactory as it impacts the convergence of ECoHeN. Therefore, in Corollary 4.3.1, we provide an alternative specification of (4.5) that disregards the possibility of $u^{[l]}$ connecting to itself. This result mirrors Theorem 4.3.1 since the density of self-loops in the HDCM tends to zero as the network becomes large.

Corollary 4.3.1. *Under Assumptions 4.3.1 and 4.3.2, if* $Y_{l,n}^{[k]}(u_n^{[l]},\mathcal{B}_n) \sim$ *Binom*$(c, p_{l,n}^{[k]}(u_n^{[l]},\mathcal{B}_n))$ *where*

$$p_{l,n}^{[k]}(u_n^{[l]},\mathcal{B}_n) = \frac{\left[\sum_{w \in B_n^{[k]}} d^{[l]}(w)\right] - \mathbb{I}(k=l)\mathbb{I}(u_n^{[l]} \in \mathcal{B}_n)c}{2^{\mathbb{I}(k=l)}|E_n^{[kl]}| - \mathbb{I}(k=l)c}, \tag{4.6}$$

then $d_{TV}\left(X_n^{[k]}(u_n^{[l]},\mathcal{B}_n), Y_{l,n}^{[k]}(u_n^{[l]},\mathcal{B}_n)\right) \to 0$ *as* $n \to \infty$.

Provided Corollary 4.3.1, one can then reasonably approximate the significance of connectivity between a node, $u^{[l]}$, and a set of nodes, $\mathcal{B}$, as $\hat{p}_{\mathcal{B}}(u^{[l]}) = \prod_{k=1}^{K} \mathbb{P}\left(Y_l^{[k]}(u^{[l]},\mathcal{B}) \geq x^{[k]}(u^{[l]} : \mathcal{B})\right)$, where $\hat{p}_{\mathcal{B}}(u^{[l]})$ is no longer dependent on whether $u^{[l]}$ is in $\mathcal{B}$, taking on the same value in either case and allowing us to characterize the convergence properties of ECoHeN in Section 4.3.3. We provide a proof of Theorem 4.3.1 and Corollary 4.3.1 in Appendix C.2.

4.3.3 ECoHeN Algorithm

The ECoHeN algorithm comprises three operations: **Initialization**, **Extraction**, and **Refinement**. At the initialization step, a collection of seed sets of nodes is chosen, where $\mathcal{B}_0$ denotes a single seed set. Following, each seed set is iteratively updated according to the set of $\{\hat{p}_{\mathcal{B}_i}(u) : u \in \mathcal{V}\}$ until no nodes are included or excluded, where $\mathcal{B}_i$ denotes the set after i updates. The resulting collection of non-empty sets of nodes represent the extracted communities and is denoted $\boldsymbol{\mathcal{C}}_T = \{\mathcal{C}_t\}_{t \in T}$, where $\mathcal{C}_t$ denotes a single extracted community. Lastly, the collection of extracted communities are refined according to a practitioner's preferences, such as desired community sizes and maximal pairwise overlap between communities. The refined communities are denoted $\boldsymbol{\mathcal{C}}_H$, where $H \subseteq T$. The full algorithm is presented in Algorithm 4.1, with supplemental extraction routines provided in Appendix C.4. We detail each component of the algorithm in the following subsections:

Initialization

By default, the collection of seed sets, denoted $\boldsymbol{\mathcal{B}_0}$, consists of the neighborhood of each node in the observed network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The neighborhood of a node $u \in \mathcal{V}$ is the set including u and all nodes adjacent to u, that is, $N(u) = \{u\} \cup \{v \in \mathcal{V} : \{u, v\} \in \mathcal{E}\}$. Hence, the collection of seed sets considered is $\boldsymbol{\mathcal{B}_0} = \{N(u) : u \in \mathcal{V}\}$. There are other ways to define seed sets; for example, locally optimal neighborhoods has been used in other works (Gleich and Seshadhri, 2012; Whang et al., 2013).

Extraction

Provided a prespecified significance level $\alpha \in (0, 1)$ and a seed set $\mathcal{B}_0 \in \boldsymbol{\mathcal{B}_0}$, the extraction procedure iteratively updates the set through a two-step, dynamic procedure until no nodes are recommended for inclusion or exclusion. We denote the set of nodes after i updates as $\mathcal{B}_i$ and the complement as $\mathcal{B}_i^c = \mathcal{V} - \mathcal{B}_i$. At each iteration, external nodes, being all $u \notin \mathcal{B}_i$, which are well-connected to $\mathcal{B}_i$ are added to $\mathcal{B}_i$. Since this involves $|\mathcal{B}_i^c|$ tests, a false discovery rate (FDR) correction (Benjamini and Hochberg, 1995) is evoked. The intermediate set containing $\mathcal{B}_i$ and

ECoHeN Algorithm

Initialization

Inputs: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.
Return: A collection of seed sets.
Let $\boldsymbol{\mathcal{B}_0} = \{N(u) : u \in \mathcal{V}\}$ where $N(u) = \{u\} \cup \{v \in \mathcal{V} : \{u, v\} \in \mathcal{E}\}$.

Extraction

Inputs: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, seed sets $\boldsymbol{\mathcal{B}_0}$, significance level $\alpha \in (0, 1)$, learning rate $\xi \in [0, 1]$, and decay rate $\phi \in [0, 1)$.
Return: A collection of communities, $\boldsymbol{\mathcal{C}_T}$.
For each $\mathcal{B}_0 \in \boldsymbol{\mathcal{B}_0}$ **do** (in parallel) **:**
Let $\mathcal{B}_1 = \mathcal{B}_0$ and $\mathcal{B}_0 = \mathcal{B}_0^+ = \emptyset$.
Let $i = 0$.
While $\mathcal{B}_i \neq \mathcal{B}_i^+$ **or** $\mathcal{B}_i^+ \neq \mathcal{B}_{i+1}$ **:**
Let $i = i + 1$ denote the current iteration.
Let $\mu = \max(1, \lfloor \xi \phi^{i-1} |\mathcal{V}| \rfloor)$ denote the maximal allowance at the current iteration.
Let $\mathcal{B}_i^+ =$ **AddWellConnected**$(\mathcal{G}, \mathcal{B}_i, \mu)$ denote the set $\mathcal{B}_i$ and any well-connected external nodes.
Let $\mathcal{B}_{i+1} =$ **RemoveLooselyConnected**$(\mathcal{G}, \mathcal{B}_i^+, \mu)$ denote the set $\mathcal{B}_i^+$ without any loosely connected internal nodes.
If $\mathcal{B}_i \neq \emptyset$ **then :**
Return $\mathcal{B}_i$.
Let $\boldsymbol{\mathcal{C}_T} = \{\mathcal{B}_i\}$ be the set of nonempty, extracted communities.

Refinement

Inputs: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, communities $\boldsymbol{\mathcal{C}_T}$, minimum size s_{min}, maximum size s_{max}, whether to remove fully connected communities f_{rm}, and maximal overlap β.
Return: A collection of communities, $\boldsymbol{\mathcal{C}_H}$, where $H \subseteq T$.
Let $\boldsymbol{\mathcal{C}_T} = \{\mathcal{C}_t \in \boldsymbol{\mathcal{C}_T} : s_{\text{min}} \leq |\mathcal{C}_t| \leq s_{\text{max}}\}$.
If f_{rm} **then :**
Let $\boldsymbol{\mathcal{C}_T} = \boldsymbol{\mathcal{C}_T} - \{\mathcal{C}_t \in \boldsymbol{\mathcal{C}_T} : \forall u, v \in \mathcal{C}_t, \{u, v\} \in \mathcal{E}\}$.
Filter $\boldsymbol{\mathcal{C}_T}$ according maximal pairwise overlap **:**
Identify the largest community $\mathcal{C}_h \in \boldsymbol{\mathcal{C}_T}$, and let $H = \{h\}$.
While there exists an $h^* \notin H$ such that for all $h \in H$, $J(\mathcal{C}_{h^*}, \mathcal{C}_h) \leq \beta$ **do :**
Identify the largest community $\mathcal{C}_{h^*} \in \boldsymbol{\mathcal{C}_T}$ such that $h^* \notin H$ and $J(\mathcal{C}_{h^*}, \mathcal{C}_h) \leq \beta$, and let $H = H \cup \{h^*\}$.
Let $\boldsymbol{\mathcal{C}_H}$ be the set of refined communities.

Algorithm 4.1: Pseudocode for the ECoHeN algorithm. We include pseudocode for the extraction routines AddWellConnected and RemoveLooselyConnected in Appendix C.4.

any additions is denoted $\mathcal{B}_i^+$. Following, internal nodes, being all $u \in \mathcal{B}_i^+$, which are no longer well-connected to $\mathcal{B}_i^+$ are removed. Again, since this involves $|\mathcal{B}_i^+|$ tests, an FDR correction is evoked. The set following any removals is denoted $\mathcal{B}_{i+1}$ as this completes one iteration of the update procedure. The procedure continues in this way until no nodes are added or removed, that is, until $\mathcal{B}_i = \mathcal{B}_i^+ = \mathcal{B}_{i+1}$.

Maximal Allowance In ECoHeN's predecessor ESSC, all nodes are simultaneously considered for inclusion and exclusion at each iteration, yet this does not have to be the case. In fact, the choice of how many nodes' memberships (either in $\mathcal{B}_i$ or in $\mathcal{B}_i^c$) are updated can have major implications on whether the update procedure converges necessarily and the number of iterations required until convergence. We define the *maximal allowance* at iteration i, denoted μ_i, to be the maximum number of nodes permitted into $\mathcal{B}_i$ and the maximum number permitted out of $\mathcal{B}_i^+$ at iteration i. If $\mu_i = |\mathcal{V}|$ for all i, the update routine is relatively fast but can result in *cycles*: the infinite alternation between two sets of nodes (problematic in Wilson et al. (2014) and Bodwin et al. (2018)). If $\mu_i = 1$ for all i, the update routine converges but may take unnecessarily many iterations to converge. As such, we propose initializing the maximum allowance to a large value and then progressively decreasing it to guarantee convergence while reducing the number of iterations. We propose defining the maximal allowance using an exponential decay function parameterized by an initial learning rate, $\xi \in [0, 1]$, and a decay rate, $\phi \in [0, 1]$:

$$\mu_i = \max(1, \lfloor \xi \phi^{i-1} |\mathcal{V}| \rfloor). \tag{4.7}$$

The learning rate controls the maximal allowance on the first iteration, and the decay rate controls the maximal allowance thereafter. Together, ξ and ϕ control the scale of updates where smaller values of ξ and ϕ necessitate small, micro-level changes to $\mathcal{B}_i$ at each iteration, and larger values of ξ and ϕ allow for larger, macro-level changes to $\mathcal{B}_i$ at early iterations. The outer maximum operation in (4.7) ensures that if the number of nodes dictated by the floor function reaches zero before convergence, then one node is permitted to transition into and out of $\mathcal{B}_i$ until convergence.

Setting $\xi = \phi = 1$ ($\xi = \phi = 0$) is equivalent to setting $\mu_i = |\mathcal{V}|$ ($\mu_i = 1$) for all i; each are problematic as previously discussed.

We first characterize the convergence properties of the extraction procedure in Theorem 4.3.2 before discussing the practical implications.

Setup. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote an observed, heterogeneous network. In the extraction procedure, let $\hat{p}_{\mathcal{B}}(u^{[l]}) = \prod_{k=1}^{K} \mathbb{P}\left(Y_l^{[k]}(u^{[l]}, \mathcal{B}) \geq x^{[k]}(u^{[l]} : \mathcal{B})\right)$ for any set $\mathcal{B}$ by Corollary 4.3.1 where $u^{[l]}$ is an arbitrary node of type l and $Y_l^{[k]}(u^{[l]}, \mathcal{B}) \sim$ Binom$(d^{[k]}(u^{[l]}), p_l^{[k]}(u^{[l]}, \mathcal{B}))$.*

Theorem 4.3.2. *Let $\alpha \in (0, 1)$, $\xi \in [0, 1]$, and $\phi \in [0, 1)$ be fixed constants. Suppose $\mathcal{B}_0 \subseteq \mathcal{V}$ denotes a seed set for the extraction procedure. If there exists a j such that $|\mathcal{B}_i| < |\mathcal{V}|/2$ for all $i \geq j$, the extraction procedure will not cycle.*

Based on Theorem 4.3.2, setting $\phi < 1$ is enough to guarantee ECoHeN will never cycle between two sets, no matter the choice of ξ. While we cannot prove theoretically that ECoHeN will not alternate between more than two sets (e.g., a three-set cycle), this phenomena has not been observed empirically. When $\xi = 1$ and $\phi < 1 - \epsilon$ for small $\epsilon > 0$, the maximal allowance is relatively large for many early iterations, often prompting early convergence compared to $\xi = \phi = 0$, while simultaneously guaranteeing convergence. The impact of ξ and ϕ on the quantity and quality of communities found via simulation is documented in Section 4.3.4. We provide a proof of Theorem 4.3.2 in Appendix C.2.

Refinement

The extraction procedure identifies a collection $\boldsymbol{\mathcal{C}}_T = \{\mathcal{C}_t\}_{t \in T}$ of communities based on the collection of seed sets $\boldsymbol{\mathcal{B}}_\mathbf{0}$. In many applications, only communities with certain properties are considered desirable. For example, the DREAM challenge, described in Choobdar et al. (2018), required all communities submitted for competition be non-overlapping and contain 3-100 members. In other settings, community discovery is used as an aid for link prediction (Soundarajan and Hopcroft, 2012), in which case fully connected communities are not of interest. The refinement

process reduces the collection of discovered communities to a subset $\boldsymbol{\mathcal{C}}_H$, $H \subseteq T$, of communities satisfying the requested constraints.

While the constraints on size and connectedness are easily implemented, refining a collection of communities based on overlap requires more care. Let $\beta \in [0, 1]$ denote a user specified maximum pairwise overlap between communities based on the Jaccard similarity measure, denoted J. The refinement process begins by identifying the largest community $\mathcal{C}_h$, $h \in T$, and letting $H = \{h\}$. Following, we identify the largest community $\mathcal{C}_{h^*}$, such that $h^* \notin H$ and $J(\mathcal{C}_{h^*}, \mathcal{C}_h) \leq \beta$ for all $h \in H$, and set $H = H \cup \{h^*\}$. The process continues until no additional communities maintain the overlap constraint dictated by β, i.e., until for all $h^* \notin H$ there exists an $h \in H$ such that $J(\mathcal{C}_{h^*}, \mathcal{C}_h) > \beta$. The refinement procedure is similar to that of Wilson et al. (2017).

4.3.4 Parameter Choices

ECoHeN is implemented as an R package `ECoHeN` (Gibbs, 2022) available on GitHub with the full algorithm, including all extraction routines, provided in pseudocode in Algorithm 4.1 and Algorithm C.1 of Appendix C.4. The extraction procedure is implemented in `C++` for efficiency with parallelized extractions across initial seed sets. ECoHeN allows the user to specify the significance level α, learning rate ξ, and decay rate ϕ. A brief description of each parameter, the parameter ranges, and default recommendations are provided in Table 4.1. The significance threshold, α, controls how conservative one would like to be in defining a community; smaller values of α result in fewer, smaller communities which tend to be incredibly densely connected compared to the rest of the network. Larger values of α result in more, larger communities which are less dense.

The learning and decay rates control the maximum allowance at each iteration of the extraction procedure and should be set together. If $\xi = \phi = 0$, then the extraction procedure provides the greatest resolution in identifying a single community from background noise (see Appendix C.3); however, there are a number of downsides associated with this setting, including the extraction of many communities with substantial pairwise overlap, many required iterations before convergence, and most notably, the identification of communities in truly random networks (see Appendix C.3).

Table 4.1: Descriptions and notation for each parameter along with default suggestions. Defaults are set to speed up the extraction procedure with guaranteed convergence and minimal loss in the power to discover communities.

Parameter	Default	Description
Significance level • $\alpha \in (0, 1)$	0.10	Determines whether $\hat{p}_{\mathcal{B}}(u)$ is small (large) enough to justify the inclusion (exclusion) of node u to the set $\mathcal{B}$ for each u at each iteration of the extraction procedure.
Learning rate • $\xi \in [0, 1]$	1	Controls the maximal allowance on the first iteration of the extraction procedure; $\xi = 1$ allows for large changes to $\mathcal{B}$ on the first iteration.
Decay rate • $\phi \in [0, 1]$	0.99	Controls the maximal allowance after the first iteration of the extraction procedure; $\phi = 1 - \epsilon$ for small ϵ allows for large changes to $\mathcal{B}$ for many iterations, often reducing the number of iterations until guaranteed convergence.

When $\xi = \phi = 1$, the extraction procedure often requires fewer iterations until convergence and results in fewer communities with less pairwise overlap; however, convergence of the algorithm is not guaranteed. Notably, when $\xi = \phi = 1$, densely connected communities of interest may be unidentified as the extraction procedure may cycle between densely connected sets of nodes until a maximum number of iterations is reached and the extraction procedure is terminated.

Setting $\xi = 1$ and $\phi < 1$ guarantees that ECoHeN will converge and resolves issues with identifying communities in random networks, allowing ECoHeN to escape the low conductance seed sets; however, communities may still have substantial pairwise overlap. The problem is abated for ϕ closer to one which is also shown in Appendix C.3, whereby the algorithm is best able to identify a single community from background noise when $\xi = 1$. Based on these findings, the default choice for the learning and decay rates are $\xi = 1$ and $\phi = 0.99$, respectively. The refinement procedure further reduces the degree of pairwise overlap in the final set of identified communities when $\beta < 1$.

4.4 Simulation Study

In this section, we investigate the conditions for which ECoHeN identifies simulated community structure. We leverage an extension of the classic stochastic block model, referred to here as the heterogeneous stochastic block model (HSBM), to generate the heterogeneous networks under study. Described in Zhang and Chen (2020) for two node types and implemented in Gibbs (2022) for K node types, the HSBM is a flexible framework for generating networks with numerous node types and is detailed in Appendix C.3. In this study, we consider networks with 500 red (i.e., type one) and 500 blue (i.e., type two) nodes split between a background block and a high connectivity block (HCB). Nodes are assigned to blocks according to the parameter p where connections between nodes are determined stochastically according to parameters $\{b, r_{11}, r_{22}, r_{12}\}$. We detail these parameters in the following paragraph using Figure 4.4 to motivate their definition.

Each network under study is generated from the HSBM with parameters $\{b, r_{11}, r_{22}, r_{12}\}$. The parameter p is the proportion of nodes of each type assigned to the HCB. A larger p corresponds to a larger community size. Connections between nodes are sampled according to Bernoulli random variables with expectation dependent on the block assignment and node type of each node. The probability of connection between two nodes is b, interpreted as the *background rate*, if each node be a member of the background block or members of different blocks (i.e., one in the background and one in the HCB). The probability of connection between two nodes in the HCB is $b + r_{ij}$ where r_{ij} provides the additive increase in the probability of connection between nodes of type i and j. Figure 4.4 is constructed with parameters $p = 0.45$, $b = 0.05$, $r_{11} = 0.25$, $r_{22} = 0.20$, and $r_{12} = 0.10$.

We wish to characterize ECoHeN's ability to identify two classes of community structure: heterogeneous and homogeneous community structure. The network presented in Figure 4.4 embodies heterogeneous community structure provided $r_{ij} > 0$ for all $1 \leq i \leq j \leq 2$. In particular, when $r_{12} > 0$, nodes of different type in the HCB have a higher propensity to connect compared to the background, implying the existence of one heterogeneous community composed of both red and blue nodes. When $r_{ii} > 0$ for $i \in \{1, 2\}$ and $r_{12} = 0$, nodes of different type in the HCB con-

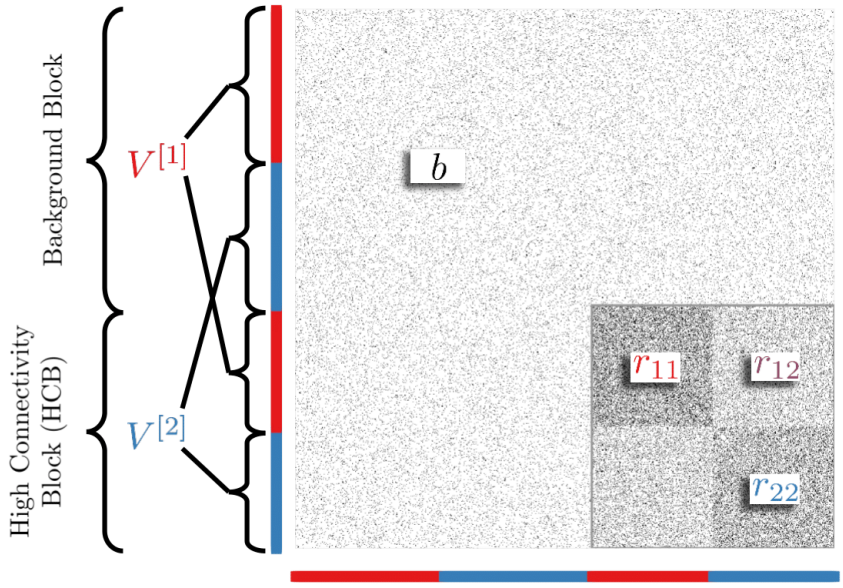


Figure 4.4: The adjacency matrix of a network generated under the HSBM with parameters $p = 0.45$, $b = 0.05, r_{11} = 0.25, r_{22} = 0.20$, and $r_{12} = 0.10$. Node type is illustrated by the colored bars adorning the axes, and links are illustrated by black dots. Nodes are assigned to blocks according to p where edges are sampled independently according to Bernoulli probabilities. Two nodes connect with probability b if each are in the background block or do not share a block. The parameter r_{ij}, $1 \leq i \leq j \leq 2$, provides the additive increase in the rate of connection between nodes of type i and j if each node is in the HCB. Connections between nodes in the HCB are bordered in a black square. When $r_{ij} > 0$ for all $1 \leq i \leq j \leq 2$, the network embodies heterogeneous community structure with one community composed of red and blue nodes. When $r_{ii} > 0$ for $i \in \{1, 2\}$ and $r_{12} = 0$, the network embodies homogeneous community structure with two communities: one composed of red nodes, and one composed of blue nodes. When $r_{ij} = 0$ for all $1 \leq i \leq j \leq 2$, the network is an ER network with no underlying community structure.

nect at the same rate as the background, implying the existence of two homogeneous communities: one composed of red nodes and one composed of blue nodes. When $r_{ij} = 0$ for all $1 \leq i \leq j \leq 2$, the network is a heterogeneous analog of an Erdős-Rényi-Gilbert (ER) network (Erdős and Rényi, 1960; Gilbert, 1959) with probability of connection b. ECoHeN's ability to assign all nodes to the background in ER networks is explored in Appendix C.3.

For each class of community structure, we consider a range of HCB sizes and network connectivity patterns. Three community discovery methods designed for homogeneous networks (i.e., ESSC (Wilson et al., 2014), Infomap (Rosvall et al., 2009), and Walktrap (Pons and Latapy, 2006)) and two community discovery methods designed for heterogeneous networks (i.e., ECoHeN and ZCmod (Zhang and Chen, 2020)) are applied. While no community discovery method is univer-

sally optimal, Infomap and Walktrap tend to partition a network into smaller, more dense modules than other canonical community detection methods (Smith et al., 2020) and outperform many competing methods for a variety of benchmarks (Javed et al., 2018). ESSC is a special case of ECoHeN when the number of node types is one, i.e., the network is homogeneous, justifying its inclusion. To our knowledge, ZCmod is the only existing community detection designed to identify heterogeneous community structure. We ignore node type when applying ESSC, Infomap, and Walktrap on the simulated networks.

One-hundred networks are generated for a given simulated condition and are referred to as replicates. For each replicate, we apply the aforementioned community discovery methods and record the respective set of discovered communities. For each method and each replicate, we compute the maximum Jaccard similarity measure between the underlying community structure and the set of discovered communities. A value near one (zero) indicates the community was near perfectly identified (not identified) by the method. ECoHeN, ESSC, and ZCmod are implemented in the `ECoHeN` package (Gibbs, 2022), whereas Infomap and Walktrap are implemented in the `igraph` package (Csardi and Nepusz, 2006) of the R programming language (R Core Team, 2022).

4.4.1 Heterogeneous Community Structure

To generate heterogeneous networks with heterogeneous community structure, we fix $b = 0.05$, and let $r_{ii} \in (0.15, 0.20, 0.25, 0.30)$ for $i \in \{1, 2\}$ and $r_{12} \in (0.025, 0.05, 0.075)$. We allow the proportion of nodes assigned to the HCB to vary according to $p \in (0.05, 0.10, 0.15, 0.20)$. Hence, we consider a total of 192 simulated conditions. Random networks under these settings have heterogeneous community structure, particularly one community composed of red nodes and blue nodes.

Figure 4.5 shows the results for identifying the heterogeneous community when $r_{11} = r_{22}$. ECoHeN and ZCmod's ability to identify the heterogeneous community notably improves as the within-HCB, within-type density (i.e., $b + r_{ii}$) increases *and* the within-HCB, between-type density

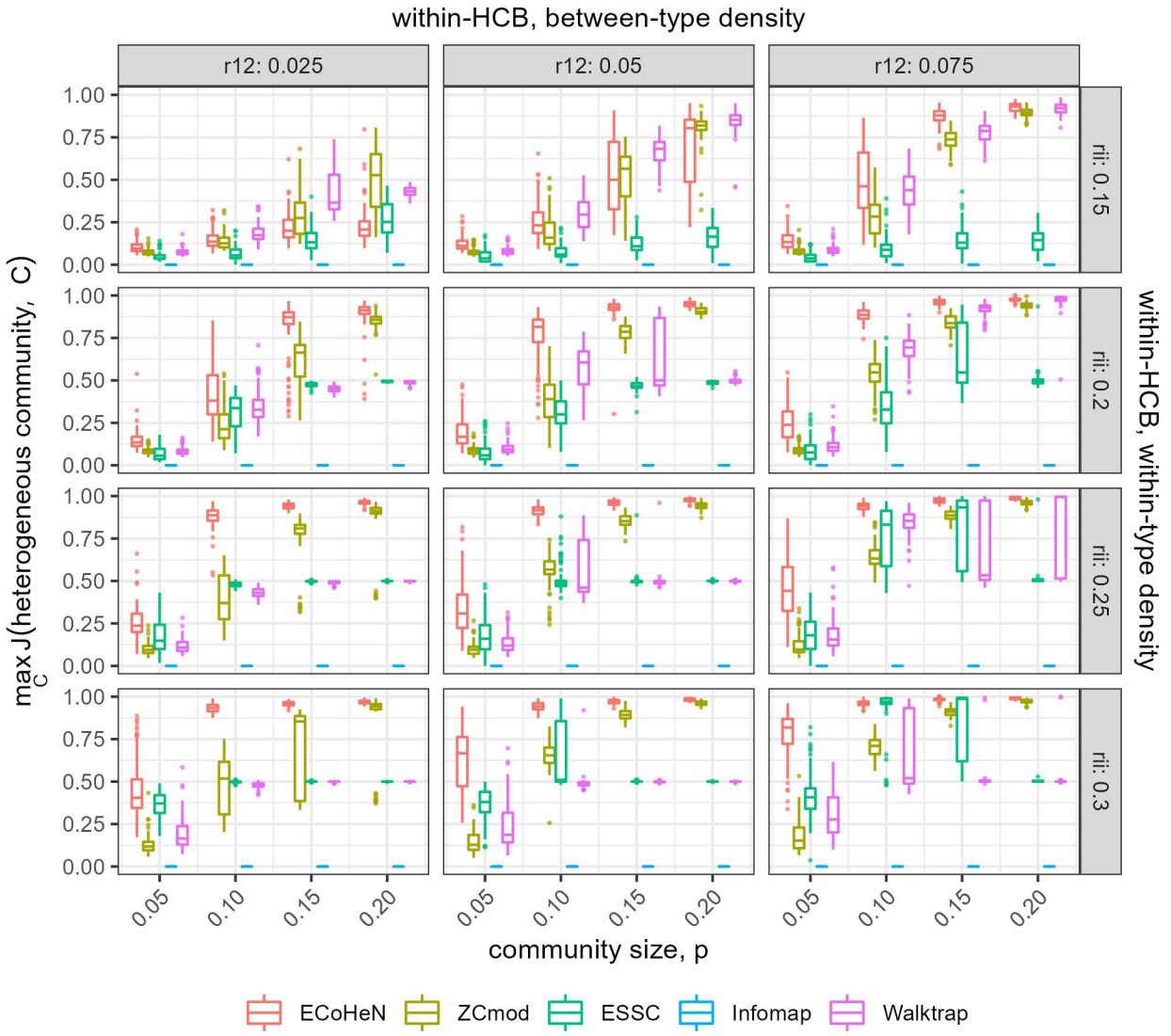


Figure 4.5: Each method's ability to identify the heterogeneous community is shown as a function of the size of the heterogeneous community. As the density of within-type and between-type links increases, ECoHeN identifies the heterogeneous community with increasingly better precision and marked improvements for small, heterogeneous communities. In nearly all simulated conditions, ECoHeN performs as well or better than ZCmod. The performance of ESSC and Walktrap is highly varied compared to ECoHeN and ZCmod, often varying between 0.5 and 1 (e.g., $r_{12} = 0.075$ and $r_{ii} = 0.25$). Unlike ECoHeN and ZCmod, these methods fail to account for node type, ergo fail to recognize the heterogeneous community for some replicates and instead identify the two homogeneous subsets of the HCB.

(i.e., $b + r_{ij}$) increases. On the other hand, ESSC and Walktrap's ability to identify the heterogeneous community improves for increasing r_{ii} and r_{12} but with more variability and less precision compared to ECoHeN and ZCmod. In general, Infomap fails to find any communities, consistently returning a trivial partition of 1000 communities each containing one node. ECoHeN performs as well or better than ZCmod at each simulated condition, consistently outperforming ZCmod at identifying small, heterogeneous communities (i.e., when p is small). Since ZCmod is a modularity optimization method, it appears to suffer from a known resolution limit characteristic of modularity optimization methods (Fortunato and Barthelemy, 2007), struggling to recover small communities with a reasonable degree of accuracy. For larger values of p, ECoHeN performs similarly to ZCmod in its ability to recover the heterogeneous community. When $r_{12} = 0.05$ and $r_{ii} = 0.15$, ECoHeN can identify larger simulated heterogeneous communities with reasonable accuracy, but there is significant variability in the results. As the density of community links increases (i.e., r_{12} or r_{ii} increases), ECoHeN demonstrates improved accuracy with less variability, reaching a natural inflection point where its performance significantly improves. A broad range of simulated conditions are considered and presented in Appendix C.3.

For many settings (e.g., $r_{12} = 0.075$ and $r_{ii} = 0.25$), the maximum Jaccard for ESSC and Walktrap will vary from 0.5 to one with the median often equal to 0.5 or one. In these settings, ESSC and Walktrap identify the heterogeneous community for some replicates (resulting in a Jaccard of one) and the homogeneous communities for other replicates (resulting in a Jaccard of 0.5). On the other hand, ECoHeN and ZCmod consistently identify the heterogeneous community in these settings, recognizing that the within-HCB, between-type density is larger than the background rate (e.g., 2.5 times larger when $r_{12} = 0.075$), albeit much less than the within-HCB, within-type density. By comparison, ESSC and Walktrap will only identify the heterogeneous community when the overall density of the HCB is sufficiently high because the between-type density is sufficiently high.

4.4.2 Homogeneous Community Structure

To generate heterogeneous networks with homogeneous community structure, we fix $b = 0.05$, and let $r_{ii} \in (0.15, 0.20, 0.25, 0.30)$ for $i \in \{1, 2\}$ and $r_{12} = 0$. Again, we allow the proportion of nodes assigned to the HCB to vary according to $p \in (0.05, 0.10, 0.15, 0.20)$. Random networks under these settings have homogeneous community structure, particularly two communities: one composed of red nodes, i.e., the red community, and one composed of blue nodes, i.e., the blue community. We start by investigating each method's ability to identify the red community when $r_{11} \in (0.20, 0.25, 0.30)$ and $r_{22} = 0.25$ for a total of 36 simulated conditions.

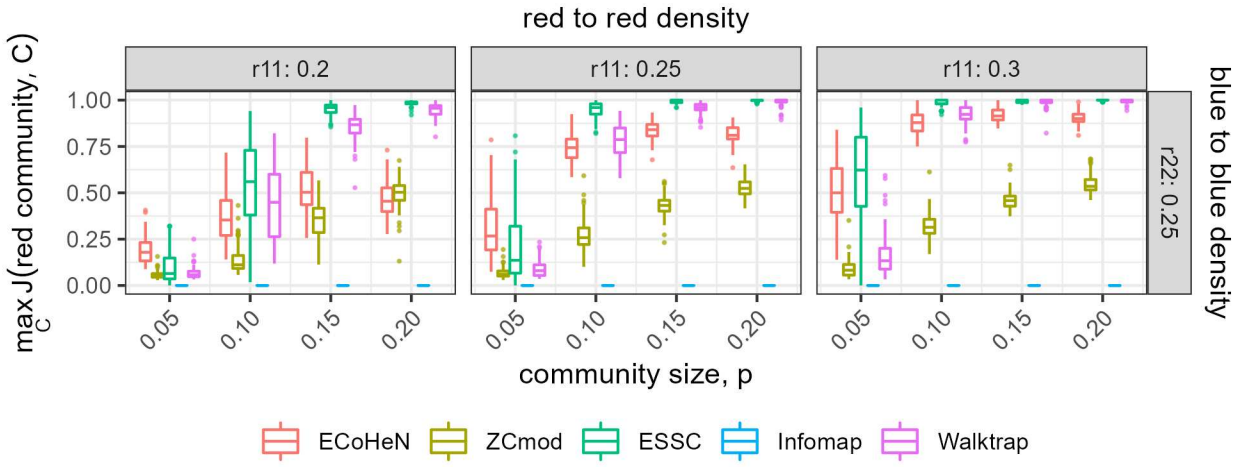


Figure 4.6: Each method's ability to identify the red, homogeneous community is shown for varying red community sizes p. As the density of red-to-red links increases, ECoHeN identifies the red community with increasingly better precision and is notably better when the red community is small. ECoHeN can recover both homogeneous and heterogeneous community structure; however, the tradeoff for this functionality is a reduction in power for identifying homogeneous communities. Nevertheless, ECoHeN's ability to recover the homogeneous community far surpasses ZCmod, which requires each community maintain at least one node of each node type.

Figure 4.6 shows the results for identifying the red community when the density of blue-to-blue links in the HCB is fixed according to $r_{22} = 0.25$. As the density of red-to-red links in the HCB, controlled by r_{11}, increases, ECoHeN can identify the red community with increasingly better precision and with marked improvements for small, homogeneous communities (i.e., when p is small). There are no such improvements from ZCmod which requires each discovered

community maintain at least one node of each type. At the same time, ESSC and Walktrap consistently outperform ECoHeN and ZCmod at identifying homogeneous community structure. This is not surprising considering these methods identify communities irrespective of node type. While ECoHeN is designed to identify both homogeneous and heterogeneous community structure, the tradeoff for this functionality is a reduction in power for identifying homogeneous communities. This is unsurprising as ECoHeN considers both same-type and between-type edges simultaneously, rather than just same-type edges, looking for evidence of excess connectivity.

Comparing methods designed for homogeneous networks, ESSC performs better or similarly to Walktrap at identifying dense, homogeneous communities. Infomap continues to return a trivial partition of 1000 communities containing one node each, incapable of identifying a community amongst background noise. The results over all simulated conditions, provided in Appendix C.3, demonstrate that the density of blue-to-blue links in the HCB, controlled by r_{22}, does not impact any method's ability to identify the red community. Furthermore, the results for identifying the blue community are provided in Appendix C.3 along with a broad range of simulated conditions.

4.5 Empirical Study

To illustrate the utility of ECoHeN in practice, we extract communities from the political blogs network of Adamic and Glance (2005). This iconic network consists of political blogs (represented as nodes) and the hyperlinks between them (represented as undirected edges). Collected shortly after the 2004 U.S. presidential election, the largest connected component of the political blogs network consists of 1222 blogs and 16,714 links. As seen in Figure 4.1a, blogs were classified according to their political ideology based on a text analysis of their content, where the 636 red nodes represent conservative leaning blogs and the 586 blue nodes represent liberal leaning blogs. There are drastically more connections between blogs of the same political ideology (precisely 15,139) than connections between blogs of differing political ideology (precisely 1,575). This translates to a propensity of connection between liberal (conservative) blogs of 0.043 (0.039), whereas the propensity of connection between liberal and conservative blogs is 0.004.

The political blogs network has been studied time and time again within the community discovery literature (Jin, 2015; Karrer and Newman, 2011; Newman, 2006, 2013), and in this vast body of work, political ideology is conflated with community structure. Authors deem their community discovery methods successful after dividing nodes into groups which largely align with the observed political ideology. However, this is arguably a rather trivial partition of the nodes which provides little insight about the connections between liberals and conservatives. Peel et al. (2017) further warn against treating node meta-data, like political ideology, as ground truth for community structure, recognizing that (1) community discovery is largely task dependent for which no method is universally optimal, and (2) the political blogs network has substantial substructure that is often overlooked in favor of the traditional narrative. By conditioning on political ideology, ECoHeN identifies communities of blogs which are densely connected considering the political ideology of each community's members.

When ECoHeN is applied to the political blogs network with the political ideology labels, 81 communities are found. For reasons discussed in Section 4.3.3, the number of communities is likely overstated due to a significant amount of overlap among the discovered communities. As such, these 81 communities are refined such that each community has at least four nodes with a $\beta = 0.10$ for maximum Jaccard overlap, which results in a set of 15 communities that overlap yet are largely distinct. The largest partisan (homogeneous) and bipartisan (heterogeneous) communities identified by ECoHeN are presented in Figure 4.7.

To gauge the quality of the 15 communities extracted by ECoHeN, we compute the *ratio of densities* (RatD) for each community; that is, we compute the density of links among community members divided by the density of links between community members and the rest of the network. Figure 4.8 provides the RatD for all ECoHeN communities. The RatD for all liberal (conservative) blogs is a natural baseline when assessing the assortativity of communities composed near entirely of liberals (conservatives). In general, a RatD of one implies that the density of links within a set of nodes is equivalent to the density of links to the rest of the network and is a natural baseline

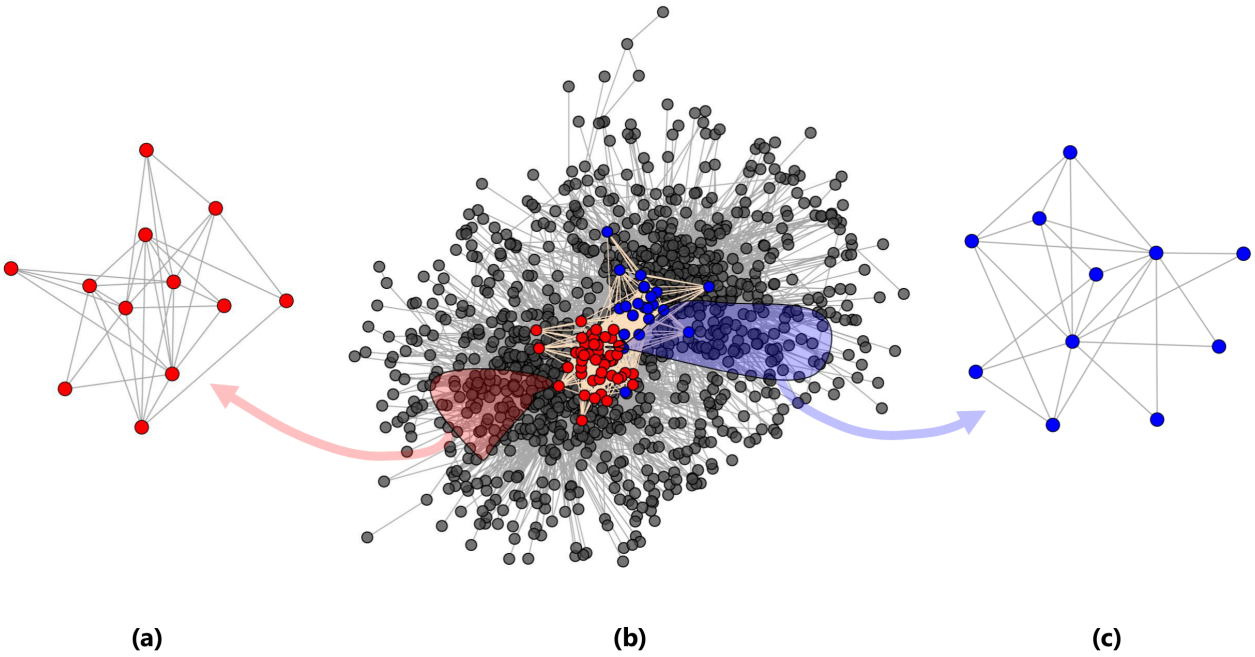


Figure 4.7: The largest partisan and bipartisan communities identified by ECoHeN. Panel (a) depicts the largest conservative community which consists of 12 blogs with a ratio of densities of 23.8. Panel (c) depicts the largest liberal community which consists of 11 blogs with a ratio of densities of 20.8. The colored polygons in panel (b) provide the origin of each partisan community in the political blogs network. Furthermore, the largest community found by ECoHeN is a bipartisan community provided in panel (b) which consists of 73 blogs with a ratio of densities of 8.5.

when assessing the assortativity of an identified community regardless of community members' political affiliation.

For comparison, we apply ZCmod and Walktrap to the political blogs network, attain respectively 11 and 5 communities with at least four members, and compute the RatD for each identified community. To assess political composition of each community, we compute the proportion of liberals (equivalently, conservatives) in each community. Figure 4.8 provides the RatD for each identified community along with the size and political composition of the community. When the network is partitioned irrespective of political affiliation with Walktrap, at least one community is largely comprised of liberals and one community is largely comprised of conservatives, a rather trivial partition. When the network is partitioned via heterogeneous modularity maximization using ZCmod, the resulting communities must maintain at least one node of each node type, a rather

stringent assumption. In comparison, ECoHeN is able to identify small, bipartisan and partisan communities, each with a connection density higher than competing methods.

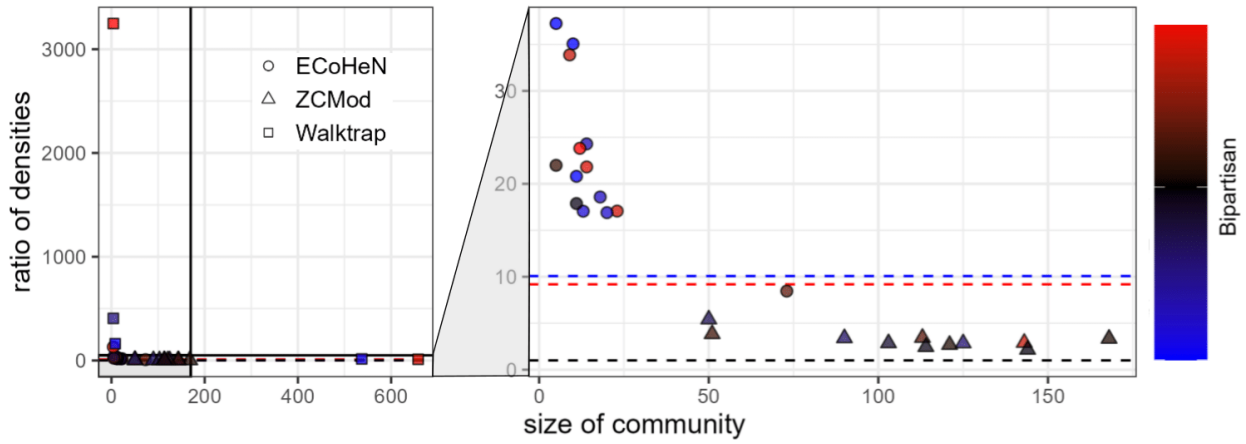


Figure 4.8: The ratio of density (RatD) for each of the 15 communities extracted from ECoHeN (circles), 11 communities detected from ZCmod (triangles), and five communities detected from Walktrap (squares) along with the size of the communities. Each community is colored according to the proportion of liberals (equivalently, conservatives) in the community where bluer (redder) points are largely composed of liberals (conservatives). Blacker points illustrate bipartisan communities. The black horizontal line illustrates the expected RatD in a random network. The blue (red) horizontal line illustrates the RatD provided all liberals (conservatives). As opposed to ZCmod, ECoHeN can identify both partisan and bipartisan communities. The largest bipartisan community found by ECoHeN features a RatD nearly 1.6 times larger than the largest ratio of densities attained via ZCmod. Walktrap partitions the network into at least two large communities which align with political ideology, a trivial partition which provides little insight about the connections between liberals and conservatives.

Both ECoHeN and ZCmod result in communities whose links are relatively more dense internally than to the rest of the network; however, the ratio of densities observed from ECoHeN communities are much higher than those communities from ZCmod. The larger RatD is partly because (1) the ECoHeN communities tend to be smaller than the ZCmod communities and (2) the fact that ZCmod is a partitioning method and must assign each node to a community, potentially diluting the density of otherwise well-connected collections of nodes. Nevertheless, the largest ECoHeN community (shown in Figure 4.7b has a RatD that is about 1.6 times larger than the ZCmod community with the largest RatD, both of which are largely bipartisan communities.

This application highlights the importance of conditioning on node type when performing community discovery. Since partisan links are particularly common compared to bipartisan links, the partisan communities identified by ECoHeN are also particularly dense—denser than the natural baseline induced from taking all liberals or all conservatives, respectively. On the other hand, the RatD required by ECoHeN to consider a set of bipartisan nodes a community is naturally lower, a testament to the strengths of ECoHeN as it leverages differences in the connectivity between node types. If a less connected political party, say independents, were included in the network, ECoHeN would be uniquely positioned to identify both partisan and bipartisan communities including independents since ECoHeN identifies communities considering the density with respect to type and does not place constraints or assumptions on the political composition of each community.

4.6 Discussion

ECoHeN is a generalization of an existing community extraction method called the extraction of statistically significant communities (ESSC). ECoHeN iteratively updates a candidate community by assessing the significance of connections between each node and the candidate community through a reference distribution derived under the heterogeneous degree configuration model. Like its predecessor ESSC, ECoHeN can identify background nodes and overlapping communities, two common properties of realistic networks. Compared to ECoHeN, many community discovery methods assign background nodes to otherwise tightly connected communities, reducing the overall density of the community, and assume that communities are disjoint, unrealistically positing that no node may be tightly connected to more than one collection of nodes. Unlike ESSC, ECoHeN takes advantage of differences between nodes' types and any resulting differences in the density of connections between them to identify communities. A key advantage of ECoHeN is its ability to discover communities that are topologically dense with respect to the node types of each community's members without making assumptions or imposing constraints on the resulting type composition of each community. ECoHeN is the first extraction method capable of identifying both homogeneous and heterogeneous community structure. Furthermore, ECoHeN can be pa-

rameterized such that it is guaranteed to converge, resolving issues with cycles present in ESSC's implementation.

Generalizations of ECoHeN are possible and an area for future work. One particular generalization of interest is the extension to directed, heterogeneous multigraphs which would be possible by generalizing the heterogeneous degree sequence of each node to include both in- and out-degrees. Other avenues of work include the derivation of a finite sampling distribution for the measure of connectivity $p_{\mathcal{B}}(u)$ as defined in (4.4) and a temporal network extension. Furthermore, while ECoHeN extracts communities parallelized across the initial seed sets and is partially implemented in `C++` for efficiency, scalability to large networks is still a concern. One way to improve scalability is to reconsider how the method is initialized (e.g., define locally optimal seed sets for heterogeneous networks), and conduct tests for inclusion and exclusion locally (e.g., test only direct neighbors for inclusion in the extraction procedure). The implications of such changes on the multiple testing correction and convergence properties of the algorithm is also an area of future work.

Chapter 5

Discussion

5.1 Overview

In this dissertation, we presented new methods and approaches for drawing causal inference with temporally dependent units and clustering nodes in a network. Particularly, in Chapter 2 we estimated the causal effect of a timeout at stopping an opposing run in the National Basketball Association (NBA) in light of the temporal dependence among runs. In Chapter 3, we outlined an analytic pipeline for the identification of clusters containing undiscovered gene to phenotype relations in a very large, temporal heterogeneous biological network. Finally, in Chapter 4, we introduced an iterative hypothesis testing procedure called ECoHeN to extract communities from heterogeneous networks in a statistically meaningful way. There are numerous ways the work presented here could be built upon. In this chapter, we discuss a possible characterization of the collection of snowballed subgraphs and an extension to heterogeneous networks. Furthermore, we discuss how the ECoHeN algorithm can be made more scalable for its motivating application: the large, biological network in Chapter 3.

5.2 Snowballed Subgraphs

Defined in Section 3.3.1 and illustrated in Figure 3.3, snowball sampling is a generative process for attaining subgraphs of specified size through network ties, originally developed by Coleman (1958) and Goodman (1961) to study the structure of social networks. Associated with any finite network is a finite collection of subgraphs possibly attained through snowball sampling. That is, if the network is finite, the sample space of snowballed subgraphs is finite; however, the likelihood of a subgraph being generated through snowball sampling is unknown. If this distribution were known, we could, for example, analytically compute the potential for future discovery in (3.2) and avoid the computationally cumbersome approximation provided in (3.3). This solution, how-

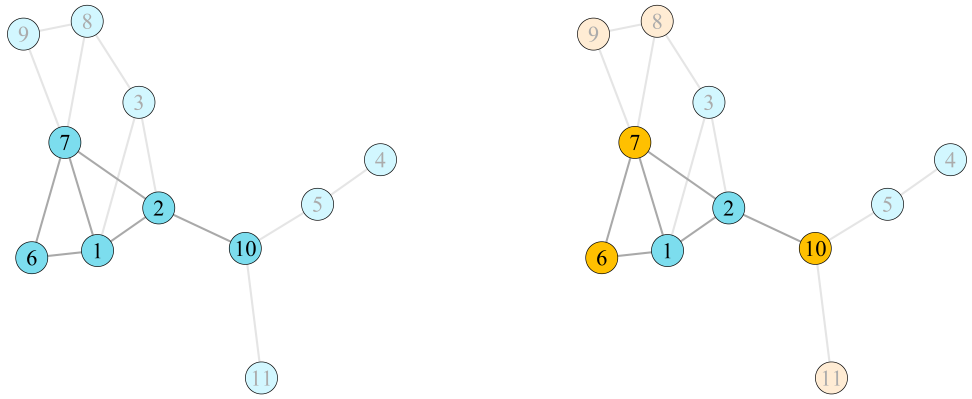
ever, does not address another issue highlighted in Section 3.6, which is that snowball sampling does not control the number of nodes of each type, only the total number of nodes.

In this section, we clearly define the sample space of snowballed subgraphs, that is, subgraphs attained from snowball sampling. While the concept of snowball sampling is not well-defined for heterogeneous networks, we can extend our definitions to incorporate constraints on the number of nodes of each type, rather than simply the number of nodes. Using our definitions, we outline the sample space of snowballed subgraphs in a heterogeneous network with constraint on the number of nodes of each type in each subgraph. Finally, we postulate how the probability of attaining each subgraph through snowball sampling could be connected to our definitions and demonstrate how our definitions could be used to sample snowballed subgraphs uniformly at random without enumerating the entire sample space of snowballed subgraphs.

To illustrate the utility of our definitions and demonstrate its applicability to homogeneous and heterogeneous networks, we consider the homogeneous and heterogeneous networks shown in Figure 5.1, denoted $G = (V, E)$ and $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, respectively. Notice, a relevant subgraph of the network is highlighted for each network. Furthermore, the edge structure of each network is identical; however, the heterogeneous network distinguishes between different types of nodes (illustrated by color) and their interconnections. To avoid degeneracies regarding isolates, we assume relevant networks have one connected component.

5.2.1 Sample Space for Homogeneous Networks

Snowball sampling from a homogeneous network yields subgraphs with minimally short path lengths between nodes. Let $\tilde{V} \subseteq V$ be a subset of the node set. We could formally say that an induced subgraph $G[\tilde{V}]$ is a snowballed subgraph of G of size $|\tilde{V}|$ if there exists a node $v \in \tilde{V}$ whose path length to every other node in $\tilde{V}$ is *shortest* and *minimal*. That is, the path length between v and every other node in $\tilde{V}$ in $G[\tilde{V}]$ is the shortest distance between the two nodes in G (i.e., shortest), and there does not exist a shorter path between v and any node in $V - \tilde{V}$ in G (i.e., minimal).



(a) $G = (V, E)$ **(b)** $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

Figure 5.1: Examples of a simple, homogeneous network, G, and a simple, heterogeneous network, $\mathcal{G}$, used to exemplify a characterization of snowballed subgraphs in homogeneous and heterogneeous networks based on path length. In each network, a subgraph composed of five nodes and six edges is highlighted.

To motivate this definition, consider the set $\tilde{V} = \{1, 6, 7\}$ from G and $v = 6$, for example. The induced subgraph, $G[\tilde{V}]$, is a triangle. In this case, the path from node 6 to node 7 is the shortest path between the two nodes, similarly for the path between node 6 and node 1. Furthermore, there does not exist a node in $V - \tilde{V}$ with a shorter path to node 6. Hence, the subgraph $G[\tilde{V}]$ is a snowballed subgraph of G of size 3. Less trivially, consider $\tilde{V} = \{6, 7, 8\}$ and $v = 7$. The induced subgraph, $G[\tilde{V}]$, mimics a line. The path from node 7 to node 6 is the shortest path between the two nodes, similarly for the path between node 7 and node 8. Furthermore, there does not exist a node in $V - \tilde{V}$ with a shorter path to node 7; however, there are other nodes with as short of a path (i.e., nodes 1, 2, and 9). Hence, $G[\{6, 7, 8\}]$ is a snowballed subgraph of G but so is $G[\{6, 7, 1\}]$.

Finally, consider $\tilde{V} = \{1, 3, 10\}$. The induced subgraph, $G[\tilde{V}]$, is composed of the connection between nodes 1 and 3 and an isolated node 10. Since node 10 is isolated from node 1 and 3 in the induced subgraph, the distance between them is assumed infinite. Since the distance between node 1 and node 10 in G, for example, is two, the path lengths between nodes in $G[\tilde{V}]$ fail to pass the shortest criterion. Hence, the induced subgraph $G[\tilde{V}]$ is not an induced snowballed subgraph of G.

Through enumeration, there are 26 different subgraphs of size three possibly attained from G in Figure 5.1a via snowball sampling. In total, there are 165 ways to construct a subgraph from G of size three. Applying our path length criteria to each of these 165 tuples of three nodes,

we find there are precisely 26 induced subgraphs fitting our definition of a snowballed subgraph (i.e., induced snowballed subgraphs). This suggests that our definition for an induced snowballed subgraph may provide a characterization for the sample space of subgraphs of a given size possibly attained through snowball sampling.

5.2.2 Extension to Heterogeneous Networks

In Chapter 3, an approximation to the distribution of snowballed subgraphs is used to score and rank clusters identified from the gene and phenotype network according to their potential for undiscovered gene to phenotype relations. By construction, the clusters identified from the network are heterogeneous whereas each snowballed subgraph need not be. This means a cluster's PFD is potentially overstated by failing to control the number of genes and phenotypes in each snowballed subgraph. A more appropriate null model for heterogeneous networks would account for the number of nodes of each type in and the relative density of a set of nodes. For example, if we are interested in computing the PFD of a cluster $\mathcal{C}$ with six genes and three phenotypes, we might wish to compare to a distribution of well-connected subgraphs with precisely six genes and three phenotypes. Outlining a generative process to attain such subgraphs is difficult as one can fall victim to the pigeonhole principle; imagine, for example, starting with a gene adjacent to ten phenotypes. In this case, there would be no way to attain a set of six genes and three phenotypes with snowball sampling.

To address these concerns, we could extend the definition of induced snowballed subgraphs to define a collection of subgraphs with a specified number of nodes of each type whose path lengths are minimally short within type. Let $\tilde{\mathcal{V}} \subseteq \mathcal{V}$ denote a subset of the node set. Similar to $\mathcal{V}$, the subset $\tilde{\mathcal{V}} = \bigcup_{k=1}^{K} \tilde{V}^{[k]}$ where $\tilde{V}^{[k]}$ denotes the subset of $\tilde{\mathcal{V}}$ containing $|\tilde{V}^{[k]}|$ nodes of type k and $\tilde{V}^{[k]} \cap \tilde{V}^{[l]} = \emptyset$. We say that an induced subgraph $\mathcal{G}[\tilde{\mathcal{V}}]$ is a snowballed subgraph of $\mathcal{G}$ with size $(|\tilde{V}^{[1]}|, \ldots, |\tilde{V}^{[K]}|)$ if there exists a node $v^{[l]} \in \tilde{V}^{[l]}$ whose path length to every other node in $\tilde{\mathcal{V}}$ is *shortest* and *minimal within type*. That is, the path length between $v^{[l]}$ and each $u^{[k]} \in \tilde{V}^{[k]}$ in $\mathcal{G}[\tilde{\mathcal{V}}]$

is the shortest distance between the two nodes in $\mathcal{G}$ (i.e., shortest), and there does not exist a shorter path between $v^{[l]}$ and any node in $V^{[k]} - \tilde{V}^{[k]}$ in $\mathcal{G}$ (i.e., minimal within type) for $k \in \{1, 2, \ldots, K\}$.

Consider the set $\tilde{\mathcal{V}} = \{1^{[1]}, 2^{[1]}, 6^{[2]}, 7^{[2]}, 10^{[2]}\}$ from $\mathcal{G}$ and $v = 2$, for example. The induced subgraph, $\mathcal{G}[\tilde{\mathcal{V}}]$, is composed of two blue nodes, three orange nodes, and six edges between them—the highlighted subgraph in Figure 5.1b. In this case, the path between node 2 and every other node in $\mathcal{V}$ is shortest. Furthermore, the path lengths are minimal within type since there does not exist an external blue node with a shorter path length to node 2, similarly for external orange nodes. Notice, for example, the distance between node 2 and node 6 is two, but the distance between node 2 and the external orange nodes (i.e., nodes 8, 9, 10, and 11) is no less than two. Similarly, the distance between node 2 and node 1 is one, but the distance between node 2 and the external blue nodes (i.e., nodes 3, 4, and 5) is no less than one. Hence, $\mathcal{G}[\mathcal{V}]$ is an induced snowballed subgraph of $\mathcal{G}$ of size (2, 3). In total, there are 200 ways to construct a subgraph from $\mathcal{G}$ in 5.1b with two blue nodes and three orange nodes. Applying our path length criteria to each, there are 18 induced snowballed subgraphs of size (2, 3).

On the other hand, consider the node type agnostic set $\tilde{V} = \{1, 2, 6, 7, 10\}$ from G. The induced subgraph, $G[\tilde{V}]$, is composed of five nodes and six edges between them—the highlighted subgraph in Figure 5.1a. Since the edge structure in G and $\mathcal{G}$ is the same, we've verified the path lengths to node 2 are shortest; however, the path lengths are not minimal since node 3 has a smaller distance to node 2. In fact, no node in $\tilde{V}$ satisfies the minimal criterion. Hence, the induced subgraph $G[\tilde{V}]$ is not an induced snowballed subgraph of G of size 5. This example illustrates that our definition of a snowballed subgraph and generalization to heterogeneous networks depends both on the structure of the edges and the node types.

5.2.3 Future Work

Problematically, since not every snowballed subgraph (attained through snowball sampling) is uniquely identified by an individual seed node selected uniformly at random, the likelihood of a subgraph being generated by snowball sampling is unknown. For future work, we hope to charac-

terize the distribution of snowballed subgraphs and establish a connection between the probability of a network being generated from snowball sampling and the number of seed nodes which could possibly generate the subgraph. We believe that the likelihood of a snowballed subgraph being generated from snowball sampling is related to the number of nodes, v, in the subgraph which satisfy the minimal condition of our definition. If true, one could derive the likelihood of a snowballed subgraph if one could enumerate the possible snowballed subgraphs.

Often, however, enumerating the sample space of snowballed subgraphs is infeasible given the size of the network and/or the size of the cluster. In this case, one can use snowball sampling to approximate the (currently) unknown distribution of subgraphs attained through snowball sampling. Another approach would be to consider the uniform distribution of snowballed subgraphs. If the sample space cannot be enumerated, it is unclear how to sample from the uniform distribution of snowballed subgraphs using a generative process like snowball sampling, but it can be done using our definitions. For example, to sample a snowballed subgraph of size three uniformly at random from the network in Figure 5.1a, draw without replacement three nodes from the node set $\{1, 2, \ldots, 11\}$ and check if the tuple is an induced snowballed subgraph, by definition. If not, repeat until a subgraph whose path lengths are both shortest and minimal is attained. While this process can be computationally burdensome with many rejections, the distribution of snowballed subgraphs is uniform without needing to enumerate the sample space.

5.3 Scalability of ECoHeN

ECoHeN was motivated by the biological network presented in Chapter 3. Problematically, however, ECoHeN is not scalable to the biological network motivating its creation. Investigations into options for increasing the scalability of ECoHeN would lead to greater applicability of the method to large networks. Two ways to improve the scalability of ECoHeN is (1) to test locally for the inclusion and exclusion of nodes and (2) to redesign how the maximal allowance is parameterized at each iteration.

Suppose a practitioner supplies a list of genetic variants and abnormal phenotypes as a seed set for ECoHeN. By design, ECoHeN will refine this seed set to a community (possibly empty) through a two-step, dynamic procedure by iteratively including any well-connected external nodes and subsequently removing any loosely connected internal nodes. By design, ECoHeN tests all external nodes for inclusion even though the only ones likely added are neighbors of nodes in the current candidate set. Hence, testing only the subset of external neighbors for inclusion may improve the scalability of ECoHeN since the set of neighbors for each node can be computed and stored once.

Another way to improve the scalability of ECoHeN is to redesign how the maximal allowance is parameterized at each iteration. The maximal allowance places restriction on the number of nodes possibly allowed into or out of the candidate set with each iteration. By reducing the maximal allowance to one with each iteration, we are able to guarantee that ECoHeN will never cycle between two candidate sets. We proposed initializing the maximum allowance to a large value and then progressively decrease it according to an exponential decay function. This, however, can result in ECoHeN alternating between two candidate sets for a finite number of iterations until the maximal allowance decreases to one. Another way to guarantee convergence while fixing these inefficiencies in implementation would be to let the maximal allowance be $|\mathcal{V}|$ until a single alternation is detected. When a single alternation is detected, reduce the maximal allowance to one at which point the extraction procedure will refine the candidate set one node at a time until convergence.

Bibliography

Abadie, A. (2002). Bootstrap tests for distributional treatment effects in instrumental variable models. *Journal of the American Statistical Association*, 97(457):284–292.

Abadie, A., Drukker, D., Herr, J. L., and Imbens, G. W. (2004). Implementing matching estimators for average treatment effects in stata. *The Stata Journal*, 4(3):290–311.

Adamic, L. A. and Glance, N. (2005). The political blogosphere and the 2004 US election: Divided they blog. In *Proceedings of the 3rd International Workshop on Link Discovery*, pages 36–43.

Alshahrani, M., Khan, M. A., Maddouri, O., Kinjo, A. R., Queralt-Rosinach, N., and Hoehndorf, R. (2017). Neuro-symbolic representation learning on biological knowledge graphs. *Bioinformatics*, 33(17):2723–2730.

Aschburner, S. (2017). NBA changes timeout rules to improve game flow. https://www.nba.com/article/2017/07/12/nba-board-governors-timeout-rules-game-flow-trade-deadline [Accessed: 2020-11-12].

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Eppig, J. T., et al. (2000). Gene ontology: Tool for the unification of biology. *Nature Genetics*, 25(1):25–29.

Assis, N., Assunção, R., and Vaz-De-Melo, P. O. (2020). Stop the clock: Are timeout effects real? *arXiv preprint arXiv:2009.06750*.

Avugos, S., Köppen, J., Czienskowski, U., Raab, M., and Bar-Eli, M. (2013). The "hot hand" reconsidered: A meta-analytic approach. *Psychology of Sport and Exercise*, 14(1):21–27.

Bard, J. B. and Rhee, S. Y. (2004). Ontologies in biology: Design, applications and future challenges. *Nature Reviews Genetics*, 5(3):213–222.

Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(1):289–300.

Beuoy, M. (2021). inpredictable: NBA per-possession statistics. http://stats.inpredictable.com/nba/ssnTeamPoss.php?season=2018&po=0&frdt=2017-10-17&todt=2019-04-10&view=off [Accessed: 2021-07-01].

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008.

Bodwin, K., Zhang, K., and Nobel, A. (2018). A testing based approach to the discovery of differentially correlated variable sets. *The Annals of Applied Statistics*, 12(2):1180.

Bonald, T., Charpentier, B., Galland, A., and Hollocou, A. (2018). Hierarchical graph clustering using node pair sampling. *arXiv preprint arXiv:1806.01664*.

Boren, C. (2019). Raptors' coach Nick Nurse ripped for timeout that helped the Warriors. https://www.washingtonpost.com/sports/2019/06/11/raptors-coach-nick-nurse-ripped-timeout-that-helped-warriors/ [Accessed: 2020-11-12].

Brandes, U., Delling, D., Gaertler, M., Görke, R., Hoefer, M., Nikoloski, Z., and Wagner, D. (2006). Maximizing modularity is hard. *arXiv preprint physics/0608255*.

Bresler, A. (2019). *nbastatR: R's interface to NBA data*. R package version 0.1.120301.

Busch, R. (1990). On the history of cystic fibrosis. *Acta Universitatis Carolinae. Medica*, 36(1-4):13–15.

Callahan, T. J., Tripodi, I. J., Hunter, L. E., and Baumgartner, W. A. (2020). A framework for automated construction of heterogeneous large-scale biomedical knowledge graphs. *bioRxiv*.

Cantwell, C. Y., Fortman, J., and Seegan, A. (2021). Prazosin use in a patient with rare neurobeachin gene deletion shows improvement in paranoid behavior: A case report. *Journal of Medical Case Reports*, 15(1):1–4.

Carreras, X. and Marquez, L. (2001). Boosting trees for anti-spam email filtering. *arXiv preprint cs/0109015*.

Castellani, S., Guerra, L., Favia, M., Di Gioia, S., Casavola, V., and Conese, M. (2012). NHERF1 and CFTR restore tight junction organisation and function in cystic fibrosis airway epithelial cells: Role of ezrin and the rhoa/rock pathway. *Laboratory Investigation*, 92(11):1527–1540.

Chen, T., Benesty, M., He, T., and Tang, Y. (2018). Understand your dataset with xgboost. *R Document*.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM Sigkdd International Conference on Knowledge Discovery and Data Mining*, pages 785–794.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T., Li, M., Xie, J., Lin, M., Geng, Y., Li, Y., and Yuan, J. (2022). *xgboost: Extreme Gradient Boosting*. R package version 1.6.0.1.

Chong, J. X., Yu, J.-H., Lorentzen, P., Park, K. M., Jamal, S. M., Tabor, H. K., Rauch, A., Saenz, M. S., Boltshauser, E., Patterson, K. E., et al. (2016). Gene discovery for mendelian conditions via social networking: De novo variants in KDM1A cause developmental delay and distinctive facial features. *Genetics in Medicine*, 18(8):788–795.

Choobdar, S., Ahsen, M., Crawford, J., Tomasoni, M., Lamparter, D., Lin, J., Hescott, B., Hu, X., Mercer, J., Natoli, T., et al. (2018). Open community challenge reveals molecular network modules with key roles in diseases.

Choobdar, S., Ahsen, M. E., Crawford, J., Tomasoni, M., Fang, T., Lamparter, D., Lin, J., Hescott, B., Hu, X., Mercer, J., et al. (2019). Assessment of network module identification across complex diseases. *Nature Methods*, 16(9):843–852.

Clauset, A., Newman, M. E., and Moore, C. (2004). Finding community structure in very large networks. *Physical Review E*, 70(6):066111.

Cleveland Clinic (2021). Genetic disorders: What are they, types, symptoms and causes.

Coleman, J. (1958). Relational analysis: The study of social organizations with survey methods. *Human Organization*, 17(4):28–36.

Csardi, G. and Nepusz, T. (2006). The igraph software package for complex network research. *InterJournal*, Complex Systems:1695.

Curtis, C. (2019). The NBA rule about timeouts that partially explains Nick Nurse's weird decision to take one. https://ftw.usatoday.com/2019/06/nba-finals-raptors-nick-nurse-timeout-rule [Accessed: 2020-11-12].

De Meo, P., Ferrara, E., Fiumara, G., and Provetti, A. (2011). Generalized louvain method for community detection in large networks. In *2011 11th International Conference on Intelligent Systems Design and Applications*, pages 88–93. IEEE.

Deshpande, S. K. and Evans, K. (2020). Expected hypothetical completion probability. *Journal of Quantitative Analysis in Sports*, 16(2):85–94.

Diamond, A. and Sekhon, J. S. (2013). Genetic matching for estimating causal effects: A general multivariate matching method for achieving balance in observational studies. *Review of Economics and Statistics*, 95(3):932,945.

Edler, D., Eriksson, A., and Rosvall, M. (2022). The mapequation software package.

Erdős, P. and Rényi, A. (1960). On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5(1):17–60.

ESPN (2019). Nick Nurse disrupted Kawhi's flow, late timeout cost the Raptors in Game 5. https://www.youtube.com/watch?v=UvOvNy-3etc [Accessed 2020-11-12].

Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, 486(3-5):75–174.

Fortunato, S. and Barthelemy, M. (2007). Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41.

Fosdick, B. K., Larremore, D. B., Nishimura, J., and Ugander, J. (2018). Configuring random graph models with fixed degree sequences. *Siam Review*, 60(2):315–355.

Franks, A., Miller, A., Bornn, L., and Goldsberry, K. (2015). Characterizing the spatial structure of defensive skill in professional basketball. *The Annals of Applied Statistics*, 9(1):94–121.

Friedman, J. H. and Meulman, J. J. (2003). Multiple additive regression trees with application in epidemiology. *Statistics in Medicine*, 22(9):1365–1381.

García-Algarra, J., Mouronte-López, M. L., and Galeano, J. (2019). A stochastic generative model of the world trade network. *Scientific Reports*, 9(1):1–10.

Gene Ontology Consortium (2019). The gene ontology resource: 20 years and still GOing strong. *Nucleic Acids Research*, 47(D1):D330–D338.

Ghasemian, A., Hosseinmardi, H., Galstyan, A., Airoldi, E. M., and Clauset, A. (2020). Stacking models for nearly optimal link prediction in complex networks. *Proceedings of the National Academy of Sciences*, 117(38):23393–23400.

Gibbs, C. P. (2022). ECoHeN. https://github.com/ConGibbs10/ECoHeN [Accessed: 2023-01-24].

Gilbert, E. N. (1959). Random graphs. *The Annals of Mathematical Statistics*, 30(4):1141–1144.

Gillis, J. and Pavlidis, P. (2012). "Guilt by association" is the exception rather than the rule in gene networks. *PLoS Computational Biology*, 8(3):e1002444.

Gilovich, T., Vallone, R., and Tversky, A. (1985). The hot hand in basketball: On the misperception of random sequences. *Cognitive Psychology*, 17(3):295–314.

Girvan, M. and Newman, M. E. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826.

Gleich, D. F. and Seshadhri, C. (2012). Vertex neighborhoods, low conductance cuts, and good seeds for local community methods. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 597–605.

Goodman, L. A. (1961). Snowball sampling. *The Annals of Mathematical Statistics*, pages 148–170.

Hagberg, A., Swart, P., and S Chult, D. (2008). Exploring network structure, dynamics, and function using NetworkX. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

Hamosh, A., Scott, A. F., Amberger, J. S., Bocchini, C. A., and McKusick, V. A. (2005). Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders. *Nucleic acids research*, 33(suppl_1):D514–D517.

Hastie, T. J. and Tibshirani, R. J. (1990). *Generalized additive models*, volume 43. CRC Press.

Heckman, J. J., Ichimura, H., and Todd, P. (1998). Matching as an econometric evaluation estimator. *The Review of Economic Studies*, 65(2):261–294.

Heckman, J. J., Ichimura, H., and Todd, P. E. (1997). Matching as an econometric evaluation estimator: Evidence from evaluating a job training programme. *The Review of Economic Studies*, 64(4):605–654.

Imbens, G. W. and Rubin, D. B. (2015). *Causal inference for statistics, social, and biomedical sciences*. Cambridge University Press.

International Human Genome Sequencing Consortium (2004). Finishing the euchromatic sequence of the human genome. *Nature*, 431(7011):931–945.

Javed, M. A., Younis, M. S., Latif, S., Qadir, J., and Baig, A. (2018). Community detection in networks: A multidisciplinary review. *Journal of Network and Computer Applications*, 108:87–111.

Jin, J. (2015). Fast community detection by score. *The Annals of Statistics*, 43(1):57–89.

Johnson, M. E., Moore, L. M., and Ylvisaker, D. (1990). Minimax and maximin distance designs. *Journal of Statistical Planning and Inference*, 26(2):131–148.

Karczewski, K. J., Francioli, L. C., Tiao, G., Cummings, B. B., Alföldi, J., Wang, Q., Collins, R. L., Laricchia, K. M., Ganna, A., Birnbaum, D. P., et al. (2020). The mutational constraint spectrum quantified from variation in 141,456 humans. *Nature*, 581(7809):434–443.

Karrer, B. and Newman, M. E. (2011). Stochastic blockmodels and community structure in networks. *Physical Review E*, 83(1):016107.

Kendall, M. G. (1948). *Rank correlation methods*. Griffin.

Koehler, J. J. and Conley, C. A. (2003). The "hot hand" myth in professional basketball. *Journal of Sport and Exercise Psychology*, 25(2):253–259.

Köhler, S., Vasilevsky, N. A., Engelstad, M., Foster, E., McMurry, J., Aymé, S., Baynam, G., Bello, S. M., Boerkoel, C. F., Boycott, K. M., et al. (2017). The human phenotype ontology in 2017. *Nucleic Acids Research*, 45(D1):D865–D876.

Kuhn, M. and Frick, H. (2022). *dials: Tools for creating tuning parameter values*. R package version 1.1.0.

Kushima, I., Aleksic, B., Nakatochi, M., Shimamura, T., Okada, T., Uno, Y., Morikawa, M., Ishizuka, K., Shiino, T., Kimura, H., et al. (2018). Comparative analyses of copy-number vari-

ation in autism spectrum disorder and schizophrenia reveal etiological overlap and biological insights. *Cell Reports*, 24(11):2838–2856.

Lancichinetti, A. and Fortunato, S. (2009). Community detection algorithms: A comparative analysis. *Physical Review E*, 80(5):056117.

Lancichinetti, A., Radicchi, F., Ramasco, J. J., and Fortunato, S. (2011). Finding statistically significant communities in networks. *PLoS One*, 6(4):e18961.

Lauletta, T. (2019). Nick Nurse called a bizarre timeout late in the fourth quarter that killed the Raptors momentum and sent the Warriors on their game-winning hot streak. https://www.businessinsider.com/nick-nurse-timeout-warriors-hot-streak-raptors-momentum-2019-6/ [Accessed: 2020-11-12].

Leskovec, J., Lang, K. J., and Mahoney, M. (2010). Empirical comparison of algorithms for network community detection. In *Proceedings of the 19th International Conference on World Wide Web*, pages 631–640.

Leskovec, J. and Sosič, R. (2016). Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1):1–20.

Li, P. (2012). Robust logitboost and adaptive base class (abc) logitboost. *arXiv preprint arXiv:1203.3491*.

Li, Y., Sha, C., Huang, X., and Zhang, Y. (2018). Community detection in attributed graphs: An embedding approach. In *Thirty-second AAAI Conference on Artificial Intelligence*.

Liben-Nowell, D. and Kleinberg, J. (2003). The link prediction problem for social networks. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, pages 556–559.

Liu, W., Kuramoto, S. J., and Stuart, E. A. (2013). An introduction to sensitivity analysis for unobserved confounding in non-experimental prevention research. *Prevention Science*, 14(6):570–580.

Liu, X., Liu, W., Murata, T., and Wakita, K. (2014). A framework for community detection in heterogeneous multi-relational networks. *Advances in Complex Systems*, 17(06):1450018.

Lopez, M. J. and Gutman, R. (2017). Estimation of causal effects with multiple treatments: A review and new ideas. *Statistical Science*, 32(3):432–454.

Lopez, M. J., Matthews, G. J., and Baumer, B. S. (2018). How often does the best team win? A unified approach to understanding randomness in North American sport. *The Annals of Applied Statistics*, 12(4):2483–2516.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30.

Martin, S. (2022). BOCC. https://github.com/sky123martin/BOCC [Accessed: 2023-02-08].

Martínez, V., Berzal, F., and Cubero, J.-C. (2016). A survey of link prediction in complex networks. *ACM Computing Surveys (CSUR)*, 49(4):1–33.

McPherson, M., Smith-Lovin, L., and Cook, J. M. (2001). Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, pages 415–444.

Mi, H., Muruganujan, A., Huang, X., Ebert, D., Mills, C., Guo, X., and Thomas, P. D. (2019). Protocol update for large-scale genome and gene function analysis with the panther classification system (v. 14.0). *Nature Protocols*, 14(3):703–721.

Miller, J. B. and Sanjurjo, A. (2018). Surprised by the hot hand fallacy? A truth in the law of small numbers. *Econometrica*, 86(6):2019–2047.

Mukherjee, A., Choudhury, M., Peruani, F., Ganguly, N., and Mitra, B. (2013). *Dynamics on and of complex networks, volume 2: Applications to time-varying dynamical systems*, page 209. Springer New York, New York, NY, 2013 edition.

Mulhern, M. S., Stumpel, C., Stong, N., Brunner, H. G., Bier, L., Lippa, N., Riviello, J., Rouhl, R. P., Kempers, M., Pfundt, R., et al. (2018). NBEA: Developmental disease gene with early generalized epilepsy phenotypes. *Annals of Neurology*, 84(5):788–795.

Nacu, Ş., Critchley-Thorne, R., Lee, P., and Holmes, S. (2007). Gene expression network analysis and applications to immunology. *Bioinformatics*, 23(7):850–858.

NBA (2020). NBA advanced statistics. https://stats.nba.com [Accessed: 2020-11-12].

Newman, M. (2018). *Networks*. Oxford University Press.

Newman, M. E. (2006). Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582.

Newman, M. E. (2013). Spectral methods for community detection and graph partitioning. *Physical Review E*, 88(4):042822.

Newman, M. E. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113.

Olszewski, P. K., Rozman, J., Jacobsson, J. A., Rathkolb, B., Strömberg, S., Hans, W., Klockars, A., Alsiö, J., Riserus, U., Becker, L., et al. (2012). Neurobeachin, a regulator of synaptic protein targeting, is associated with body fat mass and feeding behavior in mice and body-mass index in humans. *PLoS Genetics*, 8(3):e1002568.

Opitz, D. and Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198.

Parker, C., Scott, S., and Geddes, A. (2019). Snowball sampling. *SAGE Research Methods Foundations*.

Peel, L., Larremore, D. B., and Clauset, A. (2017). The ground truth about metadata and community detection in networks. *Science Advances*, 3(5):e1602548.

Permutt, S. (2011). *The efficacy of momentum-stopping timeouts on short-term performance in the National Basketball Association*. PhD thesis.

Pina, M. (2019). To call a timeout, or not to call a timeout: That is the question for NBA coaches. https://sports.yahoo.com/call-timeout-not-call-timeout-141100462.html [Accessed: 2020-11-12].

Piñero, J., Bravo, À., Queralt-Rosinach, N., Gutiérrez-Sacristán, A., Deu-Pons, J., Centeno, E., García-García, J., Sanz, F., and Furlong, L. I. (2016). Disgenet: A comprehensive platform integrating information on human disease-associated genes and variants. *Nucleic Acids Research*.

Pons, P. and Latapy, M. (2006). Computing communities in large networks using random walks. In *Journal of Graph Algorithms and Applications*. Citeseer.

Psorakis, I., Roberts, S., Ebden, M., and Sheldon, B. (2011). Overlapping community detection using Bayesian non-negative matrix factorization. *Physical Review E*, 83(6):066114.

R Core Team (2022). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria.

R Core Team (2020). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria.

Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., and Parisi, D. (2004). Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences*, 101(9):2658–2663.

Ramoni, R. B., Mulvihill, J. J., Adams, D. R., Allard, P., Ashley, E. A., Bernstein, J. A., Gahl, W. A., Hamid, R., Loscalzo, J., McCray, A. T., et al. (2017). The undiagnosed diseases network: Accelerating discovery about health and disease. *The American Journal of Human Genetics*, 100(2):185–192.

Rosenbaum, P. R. (2007). Sensitivity analysis for m-estimates, tests, and confidence intervals in matched observational studies. *Biometrics*, 63(2):456–464.

Rosenbaum, P. R. (2013). Impact of multiple matched controls on design sensitivity in observational studies. *Biometrics*, 69(1):118–127.

Rosenbaum, P. R. (2015). Two R packages for sensitivity analysis in observational studies. *Observational Studies*, 1(1):1–17.

Rosenbaum, P. R. and Rubin, D. B. (1983). The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55.

Rosenbaum, P. R. and Rubin, D. B. (1985). Constructing a control group using multivariate matched sampling methods that incorporate the propensity score. *The American Statistician*, 39(1):33–38.

Rossetti, G., Milli, L., and Cazabet, R. (2019). CDLIB: A python library to extract, compare and evaluate communities from complex networks. *Applied Network Science*, 4(1):1–26.

Rosvall, M., Axelsson, D., and Bergstrom, C. T. (2009). The map equation. *The European Physical Journal Special Topics*, 178(1):13–23.

Rosvall, M. and Bergstrom, C. T. (2007a). An information-theoretic framework for resolving community structure in complex networks. *Proceedings of the National Academy of Sciences*, 104(18):7327–7331.

Rosvall, M. and Bergstrom, C. T. (2007b). Maps of information flow reveal community structure in complex networks. *arXiv preprint physics.soc-ph/0707.0609*.

Rubin, D. B. (1974). Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of Educational Psychology*, 66(5):688–701.

Rubin, D. B. (1976). Inference and missing data. *Biometrika*, 63(3):581–592.

Rubin, D. B. (1977). Assignment to treatment group on the basis of a covariate. *Journal of Educational Statistics*, 2(1):1–26.

Saavedra, S., Mukherjee, S., and Bagrow, J. P. (2012). Is coaching experience associated with effective use of timeouts in basketball? *Scientific Reports*, 2:676.

Scheipl, F., Greven, S., and Küchenhoff, H. (2008). Size and power of tests for a zero random effect variance or polynomial regression in additive and linear mixed models. *Computational Statistics & Data Analysis*, 52(7):3283–3299.

Sekhon, J. S. (2011). Multivariate and propensity score matching software with automated balance optimization: The matching package for R. *Journal of Statistical Software*, 42(7):1–52.

Sengupta, S. and Chen, Y. (2015). Spectral clustering in heterogeneous networks. *Statistica Sinica*, pages 1081–1106.

Shi, C., Li, Y., Zhang, J., Sun, Y., and Philip, S. Y. (2016). A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):17–37.

Smith, J. A. and Todd, P. E. (2005). Does matching overcome LaLonde's critique of nonexperimental estimators? *Journal of Econometrics*, 125(1-2):305–353.

Smith, L. M., Zhu, L., Lerman, K., and Percus, A. G. (2016). Partitioning networks with node attributes by compressing information flow. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 11(2):1–26.

Smith, N. R., Zivich, P. N., Frerichs, L. M., Moody, J., and Aiello, A. E. (2020). A guide for choosing community detection algorithms in social network studies: The question alignment approach. *American Journal of Preventive Medicine*, 59(4):597–605.

Soundarajan, S. and Hopcroft, J. (2012). Using community information to improve the precision of link prediction methods. In *Proceedings of the 21st International Conference on World Wide Web*, pages 607–608.

Stuart, E. A. (2010). Matching methods for causal inference: A review and a look forward. *Statistical Science*, 25(1):1 – 21.

Szklarczyk, D., Morris, J. H., Cook, H., Kuhn, M., Wyder, S., Simonovic, M., Santos, A., Doncheva, N. T., Roth, A., Bork, P., et al. (2016). The STRING database in 2017: Quality-controlled protein–protein association networks, made broadly accessible. *Nucleic Acids Research*, page gkw937.

Toumi, A. and Lopez, M. (2019). From grapes and prunes to apples and apples: Using matched methods to estimate optimal zone entry decision-making in the national hockey league. Carnegie Mellon Sports Analytics Conference.

Traag, V. A., Aldecoa, R., and Delvenne, J.-C. (2015). Detecting communities using asymptotical surprise. *Physical Review E*, 92(2):022816.

Traag, V. A. and Bruggeman, J. (2009). Community detection in networks with positive and negative links. *Physical Review E*, 80(3):036115.

UDN (2022). Undiagnosed Diseases Network (UDN) - participant 068. https://undiagnosed.hms.harvard.edu/participants/participant-068/ [Accessed: 2023-01-16].

Van Rossum, G. and Drake, F. L. (2009). *Python 3 reference manual*. CreateSpace, Scotts Valley, CA.

Vinayak, R. K. and Gilad-Bachrach, R. (2015). DART: Dropouts meet multiple additive regression trees. In *Artificial Intelligence and Statistics*, pages 489–497. PMLR.

Vock, D. M. and Vock, L. F. B. (2018). Estimating the effect of plate discipline using a causal inference framework: An application of the G-computation algorithm. *Journal of Quantitative Analysis in Sports*, 14(2):37–56.

Wasserman, S., Faust, K., et al. (1994). *Social network analysis: Methods and applications*. Cambridge University Press.

Weinreich, S. S., Mangon, R., Sikkens, J., Teeuw, M. e., and Cornel, M. (2008). Orphanet: A European database for rare diseases. *Nederlands Tijdschrift voor Geneeskunde*, 152(9):518–519.

Whang, J. J., Gleich, D. F., and Dhillon, I. S. (2013). Overlapping community detection using seed set expansion. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, pages 2099–2108.

Wilson, J. D., Palowitch, J., Bhamidi, S., and Nobel, A. B. (2017). Community extraction in multilayer networks with heterogeneous community structure. *The Journal of Machine Learning Research*, 18(1):5458–5506.

Wilson, J. D., Wang, S., Mucha, P. J., Bhamidi, S., Nobel, A. B., et al. (2014). A testing based extraction algorithm for identifying significant communities in networks. *Annals of Applied Statistics*, 8(3):1853–1891.

Wolfe, C. J., Kohane, I. S., and Butte, A. J. (2005). Systematic survey reveals general applicability of "guilt-by-association" within gene coexpression networks. *BMC Bioinformatics*, 6(1):1–10.

Wood, S. N. (2013). On p-values for smooth components of an extended generalized additive model. *Biometrika*, 100(1):221–228.

Wu, F. and Huberman, B. A. (2004). Finding communities in linear time: A physics approach. *The European Physical Journal B*, 38(2):331–338.

Xie, J. and Szymanski, B. K. (2011). Community detection using a neighborhood strength driven label propagation algorithm. In *2011 IEEE Network Science Workshop*, pages 188–195. IEEE.

Yam, D. R. and Lopez, M. J. (2019). What was lost? A causal estimate of fourth down behavior in the National Football League. *Journal of Sports Analytics*, 5(3):153–167.

Yang, J., McAuley, J., and Leskovec, J. (2013). Community detection in networks with node attributes. In *2013 IEEE 13th International Conference on Data Mining*, pages 1151–1156. IEEE.

Young, R. L., Weinberg, J., Vieira, V., Ozonoff, A., and Webster, T. F. (2011). Generalized additive models and inflated type I error rates of smoother significance tests. *Computational Statistics & Data Analysis*, 55(1):366–374.

Yousuf, S. (2018). Rick Carlisle is one of the NBA's best tacticians. How does he choose when to call a timeout? 'It's not a simple answer.'. https://theathletic.com/723304/2018/12/19/rick-carlisle-is-one-of-the-nbas-best-taciticians-how-does-he-choose-when-to-call-a-timeout-its-not-a-simple-answer [Accessed: 2020-11-12].

Zhang, J. and Chen, Y. (2020). Modularity based community detection in heterogeneous networks. *Statistica Sinica*, 30(2):601–629.

Zhang, Z., Kim, H. J., Lonjon, G., and Zhu, Y. (2019). Balance diagnostics after propensity score matching. *Annals of Translational Medicine*, 7(1):16.

Zhao, Y., Levina, E., and Zhu, J. (2011). Community extraction for social networks. *Proceedings of the National Academy of Sciences*, 108(18):7321–7326.

Zimmerman, D. L., Tang, J., and Huang, R. (2019). Outline analyses of the called strike zone in major league baseball. *The Annals of Applied Statistics*, 13(4):2416–2451.

Appendix A

Causal Effect of a Timeout in the NBA

A.1 Characterizing SUTVA

To address violations of the SUTVA, criteria (2) and (3) in the unit definition in Section 2.3.2 were introduced. These criteria handle issues with hidden variation and interference attributed to plays involving a timeout occurring in the pre-treatment or post-treatment window. While mitigating serious violations to the SUTVA, these criteria allow a reasonable control pool for matching on time-dependent covariates. All violations to the SUTVA are provided in Figure A.1, grouped by those cases which are resolved and unresolved by the current criteria.

By removing instances which feature a timeout in the pre-treatment or post-treatment window, we largely address concerns with overlapping windows among those units in the treated group (Figures A.1a, A.1b, and A.1c) and curtail issues with hidden variations (Figure A.1a). The only complication remaining are overlapping instances where the intersection between the windows is relatively small (Figures A.1d and A.1f).

Furthermore, units in the control group still experience overlapping windows (Figure A.1e). To resolve the issue with overlapping control units, controls would need to be removed until the remaining set of controls maintain mutually disjoint windows. Clearly, this set is not unique and defining the optimal set of controls to remove is unclear. In contrast, maintaining overlapping controls allow for more precision when matching on time-dependent covariates. For example, seconds after an instant of a RwoT is likely also a RwoT; however, these two RwoTs differ in their run duration, calculated by (2.1). If we were to remove one of these instances to ensure the pool of controls do not overlap, then matching on time-dependent covariates such as run duration and time left in the game may suffer. As a result, we recognize the slight violation of SUTVA so not to introduce bias in the matching procedure.

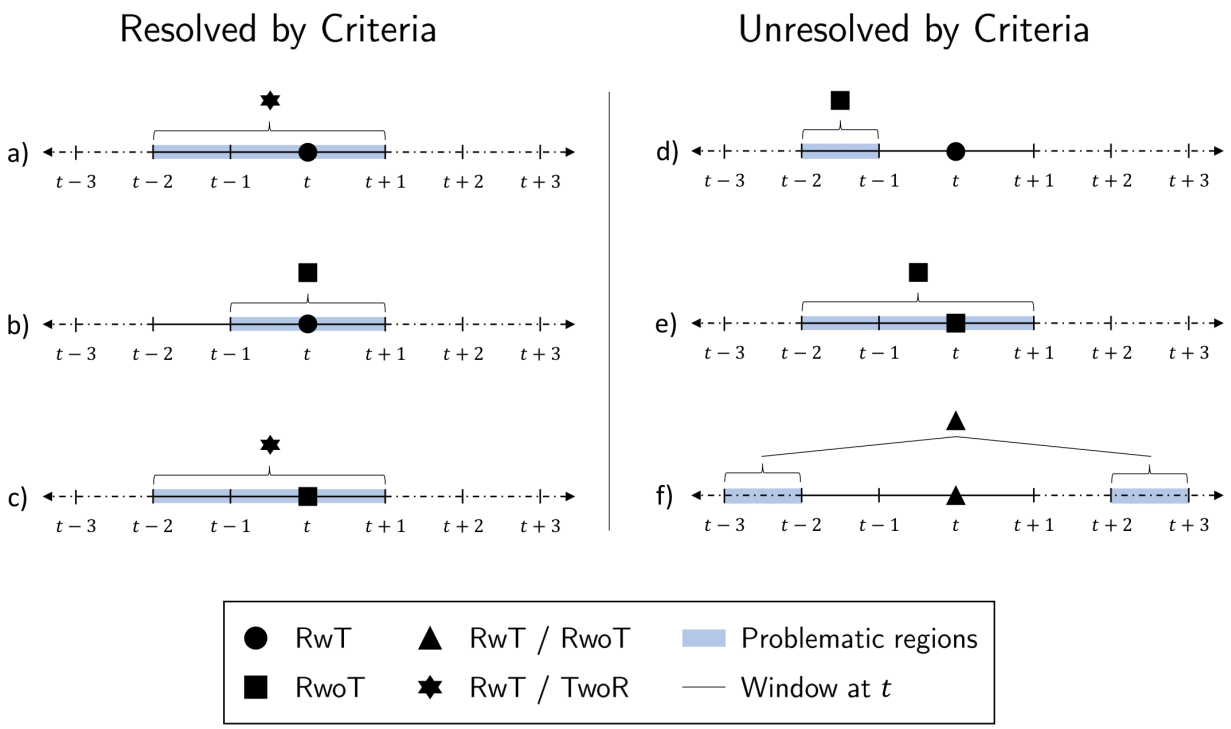


Figure A.1: Possible violations of SUTVA for a fixed play at time t. The criteria resolves issues with interference and hidden variation pertaining to runs with timeouts (RwTs), being the treatment units; however, SUTVA also requires non-interfering runs without timeouts (RwoTs), being the control units. This assumption is relaxed to adequately match of time-dependent covariates such as run duration and time left in the game and to maintain a reasonable sample size for the analysis.

After matching, we investigate the degree of overlapping windows in the matched cohort of controls. Since the matching procedure allows a control to serve as the matched counterfactual for more than one treatment, then the controls are not necessarily unique. Of the 834 matched controls, there are 617 unique controls. Of the 617 unique controls included in the matched cohort, the maximal number of disjoint windows we can construct from these 617 unique controls is 561. So, of the 617 unique controls, at most 561 of them have non-overlapping windows. While we allowed for overlapping windows in the group of controls, after employing the matching algorithm, the number of overlapping windows within the cohort of matched controls is minor.

A.2 Sensitivity to Run Definition

The analysis herein is replicated for various definitions of a run obtained by considering four different run point totals and four different pre-treatment window lengths (see Section 2.5 and Table A.1). Regardless of specific choice in run definition, the estimated causal effect of a timeout is negative and significant, aligning with the presented results. This indicates that the results are robust to the characterization of a run.

Table A.1: Results of the analysis for various definitions of a run. In particular, the inference drawn is relatively the same, illustrating robustness of the results to the characterization of a run.

		N			Results			
Points	Pre-Trt Window	Units	RwT	RwoT	$\widehat{\text{ATT}}$	SE	p	
7	1.5	20,603	2,626	17,977	-0.23	0.04	< 0.001	
7	2.0	34,005	3,694	30,311	-0.25	0.03	< 0.001	*
7	2.5	41,520	4,285	37,235	-0.28	0.03	< 0.001	*
7	3.0	43,821	4,472	39,349	-0.26	0.03	< 0.001	*
8	1.5	6,816	1,090	5,726	-0.38	0.06	< 0.001	
8	2.0	13,677	1,911	11,766	-0.31	0.05	< 0.001	
8	2.5	18,868	2,470	16,398	-0.34	0.04	< 0.001	
8	3.0	21,618	2,745	18,873	-0.26	0.04	< 0.001	*
9	1.5	1,600	325	1,275	-0.37	0.14	0.010	
9	2.0	4,684	834	3,850	-0.35	0.07	< 0.001	
9	2.5	7,547	1,204	6,343	-0.31	0.06	< 0.001	
9	3.0	9,572	1,459	8,113	-0.28	0.05	< 0.001	
10	1.5	370	103	267	-0.51	0.20	0.009	**
10	2.0	1,446	297	1,149	-0.50	0.16	0.001	
10	2.5	2,923	532	2,391	-0.43	0.13	0.001	
10	3.0	4,139	710	3,429	-0.44	0.09	< 0.001	

* match tolerances were relaxed after run time exceeded seven days (see Appendix A.3)

** propensity score model failed to converge

A.3 Genetic Matching Algorithm

The genetic matching procedure used to construct a matched cohort has several arguments that dictate the speed of the process. A few of these arguments include the the population size, wait generation, maximum generation, and the distance tolerance (Sekhon, 2011). Since the theory proving genetic matching yields reasonable solutions is asymptotic in the population size, generational change yields improvement in optimization, and the distance tolerance defines the closeness of a proposed match, the population size and generations are set reasonably high while the tolerance is set reasonably low. In particular, the arguments used are a population size of 8,000, wait generation of 4, max generation of 100, and tolerance of 0.00001. At these values, the optimization routine yields a solution before the generational limits are met. However, for definitions of a run yielding many observations (marked by * in Table A.1), the optimization routine was manually stopped after running for one week. At this point, we relaxed the arguments until the routine reached a run time less than one week. The following arguments are used for such run definitions: population size of 1,000, wait generation of 2, max generation of 15, and tolerance of 0.1. At these relaxed constraints, the optimization routine yielded a solution after nearly three days. On the other hand, the run definitions yielding relatively few observations (marked by ** in Table A.1) induced complications in estimation of the propensity score. When fitting the generalized additive model discussed in Section 2.3.3 to this data set, the procedure failed to converge. We recommend caution in interpreting results demarcated by asterisks.

A.4 Data Preparation and Manipulation

After gathering play-by-play data for the 2017-18 and 2018-19 seasons, data preparation is needed before defining and identifying runs. We start by collapsing multi-row plays (e.g., a foul and free throws) into one row. This is achieved by recording whether a timeout was called when the clock is stopped and retaining the last recorded scoring event. If there were no scoring events, the last event is retained. In exploring the data, we find that some rows are simply empty and others correspond to a game which is clearly misreported, taking on an infeasible score in the

time reported. These rows are removed from the data set. Since overtime is not included in the analysis, rows timestamped after the fourth period are also removed. Furthermore, the data set is limited to instances in time when the score keeper noted some change in the game; however, time is continuous and ought to be treated as such. To this end, we expand the play-by-play data set to contain rows every five seconds. This approach is not suspect since it is safe to assume that the score remains constant if no change in the score is reported. This allows us to more aptly capture runs without timeouts. Lastly, pseudo-plays (dubbed plotting plays) are rows added to the data set before and after any play dictating a change of score in the data set. These plays cannot be considered a unit (removed from consideration by criteria 2 and 3), and are only used to plot the score differential over time, making it a step-wise function to reflect the instantaneous change in score. If a play at time t marks a change in the score difference, then a plotting play is added at $t - \rho$ with the score difference prior to t and another is added at $t + \rho$ with the score difference at t where ρ is arbitrarily close to zero.

After preparing the data set for analysis, every play is evaluated based on (2.2). Any play maintaining a signed run point total of NA is removed. As such, only run plays remain, and evaluating whether a timeout was called during that run is now possible. To quail concerns with SUTVA violations, we introduce criteria (2) through (4) in Section 2.3.2. We start by removing any run play containing the beginning or end of the period in pre-treatment or post-treatment window (criterion 4) before removing any run play containing a timeout in the pre-treatment or post-treatment window (criteria 2 and 3).

Upon addressing concerns with SUTVA violations, we turn to potential issues with positivity by removing any run play with a moneyline greater than 2400 in absolute value. After their removal, we are left with 4,684 run plays which are deemed units for the analysis. Of these run plays, 834 are runs with a timeout (RwTs) and 3,850 are runs without a timeout (RwoTs). More details regarding how many observations are removed with each step can be found in Table A.2.

Table A.2: Number of observations remaining after each step of the data analysis: including data preparation, criteria for defining units, and positivity concerns. In total, there are 4,684 units, 834 of which are runs with timeouts and 3,850 of which are runs without timeouts.

	Observations	Runs with a Timeout	Runs without a Timeout
Gathering Data			
Procure data with nbastatR package in R.	1,144,461		
Data Preparation			
Collapse multi-row plays into one row.	778,828		
Remove misreported games, overtime plays, and empty rows.	617,187		
Discretize time to five second intervals.	2,036,030		
Create plotting plays.	2,813,946		
Invoke Criteria			
Play must be a run.	31,081	1,149	29,932
Windows must be uncensored.	27,340	1,101	26,239
Windows must exclude a timeout.	4,730	838	3,892
Address Positivity			
Consider moneyline less than 2,400 in absolute value.	4,684	834	3,850

A.5 Propensity Score Model

After compiling the pre-treatment covariates listed in Table 2 of Chapter 2, we start by fitting a generalized additive model using all variables but those corresponding to team identity (both the BiT team and the opposing team). Initially, we decided to omit team as a matching covariate since no coach employs a universally consistent timeout strategy (see Appendix A.7). While some

Table A.3: True negative and positive rates (TNR/TPR) and negative and positive predicted values (NPV/PPV) estimated using 70/30% cross-validation and 1,000 Monte Carlo splits. The full model which yields larger estimated propensities for many units yields a marked improvement in the true positive rate with a negligible decline in the true negative rate. Furthermore, the full model yields marked improvements in both the negative and positive predicted values. This implies a betterment in the proportion of classified treatments (controls) which are actually treatments (controls).

	TNR	TPR	NPV	PPV
Restricted Model	0.987	0.089	0.834	0.599
Full Model	0.974	0.185	0.847	0.612

coaches openly claim to strictly adhere to a particular timeout dogma, we've demonstrated this to be false. Our belief is that the act of calling a timeout is predominately driven by the covariates with which we originally constructed the propensity score and conducted the matches. We failed to include team as a matching dimension, feeling an exact match might attenuate the importance of some of the other matching variables. After further investigation, we find that ignoring team identity yields a matched cohort which is significantly unbalanced in terms of the BiT team ($p < 0.001$) and marginally unbalanced in terms of the opposing team ($p = 0.06$). We decide to investigate the inclusion of these variables in the propensity score model. To this end, we construct three nested propensity score models: one including every covariate in Table 2 of Chapter 2, one excluding opposing team, and one excluding both the BiT and opposing team. We conduct two Chi-squared goodness-of-fit tests (Scheipl et al., 2008; Wood, 2013; Young et al., 2011) for nested GAMs to assess the inclusion of BiT team and opposing team, respectively. We find strong evidence that both BiT team ($p < 0.001$) and opposing team ($p < 0.001$) should be included in the model.

Comparing the estimated propensities yielded from the full model (including BiT and opposing teams) and the restricted model (excluding BiT and opposing teams), we notice an increase in the density of larger propensities (see Figure A.2). In general, plays are assigned a higher propensity, indicating more confidence in a called timeout. To investigate the effects of these changes, we employ cross-validation with 1,000 Monte Carlo replicates to estimate the increase (or decrease) in the true positive (negative) rate and the positive (negative) predicted value. For a given replicate,

the units are partitioned into a training and testing set according to a uniformly at random 70/30% split. The restricted and full models are fit using the training set, predictions are made using the testing set, and the metrics are recorded. An average is taken across the 1,000 replicates to estimate the metrics; the results are provided in Table A.3. Inclusion of BiT and opposing teams leads to a marked improvement in the true positive rate as well as the negative and positive predicted values with a negligible decline in the true negative rate. Overall, inclusion of these covariates improves the proportion of correctly classified runs with a timeout and the proportion of classified runs with a timeout (runs without a timeout) which are actually runs with a timeout (runs without a timeout).

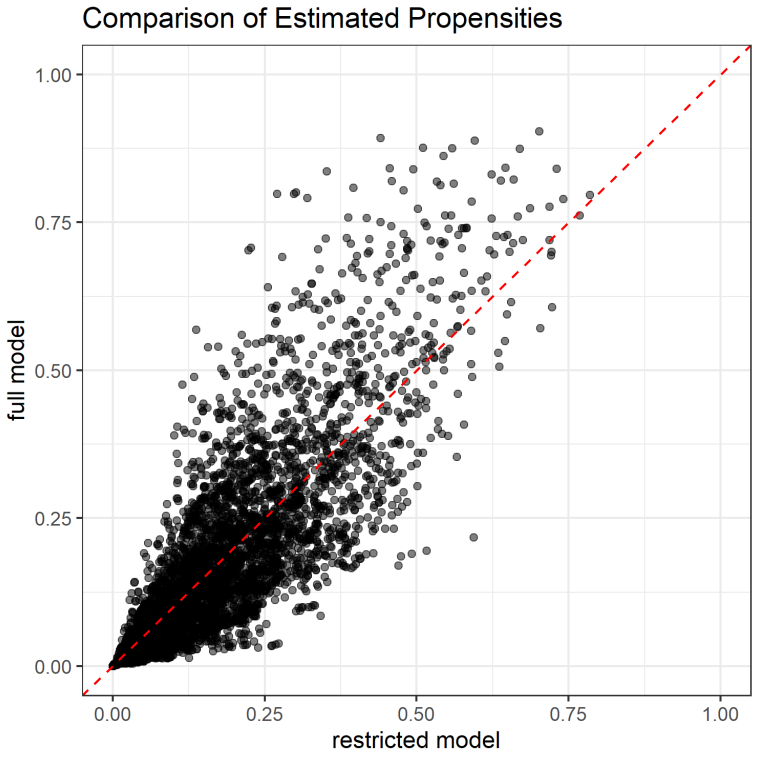


Figure A.2: Estimated propensities for the full model (including BiT and opposing teams) and the restricted model (excluding BiT and opposing teams). After including these covariates, more plays are given a larger propensity score, noted by the larger number of observations above the 45 degree line.

A.6 Matching Variability

Since the genetic matching algorithm provided in the R package `Matching` is non-deterministic, the matching algorithm was initialized at 20 different random seeds and the ATT was estimated for each of the resulting matched cohorts to explore the variability of the estimator. Figure A.3 shows the ATT estimates and the corresponding 95% confidence intervals for the twenty matched cohorts. The random seed associated with the estimated ATT closest to the average of the twenty estimated ATTs was used for the body of Chapter 2. We observe here that the empirical standard deviation of the estimates closely resembles the theoretically derived Abadie-Imbens standard error, which explicitly accounts for the uncertainty of the matching procedure.

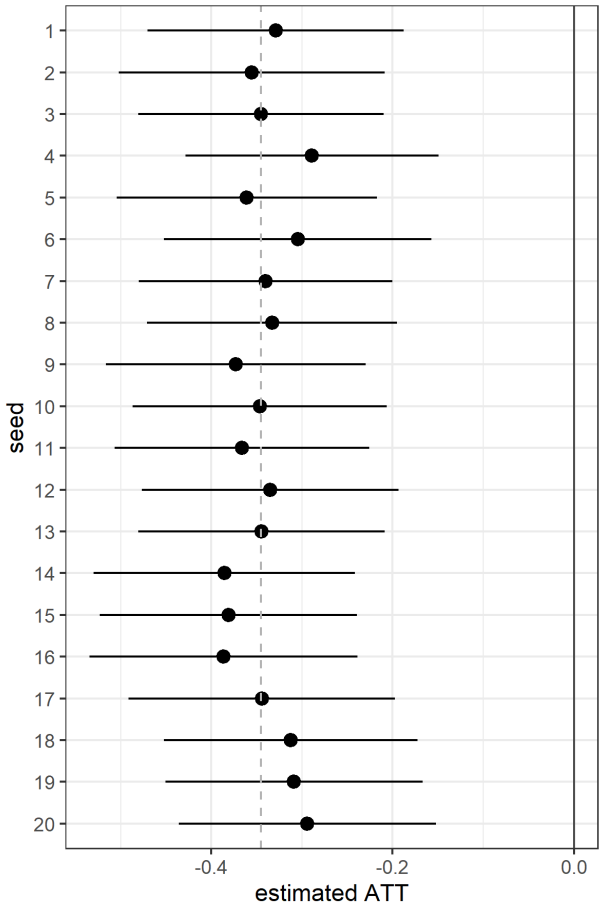


Figure A.3: Estimated ATT and 95% confidence interval for 20 runs of the matching algorithm. The dashed, vertical line represents the estimate used in this chapter. Each realization of the matching algorithm produced consistent results.

A.7 Treated and Control Franchise Frequency

To investigate each franchise's strategy in responding to an opposing run (i.e., as the BiT team), we count the number of runs with and without a timeout before and after matching for each franchise, provided in Figure A.4. We find that no franchise adheres to one and only one strategy when faced with an opposing run. In fact, the teams with the fewest recorded number of runs with a timeout called a timeout at only roughly half the frequency as the team with the most recorded number of runs with a timeout, 19 versus 41, respectively. Since no franchise adheres to one and only one strategy, implying any franchise can viably serve as a matched control, positivity is likely not an issue in estimation of the franchise effects.

Furthermore, since each franchise called a timeout during an opposing run at least 19 times, the franchise-specific effects of a timeout can be estimated; however, some franchises suffer from relatively small sample sizes when compared to other franchises. We take this into consideration by reporting the standard errors alongside the point estimates for the team effects in Figure 8 of Chapter 2. The standard errors, as expected, ingrain the inequity in the number of observations by franchise where teams with more observations (Chicago Bulls) have much smaller standard errors than those teams with fewer observations (San Antonio Spurs).

After matching, we tabulate instances for which each franchise serves as the treatment (BiT team with a timeout) and matched control (BiT team without a timeout) in Figure A.5. From this figure, we conclude that no franchise is matched to a concernedly small number of franchises. The sparsity of the plot is largely a function of the small sample size for several franchises, considering that slightly more than half of the franchises (19) have fewer runs with timeouts than the number of franchises (30) in the National Basketball Association (NBA).

A.8 Covariate Balance

To assess covariate balance after matching, distributions of the covariates for the treated and control units were created and shown in Figure A.6. Before matching, the distribution of the covariates across treatment groups appear relatively similar for most covariates; however, there ap-

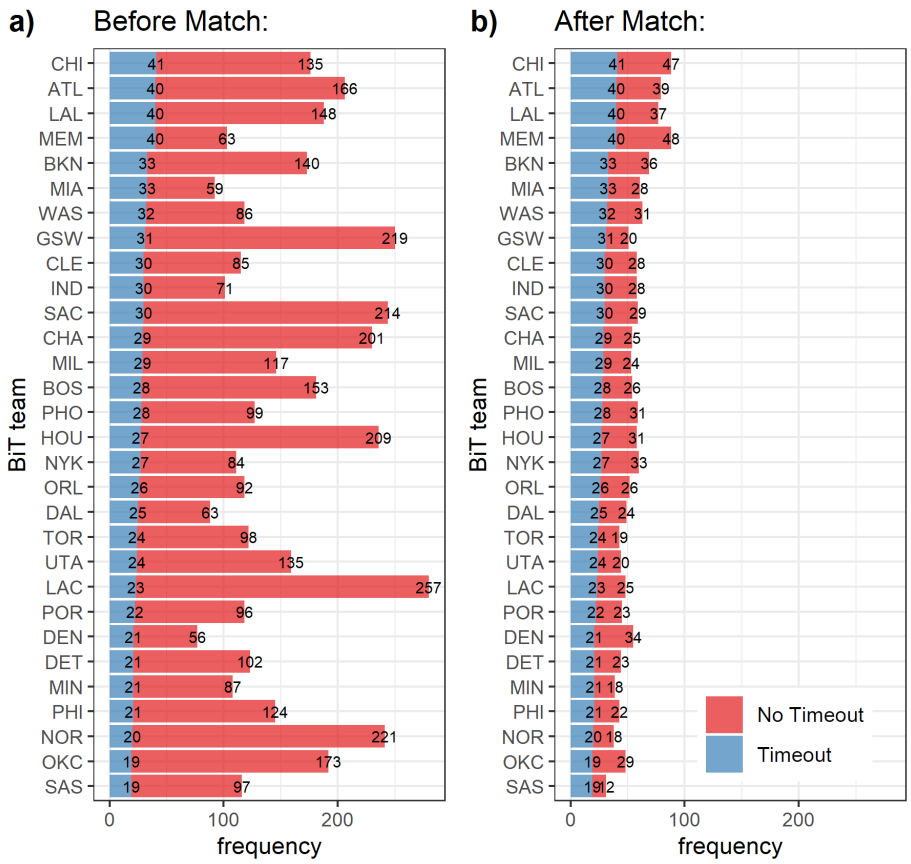


Figure A.4: The blue horizontal bars indicate the number of runs with a timeout (RwTs) for a given franchise. The red horizontal bars indicate the number of runs without a timeout (RwoTs) for a given franchise. Each of these runs adhere to the criteria and are thus considered a unit. The matched set allows for estimation of the causal estimand after invoking the assumptions therein. There is no evidence of issues with positivity in estimating franchise specific effects since all teams exhibit instances of runs with and without timeouts.

pear to be discrepancies in the BiT team, opposing team, win probability, time left, and possession across treatment group. Most notably, the distribution of the estimated propensity scores between the treatment groups was largely different. Aside from covariate imbalance, there appears to be a potential violation of positivity in the moneyline. After removing plays with a moneyline larger than $2,400$ in absolute value and invoking the matching procedure, the distribution of the estimated propensity scores appears largely balanced. Many of the discrepancies in the distributions of the covariates noted earlier seem to no longer exist, such as that in the win probability. Formal hypothesis testing was applied to each of the covariates to verify these visual interpretations.

Table A.4: Unadjusted p-values from hypothesis testing to assess for a discrepancy in the distribution of covariates before and after matching. Before matching, every covariate except for week in season exhibits imbalance between treatment groups. After matching, every covariate exhibits balance between treatment groups. In each case, multiple comparison correction is preformed to control the false discovery rate at 0.05, according to Benjamini and Hochberg (1995).

	p-value	
	Pre-Match	Post-Match
GAM Propensity Score	< 0.001	0.038
Big Trouble (BiT) Team	< 0.001	0.989
Opposing Team	0.046	0.568
Run Point Total	0.002	0.082
Run Duration	< 0.001	0.842
Time Left	< 0.001	0.502
Win Probability	< 0.001	0.064
SSD at BOR	< 0.001	0.176
SSD at EOR	0.002	0.138
Possession Indicator	< 0.001	0.912
Home Indicator	0.037	0.619
Week in Season	0.297	0.493
Over/Under	< 0.001	0.145
Spread	0.003	0.374
Moneyline	0.004	0.112

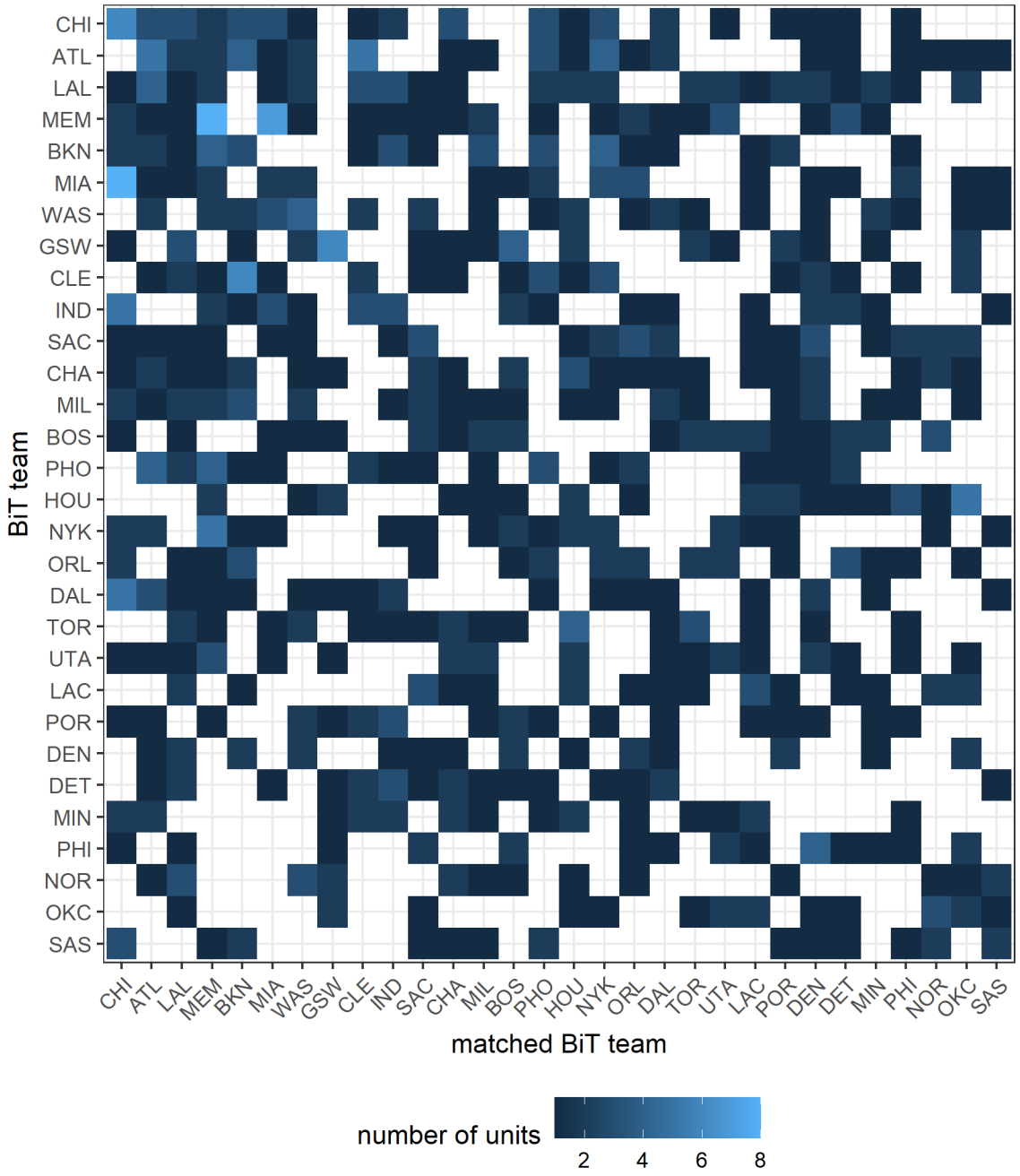


Figure A.5: The number of matched controls by franchise for each franchise treated unit. The $(1, 2)$ fill of the tile plot, for example, illustrates the number of times that an Atlanta Hawks run without a timeout was matched to a Chicago Bulls run with a timeout. No franchise is matched to a concernedly small number of franchises. The rows and columns of the tile plot have been arranged by the frequency with which that franchise serves as a matched control. That is, the New Orleans Pelicans were most often matched as a control (53), whereas the Denver Nuggets, Phoenix Suns, and Toronto Raptors were tied for least often matched as a control (14).

To check for discrepancies in the distributions of discrete and continuous covariates across treatment groups, bootstrapped Kolmogorov-Smirnov tests were applied. For binary variables, t-tests were used, and chi-squared tests were used for categorical variables. The raw p-values for each of these tests before and after matching are provided in Table A.4. Before matching, a discrepancy between the distributions of covariates associated with treatments and controls was identified in all covariates and the estimated propensity scores except for week in season which appeared sufficiently balanced. After matching, there was no evidence of discrepancy in covariate distributions for any of the covariates or the estimated propensity score. All covariates appear sufficiently balanced after matching. In each case, multiple comparison correction was preformed to control the false discovery rate at 0.05, according to Benjamini and Hochberg (1995).

To assess covariate balance when conditioning on franchise, we perform hypothesis testing to assess potential discrepancies in the distribution of covariates between the treatment units (runs with a timeout) and control units (runs without a timeout) for each of the fifteen covariates and each of the thirty franchises. In all, 450 hypothesis tests were performed (15 covariates by 30 franchises), so a multiple comparison correction was performed to control the false discovery rate at 0.05. We attain an unadjusted p-value for each covariate/franchise combination (see Figure A.7). There are seven covariate/franchise combinations with an unadjusted p-value less than 0.05 (outlined in red in Figure A.7) with the smallest being $p = 0.004$, corresponding to possession with the New Orleans Pelicans. After invoking the multiple comparison correction (Benjamini and Hochberg, 1995), there is no evidence for distributional discrepancy between matched sets of covariates for any of the franchises. This suggests it is reasonable to assume covariate balance when conditioning on the treated team's identity (i.e., the BiT team's identity) and to proceed with estimating the franchises' respective causal effects.

A.9 Data and Code

Data and code to reproduce the results in the chapter are available in the following public GitHub repository: https://github.com/ConGibbs10/nba-causal.

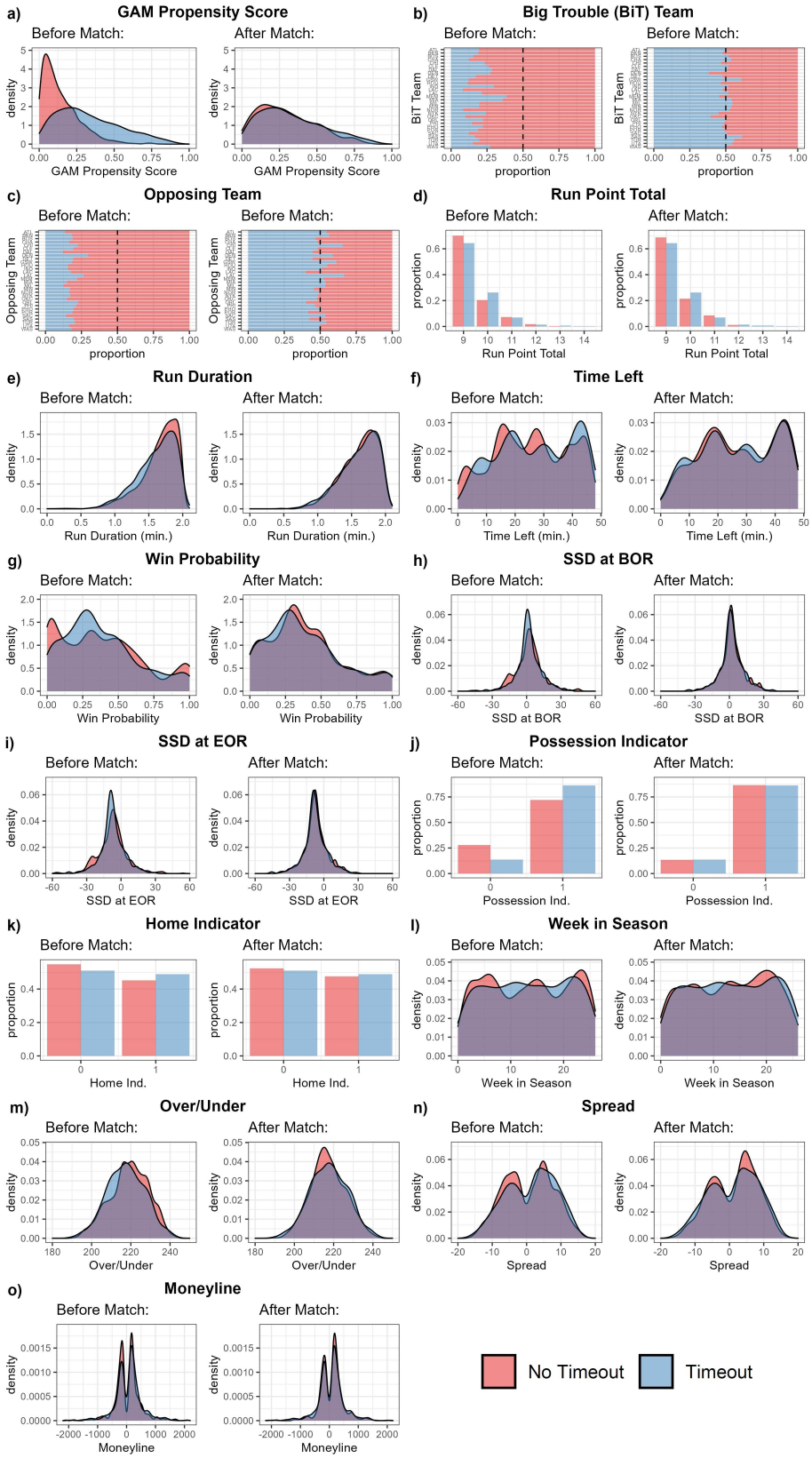


Figure A.6: Distribution of the covariates for each treatment group before and after matching. Visually, matching appears to yield distributions of covariates which are similar across treatment group.

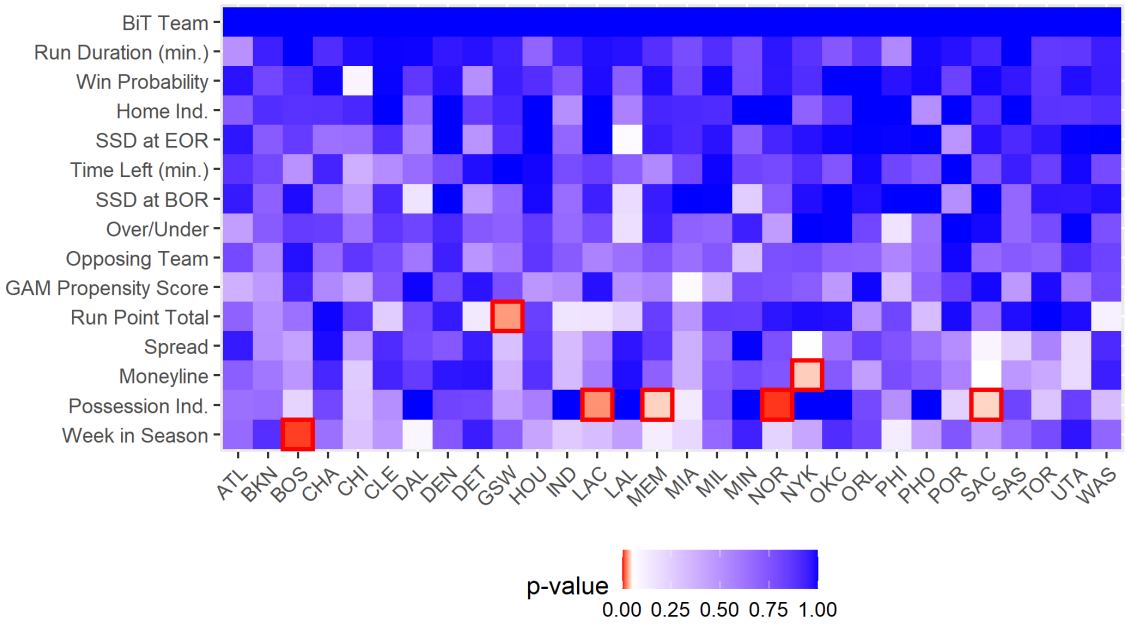


Figure A.7: Hypothesis testing was conducted on the matched samples' covariates for each franchise. Unadjusted p-values for each covariate/franchise are shown. Seven p-values (outlined in red) maintained an unadjusted p-value less than 0.05, the minimum of which was $p = 0.004$, corresponding to possession with the New Orleans Pelicans. After multiple comparison correction to control the false discovery rate at 0.05, no p-values are statistically significant. Hence, it is reasonable to assume the covariates are sufficiently balanced when conditioning on the BiT team's identity.

Appendix B

Identification of Clusters Containing Undiscovered Relations

B.1 Clustering Methods

Methods considered for this study need be scalable to the large number of nodes and edges in $\mathcal{G}_\tau$. Furthermore, to capture a broad range of communities, methods should carry different objectives or make different assumptions about the underlying community structure. We focus on disjoint and overlapping methods. The disjoint clustering methods considered include Greedy modularity maximization (Clauset et al., 2004), a Louvain style heterogeneous modularity maximization ZCmod (Zhang and Chen, 2020), Walktrap (Pons and Latapy, 2006), and Infomap (Rosvall et al., 2009; Rosvall and Bergstrom, 2007b). While each method seeks to partition the node set, the objective function to optimize differs across each method, leading to fundamentally different clusters. CESNA (communities from edge structure and node attributes) (Yang et al., 2013) and ZCmod (Zhang and Chen, 2020) is an overlapping clustering method which leverages both edge density and node type to form clusters.

Unlike the Greedy, Walktrap, and Infomap methods, CESNA and ZCmod distinguishes between genes and phenotypes, treating the biological network as a heterogeneous graph and identifying communities which are topologically dense. However, CESNA assumes community members' should share common node types whereas ZCmod assumes community members' have differing node types. That is, CESNA assume a homogeneous community structure whereas ZCMod assumes heterogeneous community structure. A summary of each method is included below:

Greedy Proposed in Clauset et al. (2004), this clustering method seeks to optimize a modularity score. In particular, a partition creating the largest disparity between the fraction of edges internal to the partition and the expected fraction from a random graph with the same degree sequence is

desired. Since maximizing this quantity is NP-complete in the strong sense (Brandes et al., 2006), a greedy heuristic is used to find reasonable clusters. To start, each node is considered a community. At each iteration of the algorithm, two communities that contribute maximum positive value to the global modularity score are merged. This process continues until no such increase in modularity is possible. The estimated complexity of this method on sparse networks is $\mathcal{O}(nlog^2n)$ where n denotes the number of nodes in the network. Greedy is implemented in the NetworkX package (Hagberg et al., 2008) of Python.

Walktrap Walktrap was introduced in Pons and Latapy (2006) as a means for clustering a graph using random walks, positing that short random walks tend to remain within a community. To start, each node is considered a community. Distances between communities are computed via random walks, and communities are merged such that there are shorter walks within a community and larger walks between communities. This process is repeated $n-1$ times implying a complexity of $\mathcal{O}(mn^2)$ where m denotes the number of edges in the graph, or $\mathcal{O}(n^2logn)$ for sparse graphs (Xie and Szymanski, 2011). Walktrap is implemented in the CDlib package (Rossetti et al., 2019) of Python.

Infomap Rooted in information theory, Infomap discovers communities by minimizing the description length of an information flow on a graph, a variant of a coding problem (Rosvall and Bergstrom, 2007a). Information flows are measured across a network using random walks where groups of nodes for which information flows easily "can be aggregated and described as a single well connected module" (Rosvall and Bergstrom, 2007b). The authors demonstrate these modules are synonymous to communities. The map equation, introduced in Rosvall et al. (2009), provides the theoretical basis for the Infomap algorithm, describing how well information about the original network is transferred through a given network partition. Mukherjee et al. (2013) estimates that the complexity of infomap is $\mathcal{O}(m)$. This method is available through the Infomap package (Edler et al., 2022) in Python.

CESNA Distinct from its competing methods, CESNA (communities from edge structure and node attributes) discovers communities using the structure of the edges within a graph and the available node attributes. In particular, CESNA treats the biological network as a heterogeneous network, distinguishing genes from phenotypes rather than treating them fundamentally the same. The method posits that a graph arises from its nodes' attributes and its nodes' community structure before inferring the latent community structure via maximum likelihood estimation (Yang et al., 2013). Discovered communities should be topologically dense and maintain similar node type. One iteration of CESNA has an estimated complexity of $\mathcal{O}(m + nk)$ where k is the number of node attributes. For this study, the node type (i.e., gene or phenotype) is the only node attribute considered implying a computational complexity of $\mathcal{O}(m+n)$. This method is implemented using the Stanford Network Analysis Platform (SNAP) (Leskovec and Sosič, 2016).

ZCmod Like CESNA, ZCMod treats the network as a heterogeneous graph. ZCmod seeks to optimize a modularity score defined as a function of the topology with respect to the node types, and identifies communities which are topologically dense and maintain different node types. ZCmod uses a Louvain style modularity maximization (De Meo et al., 2011). To start, each node in the network is assigned to its own cluster: a simple partition. In the first phase of the method, each node is moved to the cluster which results in the greatest increase in modularity, and this process is repeated no such increase in modularity is feasible. In the second phase of the method, an aggregate network is created. The clusters from the previous stage represent the nodes of the aggregate network. Edges between nodes within and between clusters represent self loops and weighted edges, respectively, in the new aggregate network. At this point, stage one and two are iteratively applied until no improvements to modularity are possible. One iteration of ZCmod has a complexity of $\mathcal{O}(m)$ where m is the number of edges; hence, the worst case scenario is $\mathcal{O}(m^2)$. This method is implemented on GitHub (Gibbs, 2022).

Clusters in real networks tend to exhibit hierarchical structure such that larger clusters are assumed composed of smaller clusters which can be further divided. For this study, we use an agglomerative, hierarchical clustering method based on the distance between node pairs to subdi-

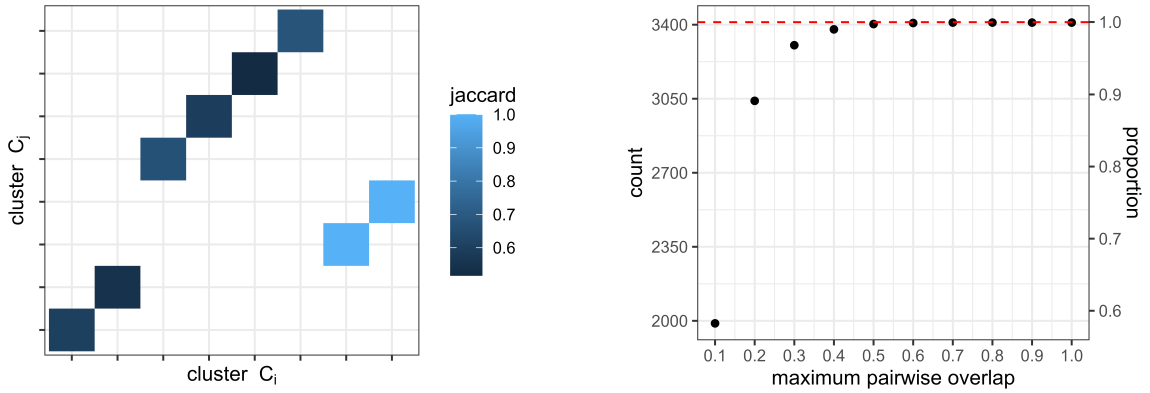
vide clusters attained from the network clustering methods into "subclusters" of manageable size. Paris (Bonald et al., 2018), the hierarchical method of choice, produces a dendrogram which can be cut to produce subclusters. The dendrograms are cut such that the largest resulting subcluster maintains fewer than 100 members. The Paris algorithm is described below:

Paris Paris is an agglomerative, hierarchical clustering method based on node pair sampling (Bonald et al., 2018). In particular, the method proposes a distance measure which compares the joint probability of sampling a pair of nodes, say i and j, to the product of the marginal probabilities of sampling i and j. In particular, two nodes are relatively close if the probability of sampling j given that i has been sampled is large compared to the probability of sampling j. To start, each node is assigned to its cluster. Recursively, the two closest clusters are combined resulting in a dendrogram. The algorithm is a modification of the Louvain algorithm (Blondel et al., 2008) where the iterative step is replaced by a single merge, implying a complexity of $\mathcal{O}(m)$ where m denotes the number of edges.

B.2 Pairwise Overlap of Clusters

When the clusters used to create the training data, i.e., $\mathbf{X}^{\intercal}$, overlap significantly with each other, it is reasonable to be concerned about independence between the units of the training data and in the cross-validation folds. A lack of independence among clusters could engender an overly complex DART model since dependent units could be split over the training and validation folds when optimizing hyperparameters. In the following paragraphs, we identify the degree of pairwise overlap among the clusters used to create the training data and consider the complexity of the resulting optimal model should high overlapping clusters be removed from the training data. We investigate these concerns by considering the results of ICCUR$(\mathcal{G}_{20}, \mathcal{G}_{21})$ where the ICCUR pipeline is provided in Algorithm 3.1. For this fit, the training data are created from biological and topological cluster features (see Table 3.3) of 3412 small heterogeneous clusters identified from $\mathcal{G}_{20}$ (see Table 3.5).

We first demonstrate clusters are largely more dissimilar than similar; that is, there is little substantial pairwise overlap between the clusters used to create the training data. To measure the degree of overlap among the 3412 clusters identified from $\mathcal{G}_{20}$, we compute the Jaccard similarity measure between all $\binom{3412}{2}$ pairs of clusters. Measuring the overlap between clusters $\mathcal{C}_i$ and $\mathcal{C}_j$, the Jaccard similarty measure is denoted $J(\mathcal{C}_i, \mathcal{C}_j) = |\mathcal{C}_i \cap \mathcal{C}_j| / |\mathcal{C}_i \cup \mathcal{C}_j|$, where $J(\mathcal{C}_i, \mathcal{C}_j) = 0$ implies the clusters are disjoint, $J(\mathcal{C}_i, \mathcal{C}_j) = 1$ implies the clusters are equal, and importantly, $J(\mathcal{C}_i, \mathcal{C}_j) < 0.5$ implies the clusters are more dissimilar than similar. Of the 3412 clusters, only 9 pairs of clusters have an overlap larger than 0.5, as seen in Figure B.1a. In this case, removing nine clusters results in a pairwise overlap less than 0.5; that is, by removing only 0.26% of clusters, the training data are constructed from clusters which are more dissimilar than similar. The number and proportion of clusters satisfying varying maximum pairwise overlap are provided in Figure B.1b.



(a) Clusters with large pairwise overlap

(b) Clusters satisfying varying maximum pairwise overlap

Figure B.1: Panel (a) provides the pairwise overlap between all pairs of high-overlap clusters. Only nine pairs have a Jaccard similarity measure greater than 0.5, indicating that the vast majority of clusters in the training set are more dissimilar than similar. Panel (b) provides the number and proportion of clusters remaining when clusters are eliminated sequentially to preserve a pairwise overlap less than some specified maximum. Only nine clusters (0.26%) need be removed to ensure that every cluster in the training set is more dissimilar than similar.

We now demonstrate that the optimal model's complexity remains consistent after training on clusters more dissimilar than similar. We remove nine clusters from the high-overlap pairs in Figure B.1a. To decide which cluster from the high-overlap pair is removed, we consider the

vector of pairwise Jaccard measures associated with each cluster and remove the cluster with the largest mean pairwise overlap. If the mean pairwise overlap associated with each cluster are equal, we randomly choose one cluster for elimination according to a Bernoulli random variable with probability parameter 0.5. We retrain the DART model over the same grid of hyperparameter values using the 3403 clusters with pairwise overlap less than 0.5, according to Section 3.3.1.

The hyperparameter combinations deemed optimal in Chapter 3 using the full set of training data and after removal of high-overlap clusters are provided as option A and B in Table B.1, respectively. After retraining, the average cross-validated RMSE for option B is 0.1907 and option A is 0.1909, so both hyperparameter combinations have effectively the same out-of-sample performance. In both cases, the model is an ensemble of deep trees with a small minimum loss reduction; however the moderately large learning rate and relatively few boosting iterations in option A, or the very small learning rate and high dropout rate in option B, combat the complexity of each individual tree. Hence, the complexity of the optimal DART model is a result of better out-of-sample performance and not a manifestation of highly dependent units present in the training and validation folds.

Table B.1: The hyperparameters of the DART model were optimized in two different ways: option A using the full set of training data in Chapter 3 and option B with consideration for cluster overlap in the training set. When accounting for overlap, both option A and B had similar average cross-validated loss, indicating that they had nearly equivalent out-of-sample performance.

			Optimal	
Parameter	Alias	Range	A	B
learning rate	`eta`	0 - 0.1	0.043	0.003
minimum loss reduction	`gamma`	0 - 30	< 0.001	< 0.001
maximum number of trees	`nrounds`	2 - 2000	141	1566
maximum tree depth	`max_depth`	1 - 15	12	14
subsample ratio	`subsample`	0.1 - 1	0.38	0.46
dropout rate	`rate_drop`	0 - 1	0.11	0.66
skip rate	`skip_drop`	0 - 1	0.86	0.78

B.3 Cluster Features

The distribution of the potential for future discovery and each cluster feature over the years under study is provided in Figure B.2. The distribution of each feature remains relatively stable over time, possibly explaining why the specification of the DART model was relatively similar year over year. Furthermore, we see that the observed PFD for each measurable year is largely non-zero, indicating the ICCUR pipeline's ability to identify clusters ripe for future discovery. The first nine features (from cell type specificity to the sum of gene's predicted loss of function) are biologically inspired features, deemed possibly predictive of the clusters' potential for future discovery. The later features are network inspired features meant to discriminate clusters. These features can measure things like how well-knit a cluster is (average embeddedness and conductance), how much of a hub-and-spoke topology the cluster has (hub dominance), and how cohesive the cluster is (triangle participation rate). By differentiating clusters, we hope to establish associations between a cluster's topology and its potential for discovery—associations we can capture with flexible modeling tools.

While complicated at times, the underlying relationships among the previous year's clusters' features and its potential for future discovery are those used to estimate the degree to which contemporary clusters may embody future G2P relations via a DART model. While the DART model is shown to be reasonably predictive, it is not very interpretable—a classical trade off with statistical models. Hence, to better understand what each feature contributes to model predictions, we leverage SHAP (SHapley Additive ExPlanations) values (Lundberg and Lee, 2017). Rooted in game theory, SHAP values provide a means of reverse engineering the output of any predictive algorithm by quantifying the marginal contribution each feature has on a model prediction. For a given prediction, a feature's negative SHAP value indicates, holding all else constant, the value of the feature decreases the model's prediction; a positive SHAP value indicates the value of the feature increases the model's prediction. The final model's prediction is then a compilation of each

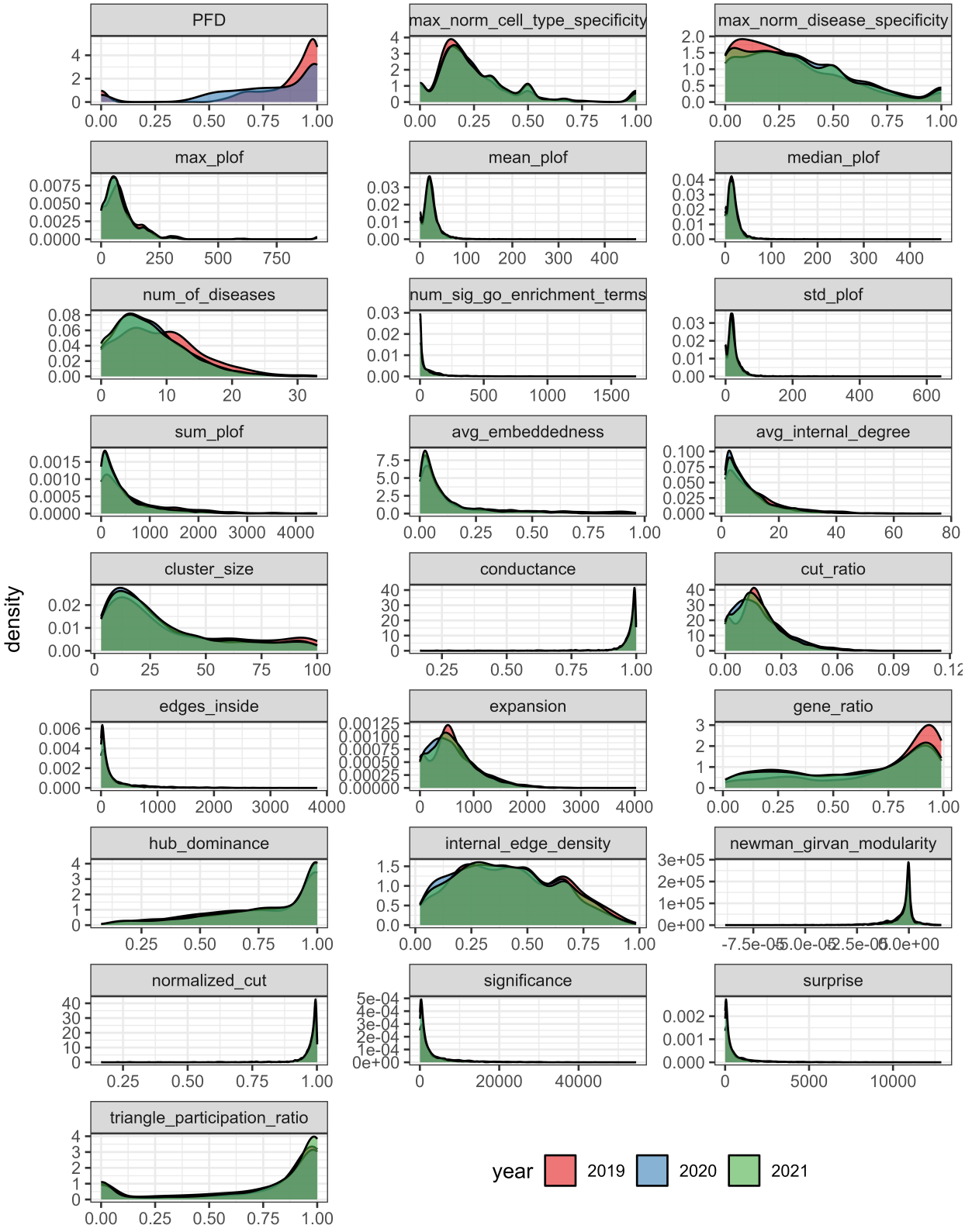


Figure B.2: Distribution of cluster features over the years under study. The relationship between the year's features are used to estimate the potential for future discovery (PFD). The distribution of each feature remains relatively stable over time.

features' marginal contributions, as measured by the SHAP values. For an aggregate measure of feature importance, one can consider the mean absolute SHAP value, provided in Table B.2 for the 2019 clusters where larger values indicate the feature has a larger marginal impact or contribution to model predictions. To better understand the relationship between a feature and the model predictions, one can consider the relationship between a feature's support and the corresponding SHAP values.

These relationships are summarized in Figure B.3 which illustrates the value of a feature and the impact of that feature's value on the model's predictions in 2019. For each feature, there are as many points as there are clusters in 2019 where the points are colored according to the feature's value. The feature values have been standardized to a scale between zero and one where zero (one) indicates the minimum (maximum) observed. The features are sorted according to the mean absolute SHAP value (which adorn the axis). Features with a larger mean absolute SHAP value (e.g., the gene ratio, hub dominance, and cluster size) are marginally more highly influential to the model predictions. Unsurprisingly, clusters which are more balanced in the number of genes and phenotypes (i.e., maintain a gene ratio near 0.5) tend to have a higher PFD. Furthermore, smaller clusters tend to embody a higher PFD. Interestingly, clusters with a large measure of hub dominance tend to have larger SHAP values, indicating that clusters with a hub-and-spoke topology tend to have a higher PFD. While this exploration is not intended to be inferential, it helps unveil the associations driving the model predictions.

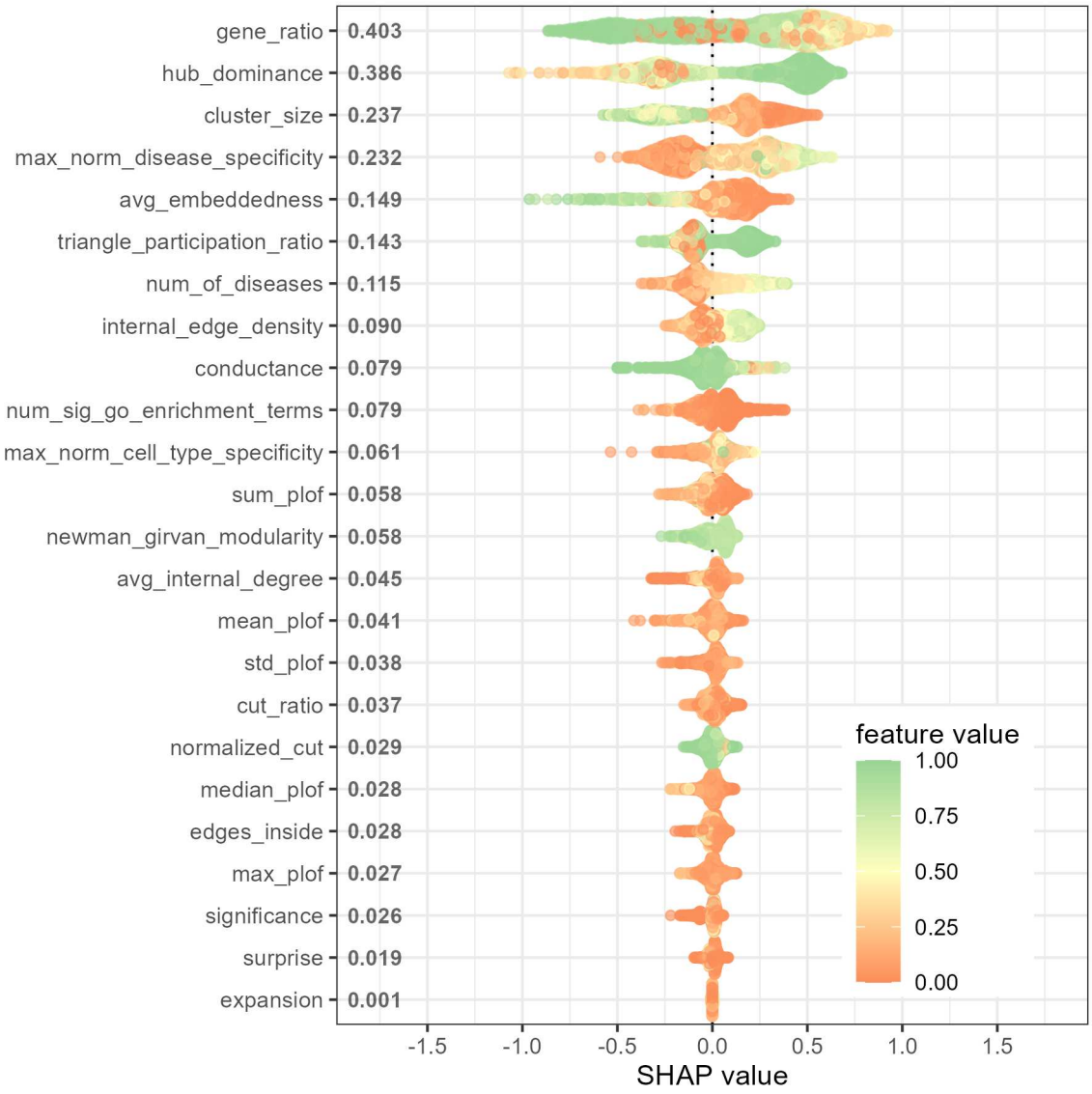


Figure B.3: Relationship between the value of each cluster feature and the marginal impact of the features' value on the 2019 model predictions. A positive (negative) SHAP value indicates that the value of the feature marginally increased (decreased) the estimated PFD for that particular cluster. Features with larger mean absolute SHAP value tend to have a higher marginal contribution to the model predictions.

Appendix C

Extracting Communities from Heterogeneous Networks

C.1 Heterogeneous Degree Configuration Model (HDCM)

Assume $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denotes an observed, heterogeneous network with the collection of heterogeneous degree sequences $\boldsymbol{\mathcal{D}}$. The "Heterogeneous Degree Configuration Model" (HDCM) section of Chapter 4 provides a generalization of the degree configuration model (DCM) capable of preserving not only the degree of each node but the heterogeneous degree sequence of each node. Using the corresponding notation and assumptions provided in the "Heterogeneous Networks" and "Heterogeneous Degree Configuration Model" sections of Chapter 4, one can efficiently conduct the stub matching process through a constrained permutation of the node labels in the edge multiset $\mathcal{E}$. Furthermore, we provide additional details justifying how the HDCM is simply a generalization of the DCM, equivalent to the DCM when $K = 1$.

C.1.1 Efficient Generative Process for the HDCM

An efficient, generative process for HDCM$(\mathcal{T}, \boldsymbol{\mathcal{D}})$ is depicted in Figure C.1 and takes advantage of the existing edge set $\mathcal{E}$. The process begins by partitioning the edge multiset into $K + K(K-1)/2$ subsets according to the observed adjacent nodes' type: $\mathcal{E} = \bigcup_{1 \le k \le l \le K} E^{[kl]}$ where

$$E^{[kl]} = \{\{u, v\} \in \mathcal{E} : u \text{ is of type } k, v \text{ is of type } l\}.$$

That is, the edge multiset $E^{[kl]} \subseteq \mathcal{E}$ is the set of all $m^{[kl]} = |E^{[kl]}|$ undirected links between nodes of type k and l. To construct a network with the same collection of heterogeneous degree sequences, we permute the node labels of $E^{[kl]}$ to construct a new edge multiset, denoted $\widetilde{E}^{[kl]}$, for

each $1 \leq k \leq l \leq K$. The process for attaining $\widetilde{E}^{[kl]}$ is dependent on whether $k = l$ or $k \neq l$, so we consider these cases.

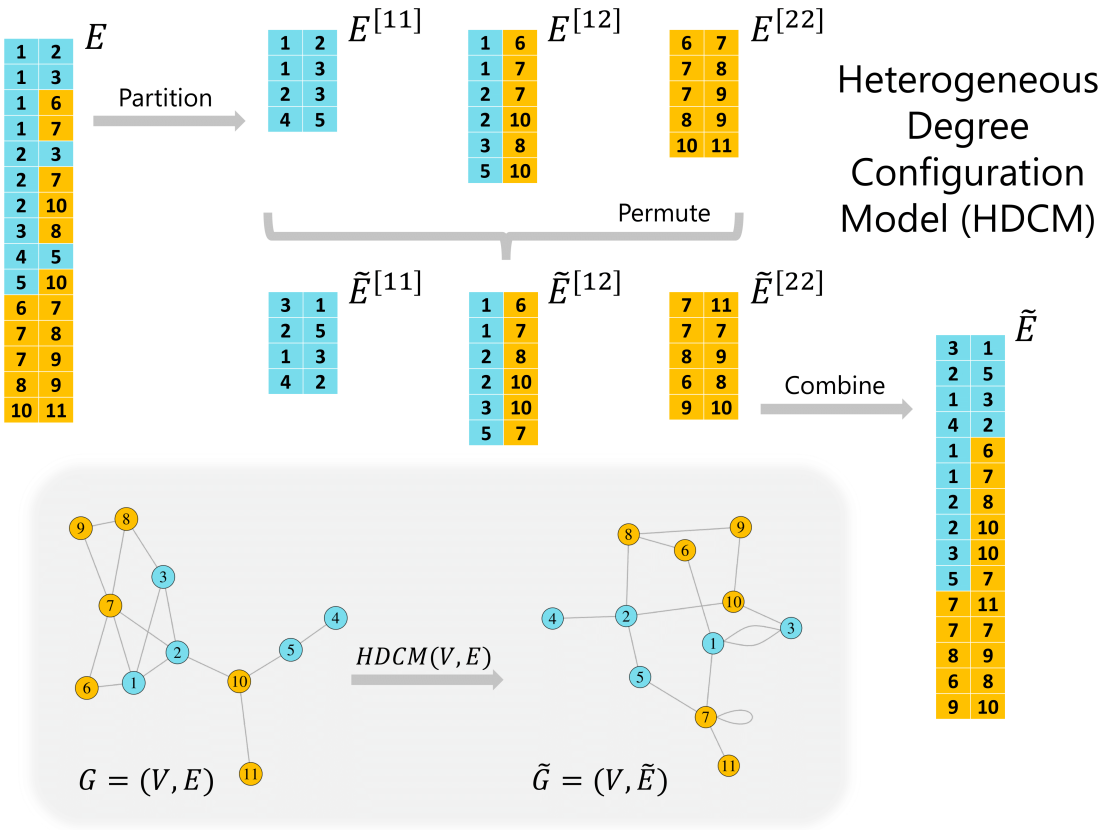


Figure C.1: A schematic of the heterogeneous degree configuration model: a model of a random network maintaining the same collection of heterogeneous degree sequences, $\boldsymbol{\mathcal{D}}$, as an observed network, $\mathcal{G}$. The process begins by partitioning the edge multiset, $\mathcal{E}$, according to adjacent nodes' type. Upon completion, node labels are permuted (or rearranged) within the elements of the partition to preserve the respective type k degree of each node. The new edge multisets are combined into one edge multiset, $\widetilde{\mathcal{E}}$, where a random network $\widetilde{\mathcal{G}} = (\mathcal{V}, \widetilde{\mathcal{E}})$ has the same degree collection $\boldsymbol{\mathcal{D}}$.

Case 1: Attaining $\widetilde{E}^{[kl]}$ for $k = l$. For same type connections, the idea is to permute all node labels and create adjacent pairs by grouping the reordered node labels two by two. This process will preserve the number of same type connections for each node. To detail formally, let $E^{[kk]}$ be structured as an $m^{[kk]}$ by 2 matrix where each row vector provides an observed edge between two type k nodes. Note, the order of the rows and columns of $E^{[kk]}$ carry no specific meaning but is used for bookkeeping. Furthermore, if $\mathcal{G}$ is a multigraph then there exist duplicate rows.

Let $\boldsymbol{w} = [w_1, \ldots, w_{2m^{[kk]}}]$ denote a row vector (with $2m^{[kk]}$ entries) attained by concatenating the transpose of the column vectors of $E^{[kk]}$. Let $\tilde{\boldsymbol{w}}$ denote a permutation (or rearrangement) of the entries of $\boldsymbol{w}$. The random configuration of k to k connections is represented similarly as an $m^{[kk]}$ by 2 matrix, denoted $\widetilde{E}^{[kl]}$, where the ith row vector of $\widetilde{E}^{[kl]}$ is given by $[\tilde{w}_i, \tilde{w}_{i+1}]$ for $i \in \{1, \ldots, m^{[kk]}\}$.

Case 2: Attaining $\widetilde{\boldsymbol{E}}^{[\boldsymbol{kl}]}$ for $\boldsymbol{k < l}$. For between type connections, the idea is to isolate and permute node labels of one type, say l, before creating adjacent pairs by joining the permuted node labels with those node labels of type k. To detail formally, let $E^{[kl]}$ be structured as an $m^{[kl]}$ by 2 matrix where each row vector provides an edge between a type k node and a type l node, respectively. That is, the first column vector of $E^{[kl]}$, denoted $\boldsymbol{z}$, contains only type k nodes and the second column vector of $E^{[kl]}$, denoted $\boldsymbol{w}$, contains only type l nodes. Note, the order of the rows of $E^{[kl]}$, however, carry no specific meaning and is used for bookkeeping. Furthermore, if $\mathcal{G}$ is a multigraph then there exist duplicate rows. Let $\tilde{\boldsymbol{w}}$ denote a permutation (or rearrangement) of the entries of $\boldsymbol{w}$. The random configuration of k to l connections is represented similarly as an $m^{[kl]}$ by 2 matrix, denoted $\widetilde{E}^{[kl]}$, where the first and second columns of $\widetilde{E}^{[kl]}$ are $\boldsymbol{z}$ and $\tilde{\boldsymbol{w}}$, respectively.

The newly attained edge multisets $\widetilde{E}^{[kl]}$ (represented as $m^{[kl]}$ by 2 matrices) for each $1 \leq k \leq l \leq K$ are combined rowwise (in any order) to attain an m by 2 matrix, denoted $\widetilde{\mathcal{E}}$, where $m := |\widetilde{\mathcal{E}}| = |\mathcal{E}|$. The random network $\widetilde{G} = (\mathcal{V}, \widetilde{\mathcal{E}})$ has the same collection of heterogeneous degree sequences as the observed network $\mathcal{G}$. That is, each node in $\widetilde{\mathcal{G}}$ has the same type k degree for all $k \in \{1, \ldots, K\}$ as in $\mathcal{G}$. The general process for attaining the random edge multiset, $\widetilde{\mathcal{E}}$, from the observed edge multiset, $\mathcal{E}$, is presented in Figure C.1 where the permutation of node labels is depicted in Figure C.2. Notice, the process for permuting node labels differs for same type connections (e.g., $E^{[11]}$ and $E^{[22]}$) compared to between type connections (e.g., $E^{[12]}$) as described in cases one and two.

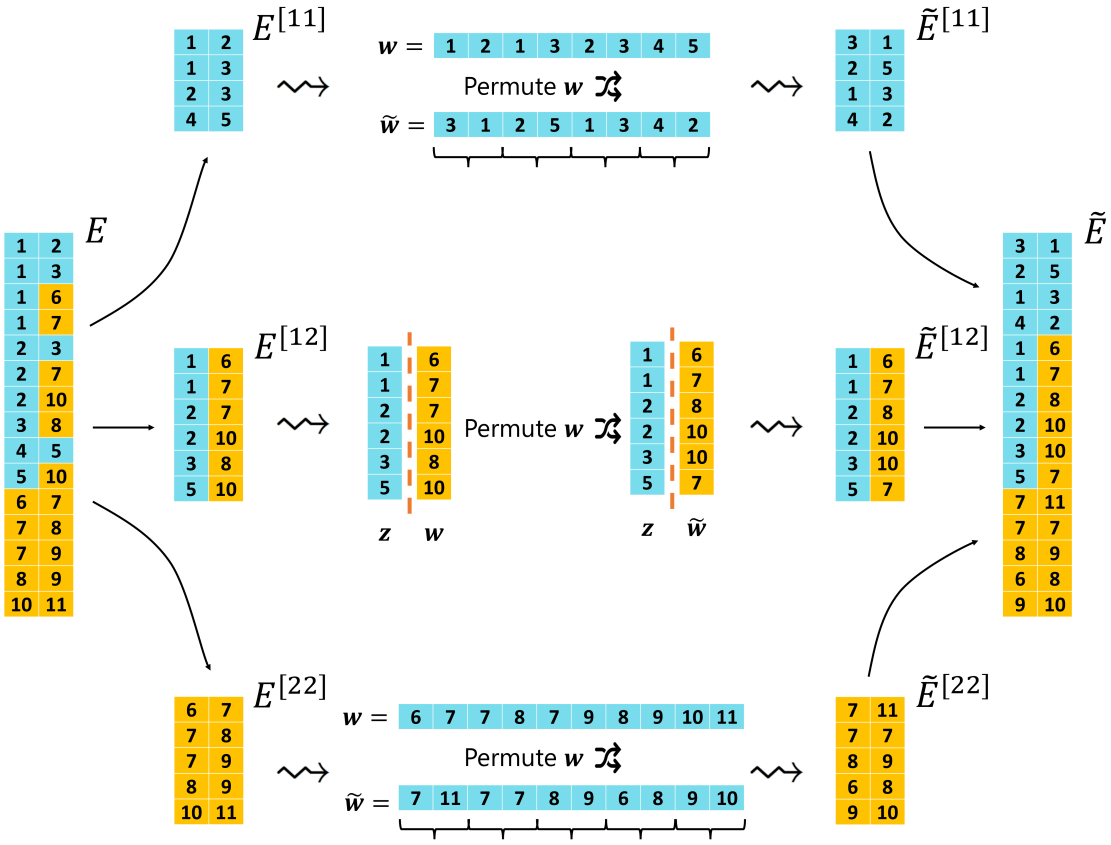


Figure C.2: A schematic describing the random permutation of node labels for each element of the partitioned edge multiset. The process for permuting node labels is different for same type connections compared to between type connections. For same type connections (i.e., those described by $E^{[11]}$ and $E^{[22]}$), all node labels as they appear in $E^{[kk]}$ are permuted and paired two by two to create a new edge multiset $\widetilde{E}^{[kk]}$. For between type connections (i.e., those described by $E^{[12]}$), only node labels of type l as they appear in $E^{[kl]}$ are permuted before being subsequently paired to the type k node labels to create the new edge multiset $\widetilde{E}^{[kl]}$. The resulting edge multisets (represented as matrices) are combined rowwise.

C.1.2 Generalization of a Degree Configuration Model

The heterogeneous degree configuration model is a generalization of a degree configuration model. A formal definition of a degree configuration model (DCM) for a homogeneous network is provided in Newman (2018, Chapter 13.2); however, Fosdick et al. (2018) provides expansive coverage on the topic. To clarify the similarities and differences between HDCM($\mathcal{T}, \boldsymbol{\mathcal{D}}$) and DCM($\mathcal{D}$), we will consider a random graph, $\tilde{\mathcal{G}}$, generated from the HDCM($\mathcal{T}, \boldsymbol{\mathcal{D}}$) for $K = 1$ and $K > 1$. For a general K, the heterogeneous degree sequence (introduced in the Heterogeneous Network section of Chapter 4) of node u in $\tilde{\mathcal{G}}$ is equal to the observed heterogeneous degree se-

quence of node u in $\mathcal{G}$ for all $u \in \mathcal{V}$. That is, node u is connected to the same number of type k nodes in $\tilde{\mathcal{G}}$ as in $\mathcal{G}$ for all $k \in \{1, \ldots, K\}$. As a result, node u is connected to the same number of nodes in $\tilde{\mathcal{G}}$ as in $\mathcal{G}$. Thus, the sample space of the HDCM$(\mathcal{T}, \boldsymbol{\mathcal{D}})$ is a subspace of the sample space of the DCM$(\mathcal{D})$. The two sample spaces are equal when $K = 1$ (i.e., the network is homogeneous) since the heterogeneous degree sequence of a node simplifies to a singleton set with the degree of a node. When $K > 1$, it is easy to construct an example where the degree of each node is the same as in the observed graph, but the heterogeneous degree sequences are different. When $K > 1$ the sample space of the HDCM$(\mathcal{T}, \boldsymbol{\mathcal{D}})$ is a strict subset of the DCM$(\mathcal{D})$. This illustrates how the heterogeneous degree configuration model is simply a generalization of the degree configuration model used in Wilson et al. (2014).

C.2 Proof of Theorems

The statement of Theorem 4.3.1, Corollary 4.3.1, and Theorem 4.3.2, along with corresponding assumptions and notation, are provided in Chapter 4. The corresponding proofs are provided here.

Asymptotic Distribution of $X_n^{[k]}(u_n^{[l]} : \mathcal{B}_n)$ We address the asymptotic behavior of the random quantity of interest in (4.4).

Proof of Theorem 4.3.1. From Assumption 4.3.2, we have

$$\mu_{l,n}^{[k]} = \int_{\mathbb{R}^+} x \, dF_{l,n}^{[k]}(x) = \sum_{t=0}^{\infty} t \frac{N_{l,n}^{[k]}(t)}{|V_n^{[l]}|} = \frac{2^{\mathbb{I}(k=l)}|E_n^{[kl]}|}{|V_n^{[l]}|} \to \mu_l^{[k]} \tag{C.1}$$

as $n \to \infty$ where $N_{l,n}^{[k]}(t)$ is the number of type l nodes with a type k degree of t, and $E_n^{[kl]}$ is the subset multiset of the edge multiset which contains links between type k nodes and type l nodes. Equation (C.1) implies

$$1 = \lim_{n\to\infty} \frac{2^{\mathbb{I}(k=l)}|E_n^{[kl]}|}{\mu_l^{[k]}|V_n^{[l]}|} \quad \text{(i.e., } 2^{\mathbb{I}(k=l)}|E_n^{[kl]}| \sim \mu_l^{[k]}|V_n^{[l]}|\text{)}$$

where $\mu_l^{[k]} < \infty$ by Assumption 4.3.2. By Assumption 4.3.1, we have

$$\frac{V_n^{[l]}}{\mathcal{V}_n} = \frac{V_n^{[l]}}{n} \to \gamma^{[l]} \in (0,1) \text{ as } n \to \infty \implies 1 = \lim_{n\to\infty} \frac{|V_n^{[l]}|}{\gamma^{[l]} n} \quad \text{(i.e., } |V_n^{[l]}| \sim \gamma^{[l]} n\text{)}.$$

Hence, we have $2^{\mathbb{I}(k=l)}|E_n^{[kl]}| \sim \mu_l^{[k]}\gamma^{[l]}n$. Note, when $\mu_l^{[k]} = 0$ then $X_n^{[k]}(u_n^{[l]}, \mathcal{B}_n) = 0, Y_n \sim \text{Binom}(c, 0)$ and $d_{TV}\left(X_n^{[k]}(u_n^{[l]}, \mathcal{B}_n), Y_n\right) \to 0$ as $n \to \infty$, trivially. We proceed assuming $\mu_l^{[k]} > 0$ before addressing this edge case formally.

We assume (WLOG) that $k \leq l$. We wish to understand the distribution of $X_n^{[k]}(u_n^{[l]} : \mathcal{B}_n)$: the random number of type k nodes in $\mathcal{B}_n$ adjacent to $u_n^{[l]}$ in $\mathcal{H}_n$ constructed via the heterogeneous degree configuration model. We make use of the procedure for sampling a network from the heterogeneous degree configuration model, denoted HDCM$(\mathcal{V}_n, \mathcal{E}_n)$, and discussed in Appendix C.1. Under the HDCM$(\mathcal{V}_n, \mathcal{E}_n)$, the edge multiset $\mathcal{E}_n = \cup_{1\leq i\leq j\leq K} E_n^{[ij]}$ is partitioned according to the adjacent nodes' types (where $E_n^{[ij]}$ contains the links between nodes of type i and j), the edges in $E_n^{[ij]}$ are randomly rearranged to preserve degree (resulting in $\tilde{E}_n^{[ij]}$ for all $1 \leq i \leq j \leq K$), and the resulting $\tilde{E}_n^{[ij]}$ are combined into one edge set (denoted $\tilde{\mathcal{E}}_n$). Each node in a random network $\mathcal{H}_n = (\mathcal{V}_n, \tilde{\mathcal{E}}_n)$ maintains the same heterogeneous degree sequence as observed in $\mathcal{G}_n = (\mathcal{V}_n, \mathcal{E}_n)$, implying there are precisely c edges incident to $u_n^{[l]}$ also incident a type k node in $\mathcal{H}_n$. Let $\tilde{v}_{ni}^{[k]}$ denote the i^{th} type k node where $\{\tilde{v}_{ni}^{[k]}, u_n^{[l]}\} \in \tilde{E}^{[kl]}$ for $i = 1, \ldots, c$. Note that the ordering carries no specific meaning but is used for bookkeeping. Self-loops and multi-edges complicate the interpretation and discussion surrounding $\tilde{v}_{ni}^{[k]}$. We refer to $\tilde{v}_{ni}^{[k]}$ simply as the ith type k node adjacent to $u_n^{[l]}$; however, we recognize the elements of $\{\tilde{v}_{ni}^{[k]}\}_{i=1}^{c}$ are not necessarily unique and count, for example, a node as twice adjacent to $u_n^{[l]}$ should there exist two edges between them. Furthermore, if $\tilde{v}_{ni}^{[k]} = u_n^{[l]}$ then $\{\tilde{v}_{ni}^{[k]}, u_n^{[l]}\}$ would constitute a self-loop; a self-loop increases the degree of a node by two, so we would say then that $u_n^{[l]}$ is twice adjacent to itself. In a simple setting (no self-loops and multi-edges), there is no distinction between counting the nodes adjacent to a node w and counting the nodes incident to an edge which is incident to a node w, so we treat them as the same objective here.

To understand the distribution of $X_n^{[k]}(u_n^{[l]} : \mathcal{B}_n)$, we focus on the adjacent type k nodes, $\{\tilde{v}_{ni}^{[k]}\}_{i=1}^{c}$, and reveal whether each $\tilde{v}_{ni}^{[k]}$ is a member of $\mathcal{B}_n$ (equivalently, a member of $B_n^{[k]}$). For $i \in \{1, \ldots, c\}$, let A_i denote a binary random variable indicating whether $\tilde{v}_{ni}^{[k]}$ is a member of $\mathcal{B}_n$. That is, $A_i = 1$ if the ith type k node adjacent to $u_n^{[l]}$ in $\mathcal{H}_n$ is also in $\mathcal{B}_n$ and 0 otherwise. Note that $X_n^{[k]}(u_n^{[l]} : \mathcal{B}_n) = \sum_{i=1}^{c} A_i$. Let $r_{l,i}^{[k]}(\mathcal{B}_n)$ denote the conditional probability of $A_i = 1$ conditional on the previous $i - 1$ revelations: $\{A_1, \ldots, A_{i-1}\}$.

Consider $r_{l,1}^{[k]}(\mathcal{B}_n)$. In this case, no type k nodes adjacent to $u_n^{[l]}$ have been revealed, so $r_{l,1}^{[k]}(\mathcal{B}_n)$ is the ratio of the number of type k nodes in $\mathcal{B}_n$ adjacent to type l nodes to the total number of type k nodes adjacent to type l nodes:

$$
\begin{aligned}
r_{l,1}^{[k]}(\mathcal{B}_n) &= \frac{\left(\sum_{w \in B_n^{[k]}} d^{[l]}(w)\right) - \mathbb{I}(k = l)\mathbb{I}(u_n^{[l]} \in \mathcal{B}_n)}{\left(\sum_{z \in V_n^{[k]}} d^{[l]}(z)\right) - \mathbb{I}(k = l)} \\
&= \frac{\left(\sum_{w \in B_n^{[k]}} d^{[l]}(w)\right) - \mathbb{I}(k = l)\mathbb{I}(u_n^{[l]} \in \mathcal{B}_n)}{2^{\mathbb{I}(k=l)}|E_n^{[kl]}| - \mathbb{I}(k = l)}.
\end{aligned}
$$

The indicators in the numerator ensure we do not count $u_n^{[l]}$ itself when counting the number of type k nodes in $\mathcal{B}_n$ which could be adjacent to $u_n^{[l]}$ (only a concern when $k = l$ and $u_n^{[l]} \in \mathcal{B}_n$). Similarly, the indicator in the denominator ensures we do not count $u_n^{[l]}$ itself when counting the number of type k nodes in $\mathcal{V}_n$ which could be adjacent to $u_n^{[l]}$ (only a concern when $k = l$). Now, consider $r_{l,2}^{[k]}(\mathcal{B}_n)$. One type k node adjacent to $u_n^{[l]}$ has been revealed. Hence, there is one fewer type k node which can possibly serve as $\tilde{v}_{n2}^{[k]}$. If $\tilde{v}_{n1}^{[k]}$ was revealed to be a member of $\mathcal{B}_n$ (i.e., $A_1 = 1$), then there is also one fewer type k node in $\mathcal{B}_n$ which can possibly serve as $\tilde{v}_{n2}^{[k]}$. Otherwise, if $A_1 = 0$, then the number of type k nodes in $\mathcal{B}_n$ which can possibly serve as $\tilde{v}_{n2}^{[k]}$ is unaffected; only the overall

count is affected. Thus,

$$r_{l,2}^{[k]}(\mathcal{B}_n) \in \left[\frac{\left(\sum_{w \in B_n^{[k]}} d^{[l]}(w)\right) - \mathbb{I}(k=l)\mathbb{I}(u_n^{[l]} \in \mathcal{B}_n) - 1}{2^{\mathbb{I}(k=l)}|E_n^{[kl]}| - \mathbb{I}(k=l) - 1}, \right.$$
$$\left. \frac{\left(\sum_{w \in B_n^{[k]}} d^{[l]}(w)\right) - \mathbb{I}(k=l)\mathbb{I}(u_n^{[l]} \in \mathcal{B}_n)}{2^{\mathbb{I}(k=l)}|E_n^{[kl]}| - \mathbb{I}(k=l) - 1} \right]$$

where the lowerbound arises if $A_1 = 1$ and the upperbound arises if $A_1 = 0$. Arguing analogously for $1 \leq i \leq c$, we have that $r_{l,i}^{[k]}(\mathcal{B}_n)$ is bounded uniformly on all prior $i-1$ revelations by

$$r_{l,i}^{[k]}(\mathcal{B}_n) \in \left[\frac{\left(\sum_{w \in B_n^{[k]}} d^{[l]}(w)\right) - \mathbb{I}(k=l)\mathbb{I}(u_n^{[l]} \in \mathcal{B}_n) - (i-1)}{2^{\mathbb{I}(k=l)}|E_n^{[kl]}| - \mathbb{I}(k=l) - (i-1)}, \right.$$
$$\left. \frac{\left(\sum_{w \in B_n^{[k]}} d^{[l]}(w)\right) - \mathbb{I}(k=l)\mathbb{I}(u_n^{[l]} \in \mathcal{B}_n)}{2^{\mathbb{I}(k=l)}|E_n^{[kl]}| - \mathbb{I}(k=l) - (i-1)} \right]$$

where the lowerbound arises if $A_j = 1$ for all $j \in \{1, \ldots, i-1\}$, and the upperbound arises if $A_j = 0$ for all $j \in \{1, \ldots, i-1\}$. Recall from (4.5) that

$$p_{l,n}^{[k]}(\mathcal{B}_n) = \frac{\sum_{w \in B_n^{[k]}} d^{[l]}(w)}{\sum_{z \in V_n^{[k]}} d^{[l]}(z)}.$$

We will show that $\sup_{1 \leq i \leq c} |r_{l,i}^{[k]}(\mathcal{B}_n) - p_{l,n}^{[k]}(\mathcal{B}_n)| \to 0$ as $n \to \infty$ by considering the case when $k = l$ and $k \neq l$.

Case 1: $k = l$

If $k = l$, then

$$r_{l,i}^{[l]}(\mathcal{B}_n) \in \left[\frac{\left(\sum_{w \in B_n^{[l]}} d^{[l]}(w)\right) - \mathbb{I}(u_n^{[l]} \in \mathcal{B}_n) - (i-1)}{2|E_n^{[ll]}| - i}, \frac{\left(\sum_{w \in B_n^{[l]}} d^{[l]}(w)\right) - \mathbb{I}(u_n^{[l]} \in \mathcal{B}_n)}{2|E_n^{[ll]}| - i} \right]$$

and

$$p_{l,n}^{[l]}(\mathcal{B}_n) = \frac{\sum_{w \in B_n^{[l]}} d^{[l]}(w)}{\sum_{z \in V_n^{[l]}} d^{[l]}(z)} = \frac{\sum_{w \in B_n^{[l]}} d^{[l]}(w)}{2|E_n^{[ll]}|}.$$

Hence,

$$\begin{aligned}
r_{l,i}^{[l]}(\mathcal{B}_n) - p_{l,n}^{[l]}(\mathcal{B}_n) &\geq \frac{\left(\sum_{w \in B_n^{[l]}} d^{[l]}(w)\right) - \mathbb{I}(u_n^{[l]} \in \mathcal{B}_n) - (i-1)}{2|E_n^{[ll]}| - i} - \frac{\sum_{w \in B_n^{[l]}} d^{[l]}(w)}{2|E_n^{[ll]}|} \\
&= \frac{i \sum_{w \in B_n^{[l]}} d^{[l]}(w) - 2i|E_n^{[ll]}| - 2\mathbb{I}(u_n^{[l]} \in \mathcal{B}_n)|E_n^{[ll]}| + 2|E_n^{[ll]}|}{2|E_n^{[ll]}|(2|E_n^{[ll]}| - i)} \\
&\geq \frac{i \sum_{w \in B_n^{[l]}} d^{[l]}(w) - 2i|E_n^{[ll]}|}{2|E_n^{[ll]}|(2|E_n^{[ll]}| - i)} \text{ taking } \mathbb{I}(u_n^{[l]} \in \mathcal{B}_n) = 1 \\
&\geq \frac{-i}{2|E_n^{[ll]}| - i} \text{ since } \sum_{w \in B_n^{[l]}} d^{[l]}(w) \geq 0,
\end{aligned}$$

and

$$\begin{aligned}
r_{l,i}^{[l]}(\mathcal{B}_n) - p_{l,n}^{[l]}(\mathcal{B}_n) &\leq \frac{\left(\sum_{w \in B_n^{[l]}} d^{[l]}(w)\right) - \mathbb{I}(u_n^{[l]} \in \mathcal{B}_n)}{2|E_n^{[ll]}| - i} - \frac{\sum_{w \in B_n^{[l]}} d^{[l]}(w)}{2|E_n^{[ll]}|} \\
&= \frac{i \sum_{w \in B_n^{[l]}} d^{[l]}(w) - 2\mathbb{I}(u_n^{[l]} \in \mathcal{B}_n)|E_n^{[ll]}|}{2|E_n^{[ll]}|(2|E_n^{[ll]}| - i)} \\
&\leq \frac{i \sum_{w \in B_n^{[l]}} d^{[l]}(w)}{2|E_n^{[ll]}|(2|E_n^{[ll]}| - i)} \text{ taking } \mathbb{I}(u_n^{[l]} \in \mathcal{B}_n) = 0 \\
&\leq \frac{i}{2|E_n^{[ll]}| - i} \text{ since } \sum_{w \in B_n^{[l]}} d^{[l]}(w) \leq 2|E_n^{[ll]}|.
\end{aligned}$$

Thus,

$$|r_{l,i}^{[l]}(\mathcal{B}_n) - p_{l,n}^{[l]}(\mathcal{B}_n)| \leq \frac{i}{2|E_n^{[ll]}| - i} \leq \frac{c}{2|E_n^{[ll]}| - i} \leq \frac{c}{2|E_n^{[ll]}| - c},$$

implying

$$\sup_{1\leq i\leq c} |r_{l,i}^{[l]}(\mathcal{B}_n) - p_{l,n}^{[l]}(\mathcal{B}_n)| \leq \frac{c}{2|E_n^{[ll]}| - c} \to 0 \tag{C.2}$$

as $n \to \infty$ since $2|E_n^{[ll]}| \sim \mu_l^{[l]}\gamma^{[l]}n$ and c is fixed.

Case 2: $k \neq l$

If $k \neq l$, then

$$r_{l,i}^{[k]}(\mathcal{B}_n) \in \left[\frac{\left(\sum_{w\in B_n^{[k]}} d^{[l]}(w)\right) - (i-1)}{|E_n^{[kl]}| - (i-1)}, \frac{\left(\sum_{w\in B_n^{[k]}} d^{[l]}(w)\right)}{|E_n^{[kl]}| - (i-1)}\right]$$

and

$$p_{l,n}^{[k]}(\mathcal{B}_n) = \frac{\sum_{w\in B_n^{[k]}} d^{[l]}(w)}{\sum_{z\in V_n^{[k]}} d^{[l]}(z)} = \frac{\sum_{w\in B_n^{[k]}} d^{[l]}(w)}{|E_n^{[kl]}|}.$$

Hence,

$$\begin{aligned}
r_{l,i}^{[k]}(\mathcal{B}_n) - p_{l,n}^{[k]}(\mathcal{B}_n) &\geq \frac{\left(\sum_{w\in B_n^{[k]}} d^{[l]}(w)\right) - (i-1)}{|E_n^{[kl]}| - (i-1)} - \frac{\sum_{w\in B_n^{[k]}} d^{[l]}(w)}{|E_n^{[kl]}|} \\
&= \frac{(i-1)\left[\sum_{w\in B_n^{[k]}} d^{[l]}(w) - |E_n^{[kl]}|\right]}{|E_n^{[kl]}|(|E_n^{[kl]}| - (i-1))} \\
&\geq \frac{-(i-1)|E_n^{[kl]}|}{|E_n^{[kl]}|(|E_n^{[kl]}| - (i-1))} \text{ since } \sum_{w\in B_n^{[k]}} d^{[l]}(w) \geq 0 \\
&= \frac{-(i-1)}{|E_n^{[kl]}| - (i-1)},
\end{aligned}$$

and

$$\begin{aligned} r_{l,i}^{[k]}(\mathcal{B}_n) - p_{l,n}^{[k]}(\mathcal{B}_n) &\leq \frac{\sum_{w \in B_n^{[k]}} d^{[l]}(w)}{|E_n^{[kl]}| - (i-1)} - \frac{\sum_{w \in B_n^{[k]}} d^{[l]}(w)}{|E_n^{[kl]}|} \\ &= \frac{(i-1)\sum_{w \in B_n^{[k]}} d^{[l]}(w)}{|E_n^{[kl]}|(|E_n^{[kl]}| - (i-1))} \\ &\leq \frac{(i-1)|E_n^{[kl]}|}{|E_n^{[kl]}|(|E_n^{[kl]}| - (i-1))} \text{ since } \sum_{w \in B_n^{[k]}} d^{[l]}(w) \leq |E^{[kl]}| \\ &= \frac{i-1}{|E_n^{[kl]}| - (i-1)}. \end{aligned}$$

Thus,

$$|r_{l,i}^{[k]}(\mathcal{B}_n) - p_{l,n}^{[k]}(\mathcal{B}_n)| \leq \frac{i-1}{|E_n^{[kl]}| - (i-1)} \leq \frac{c-1}{|E_n^{[kl]}| - (i-1)} \leq \frac{c-1}{|E_n^{[kl]}| - (c-1)},$$

implying

$$\sup_{1 \leq i \leq c} |r_{l,i}^{[k]}(\mathcal{B}_n) - p_{l,n}^{[k]}(\mathcal{B}_n)| \leq \frac{c-1}{|E_n^{[kl]}| - (c-1)} \to 0 \tag{C.3}$$

as $n \to \infty$ since $|E_n^{[kl]}| \sim \mu_l^{[k]} \gamma^{[l]} n$ and c is fixed.

Thus, $\sup_{1 \leq i \leq c} |r_{l,i}^{[k]}(\mathcal{B}_n) - p_{l,n}^{[k]}(\mathcal{B}_n)| \to 0$ as $n \to \infty$, implying $d_{TV}\left(X_n^{[k]}(u_n^{[l]}, \mathcal{B}_n), Y_n\right) \to 0$ as $n \to \infty$ where $Y_n \sim \text{Binom}(c, p_{l,n}^{[k]}(\mathcal{B}_n))$.

We now address the trivial cases when $\mu_l^{[k]} = 0$. Let $Y_n \sim \text{Binom}(c, 0)$, implying $P(Y_n = 0) = 1$. Suppose $\mu_l^{[k]} = 0$, and note that $X_n^{[k]}(u_n^{[l]}, \mathcal{B}_n)$ is a non-negative random variable. Then, for all $\epsilon > 0$

$$\mathbb{P}(X_n^{[k]}(u_n^{[l]}, \mathcal{B}_n) > \epsilon) \leq \frac{\mathbb{E}(X_n^{[k]}(u_n^{[l]}, \mathcal{B}_n))}{\epsilon} = \frac{\mu_{l,n}^{[k]}}{\epsilon} \to \frac{\mu_l^{[k]}}{\epsilon} = 0$$

as $n \to \infty$ by Markov's inequality. Hence, $d_{TV}\left(X_n^{[k]}(u_n^{[l]}, \mathcal{B}_n), Y_n\right) \to 0$ as $n \to \infty$. ■

Proof of Corollary 4.3.1. To demonstrate that $d_{TV}\left(X_n^{[k]}(u_n^{[l]}, \mathcal{B}_n), Y_{l,n}^{[k]}(u_n^{[l]}, \mathcal{B}_n)\right) \to 0$ as $n \to \infty$, it suffices to show that $\sup_{1\leq i\leq c} |r_{l,i}^{[k]}(\mathcal{B}_n) - p_{l,n}^{[k]}(u_n^{[l]}, \mathcal{B}_n)| \to 0$ as $n \to \infty$ when $k = l$ and $0 < \mu_l^{[k]} < \infty$. Using a similar proof strategy, we have

$$r_{l,i}^{[k]}(\mathcal{B}_n) - p_{l,n}^{[k]}(u_n^{[l]}, \mathcal{B}_n) \geq -\frac{2|E_n^{[kl]}|(c-1) - c(i-1)}{(2|E_n^{[kl]}| - i)(2|E_n^{[kl]}| - c)},$$

and

$$r_{l,i}^{[k]}(\mathcal{B}_n) - p_{l,n}^{[k]}(u_n^{[l]}, \mathcal{B}_n) \leq \frac{2|E_n^{[kl]}|(c-1) - c(i-1)}{(2|E_n^{[kl]}| - i)(2|E_n^{[kl]}| - c)}.$$

Thus,

$$\begin{aligned} |r_{l,i}^{[k]}(\mathcal{B}_n) - p_{l,n}^{[k]}(u_n^{[l]}, \mathcal{B}_n)| &\leq \frac{2|E_n^{[kl]}|(c-1) - c(i-1)}{(2|E_n^{[kl]}| - i)(2|E_n^{[kl]}| - c)} \\ &\leq \frac{(2|E_n^{[kl]}| - i)(c+1)}{(2|E_n^{[kl]}| - i)(2|E_n^{[kl]}| - c)} \qquad \text{(C.4)} \\ &= \frac{c+1}{2|E_n^{[kl]}| - c} \end{aligned}$$

where (C.4) holds since $c \leq |E_n^{[kl]}|$, implying

$$\sup_{1\leq i\leq c} |r_{l,i}^{[k]}(\mathcal{B}_n) - p_{l,n}^{[k]}(u_n^{[l]}, \mathcal{B}_n)| \leq \frac{c+1}{2|E_n^{[kl]}| - c} \to 0 \qquad \text{(C.5)}$$

as $n \to \infty$ since $|E_n^{[kl]}| \sim \mu_l^{[k]}\gamma^{[l]}n$ and c is fixed. Equation (C.5) implies convergence in total variation, a result mirroring Theorem 4.3.1. ■

Convergence of the ECoHeN Algorithm We address the convergence properties of the ECoHeN algorithm.

Proof of Theorem 4.3.2. Since $\xi \in [0, 1]$ and $\phi \in [0, 1)$, there exists an iteration $i^{'}$ such that $\mu_{i'} = 1$ and $\mu_i = 1$ for all $i \geq i^{'}$. Furthermore, there exists a $j^{'}$ such that $|\mathcal{B}_i| < |\mathcal{V}|/2$ for all $i \geq j^{'}$. Let

$j = \max(i^{'}, j^{'})$. For all $i \geq j$, the ECoHeN extraction procedure iteratively adds (at most) a one external node, $v \in \mathcal{B}_j^c$, with the smallest FDR adjusted p-value less than (or equal to) α before removing (at most) one internal node, $v \in \mathcal{B}_j^+$, with the largest FDR adjusted p-value greater than α. To illustrate convergence, it suffices to show that a node, v, added to the set $\mathcal{B}_j$ at iteration j is not subsequently removed from the set $\mathcal{B}_j^+ = \mathcal{B}_j \cup \{v\}$ at iteration j. We proceed by contradiction.

Importantly, note that the quantity $\hat{p}_{\mathcal{B}_j}(u)$ for an arbitrary node u of arbitrary type l is the same regardless of whether $u \in \mathcal{B}_j$ or $u \notin \mathcal{B}_j$, implying $\hat{p}_{\mathcal{B}_j}(v) = \hat{p}_{\mathcal{B}_j^+}(v)$. Let $q_{\text{ext}} := q_{\text{ext}}(j) = |\mathcal{B}_j^c|$ denote the number of external nodes, and $q_{\text{int}} := q_{\text{int}}(j) = |\mathcal{B}_j^+|$ denote the number of internal nodes at iteration j. Since $q_{\text{int}} < q_{\text{ext}}$ and $\hat{p}_{\mathcal{B}_j}(v) = \hat{p}_{\mathcal{B}_j^+}(v)$, we have $\tilde{\hat{p}}_{\mathcal{B}_j^+}(v) < \tilde{\hat{p}}_{\mathcal{B}_j}(v)$ where $\tilde{\hat{p}}$ the FDR adjusted p-value. Intuitively, since the p-value for v is the same regardless whether v is a member of $\mathcal{B}_j$, then the FDR adjusted p-value for v will be smaller when v is a member of $\mathcal{B}_j$ (i.e., $\mathcal{B}_j^+$) than when v is not a member of $\mathcal{B}_j$ (i.e., $\mathcal{B}_j$) since there are more external nodes than internal nodes at iteration j. At the same time, since v was added to $\mathcal{B}_j$ then $p_{\mathcal{B}_j}(v) \leq \alpha$. Since v was subsequently removed from $\mathcal{B}_j^+$, then $p_{\mathcal{B}_j^+}(v) > \alpha$. Hence we have $\alpha < \tilde{p}_{\mathcal{B}_j^+}(v) < \tilde{p}_{\mathcal{B}_j}(v) \leq \alpha$, a clear contradiction: $\alpha < \alpha$. As such, v will never be added and subsequently removed in the extraction procedure, implying the ECoHeN algorithm will not cycle. ■

C.3 Simulation Study

We provide a detailed account of the heterogeneous stochastic block model: a model for generating heterogeneous networks with block structure. Furthermore, we discuss the evaluation metrics used in Chapter 4 and provide a more expansive set of simulated conditions.

C.3.1 Heterogeneous Stochastic Block Model

The *heterogeneous stochastic block model* (HSBM) is a flexible framework for generating heterogeneous networks with block structure and is implemented in the R package `ECoHeN` provided in Appendix C.4. To describe the size and connectivity of sampled networks, consider a heterogeneous network with K node types with C blocks. The block sizes are described by the K

by C matrix $N = [n_{kc}]_{\substack{1\leq k\leq K\\ 1\leq c\leq C}}$ where n_{kc} provides the number of type k nodes assigned to the c^{th} block. The connectivity of a sampled network is then summarized by the symmetric matrices P and R, each of size K by K, which provide the probability of connections between nodes of type k and l for all $k, l \in \{1, \ldots, K\}$.

To detail, suppose $v^{[k]}$ and $u^{[l]}$ represent two arbitrary nodes of type k and l, respectively. In particular, the matrix $P = [p_{kl}]_{1\leq k\leq l\leq K}$ provides the probability of connection between two nodes should these nodes exist in separate blocks. That is, $v^{[k]}$ and $u^{[l]}$ do not share a block, then an edge is placed between them according to a Bernoulli random variable with rate p_{kl}. The matrix $R = [r_{kl}]_{1\leq k\leq l\leq K}$ provides the additive increase in the rate of connection between two nodes if they share a block; that is, $v^{[k]}$ and $u^{[l]}$ share a block, then an edge is placed between them according to a Bernoulli random variable with rate $p_{kl} + r_{kl}$ where $0 \leq p_{kl}, r_{kl} \leq 1$ and $0 \leq p_{kl} + r_{kl} \leq 1$ for all $1 \leq k \leq l \leq K$.

Simulated networks presented in Chapter 4 are generated by parameters b, p, r_{11}, r_{22}, and r_{12} according to the following matrices of the HSBM:

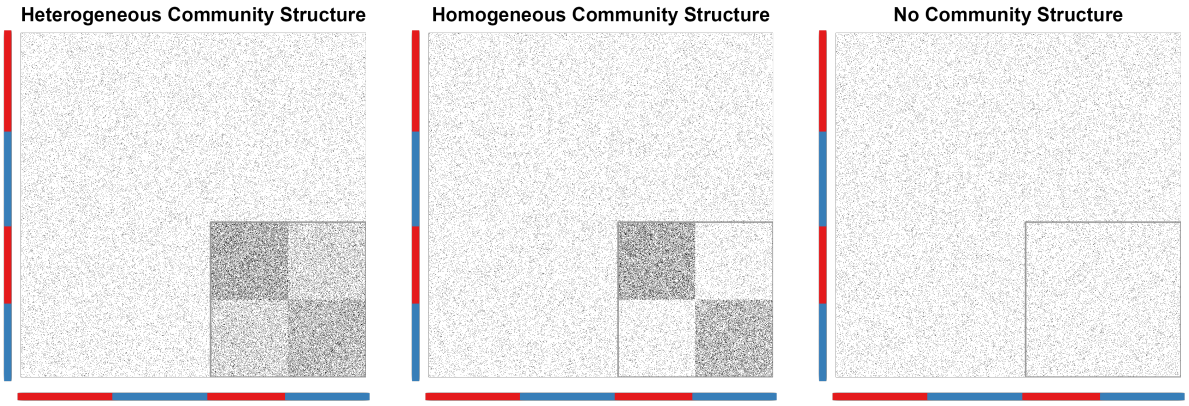
$$N(p) = \begin{bmatrix} 500(1-p) & 500p \\ 500(1-p) & 500p \end{bmatrix}, \tag{C.6}$$

$$P(b) = \begin{bmatrix} b & b \\ b & b \end{bmatrix}, \text{ and} \tag{C.7}$$

$$R(r_{11}, r_{22}, r_{12}) = \begin{bmatrix} r_{11} & r_{12} \\ r_{12} & r_{22} \end{bmatrix}. \tag{C.8}$$

Networks of this form maintain two blocks: a background block and a high connectivity block (HCB) where the parameter p in (C.6) dictates the size of the HCB, ergo, the size of the community. The parameter b in (C.7) dictates the background rate. The parameters r_{ij} in (C.8) dictate the type and degree of community structure. Figure C.3 provides three examples of a heterogeneous network from this space, including one (a) with heterogeneous community structure, (b)

with homogeneous community structure, and (c) without community structure, each visualized as an adjacency matrix with node type illustrated by the colored bars adorning the axes. In particular, Figure C.3a is said to have heterogeneous community structure as the within-block nodes are highly connected within node type and (to a lesser degree) between node type, implying the existence of one heterogeneous community. On the other hand, Figure C.3b is said to have homogeneous community structure as the within-block nodes are highly connected within node type yet sparsely connected between node type (equivalent to the background rate, b), implying the existence of two homogeneous communities: one composed of type one (red) nodes and one composed of type two (blue) nodes. Figure C.3c is a heterogeneous analog of an Erdős-Rényi network; hence, there is no underlying community structure.



(a) $R(0.25, 0.20, 0.10)$ **(b)** $R(0.25, 0.20, 0)$ **(c)** $R(0, 0, 0)$

Figure C.3: Three example heterogeneous networks generated from the heterogeneous stochastic block model with fundamentally different community structure. Each network is composed of 500 type one and 500 type two nodes (illustrated by the colored bars adorning the axes) and the subsequent connections among them. The HCB (outlined in black) contains the connections dictated by r_{ij} for $1 \leq i \leq j \leq 2$. The other connections are dictated by the background rate b. Edges are sampled according to the Bernoulli rate matrices $P(b)$ and $R(r_{11}, r_{22}, r_{12})$ of (C.7) and (C.8) with background rate specified by $b = 0.05$.

C.3.2 Evaluation Metrics

The maximum Jaccard similarity measure is used in simulation to capture each community discovery method's ability to identify the simulated community structure. For each method, let

D represent the set of nodes to discover which will vary depending on the underlying community structure. Let $\boldsymbol{C}_m$ represent a collection of the discovered communities by method m. To gauge each method's ability to identify D, we consider the *maximum Jaccard similarity measure*, denoted $J^*(D, \boldsymbol{C}_m)$, between the set of nodes to discover, D, and the collection of communities, $\boldsymbol{C}_m$:

$$J^*(D, \boldsymbol{C}_m) = \max_{C \in \boldsymbol{C}_m} \frac{|D \cap C|}{|D \cup C|} := \max_{C \in \boldsymbol{C}_m} J(D, C). \tag{C.9}$$

A value of one indicates that the set D was perfectly identified by the community discovery method; a value of zero indicates the set D was not identified in any capacity by the community discovery method. Hence, the maximum Jaccard similarity measure illustrates each method's ability to recover set D where larger values indicate more overlap.

In real networks, there is no established ground truth. To assess the assortativity of a discovered community, C, we compute the ratio of densities. The ratio of densities of C, denoted RatD(C), is the ratio of the internal edge density, $p_i(C)$, to the between edge density, $p_b(C)$: RatD$(C) = p_i(C)/p_b(C)$ where

$$p_i(C) = \frac{m_i}{|C|(|C|-1)/2}, \text{ and } p_b(C) = \frac{m_b}{|C||C^c|}.$$

We use m_i to denote the number of edges between nodes in C, and m_b to denote the number of edges between a node in C and a node in $C^c = \mathcal{V} - C$. A RatD of one implies that the density of edges within the set is the same as to the density to the rest of the network, indicating poor assortativity. Any set with a RatD sufficiently greater than one can arguably be called a community. The formulation assumes a simple network which holds for every network presented in Chapter 4.

C.3.3 Simulation Study

The simulation study in Chapter 4 features a subset of the results depicting each method's ability to identify heterogeneous and homogeneous community structure. This section presents all other results as well as the effects of ξ and ϕ on the quantity and quality of the communities

found. We also provide an investigation into each method's ability to assign nodes to background in random networks.

Heterogeneous Community Structure

In Chapter 4, we present each method's ability to identify the heterogeneous community when $r_{ii} \in (0.15, 0.20, 0.25, 0.30)$ for $i \in \{1, 2\}$ and $r_{12} \in (0.025, 0.05, 0.075)$; however, a broader range of simulated conditions are constructed and explored. We present results when $r_{11} = r_{22}$ and $r_{11} \neq r_{22}$ before exploring the effects of ξ and ϕ. As in Chapter 4, one hundred networks are generated at each simulated condition, referred to as replicates. The median maximum Jaccard similarity measure is plotted at each simulated condition with uncertainty reflected by the range of first and third quartile.

When $r_{11} = r_{22}$ Figure C.4 compares each method's ability to identify the heterogeneous community when $r_{11} = r_{22}$ at a broader range of simulated conditions. Each point represents the median maximum Jaccard at each simulated condition. The vertical range represents the middle 50% of observed maximum Jaccard measures. ECoHeN and ZCmod's ability to recover the heterogeneous community notably improves as the within-block, within-type density (i.e., $b + r_{ii}$) increases *and* the within-block between-type density (i.e., $b + r_{ij}$) increases. Each method poorly recovers the heterogeneous community when $r_{ii} < 0.15$. When $r_{ii} < 0.15$, Walktrap generally seems preferable, having a larger maximum Jaccard. However, ECoHeN and ZCmod outperform Walktrap if r_{12} is relatively large, and if r_{12} is sufficiently large, ECoHeN outperforms each of the competing methods. When $r_{ii} \geq 0.15$, ECoHeN performs relatively better than each competing methods at recovering small heterogeneous communities (i.e., when p is small), especially for relatively small r_{12}. For larger p, ECoHeN and ZCmod perform similarly well.

When $r_{11} \neq r_{22}$ We now consider simulated conditions when $r_{11} \neq r_{22}$, presented in Figure C.5. The within-block, within-type densities are provided in the facets. For each community size, we show the maximum Jaccard for increasing values of r_{12} which range from 0.025 to r_{22}.

Each method's ability to recover the heterogeneous community improves as (1) the density of red block nodes' connections increases (down the y-axis facets), (2) the density of the blue block nodes' connections increases (right across the x-axis facets), and (3) the density of the within-block, between-type connections increases (provided by the line type). Characteristically, however, ECoHeN's ability to recover the heterogeneous community drastically improves for small increases to r_{12}, outperforming each competing method at recovering small, heterogeneous communities. By comparison, ZCmod struggles to identify small, heterogeneous communities, performing well as the community size increases. When $r_{ii} > 0.20$ for $i \in \{1, 2\}$ (not shown), all methods perform similarly for large r_{12}. For small r_{12}, ESSC and Walktrap perform relatively worse than ECoHeN and ZCmod. ECoHeN outperforms ZCmod for small r_{12}, and the two methods perform more similarly as r_{12} increases.

Effects of ξ and ϕ We consider setting (ξ, ϕ) to $(0, 0)$, and $(1, \phi)$ for $\phi = 0, 0.33, 0.66$, and 0.99 and running ECoHeN and ESSC to see the effect the parameter settings have on the ability for the methods to recover the heterogeneous community. The results are provided in Figure C.6. In general, it appears that setting the maximal allowance to 1 for each iteration of the extraction with $\xi = 0$ and $\phi = 0$ provides the best resolution for uncovering the heterogeneous community at a wide range of simulated conditions. When $\xi = 1$, a larger ϕ provides the best resolution, suggesting that if we are to speed up the algorithm by increasing the maximal allowance for early iterations, it is best to allow for a larger maximal allowance for many early iterations. However, the effect is relatively minute for most simulated conditions, suggesting that the choice of ϕ when $\xi = 1$ will have minimal impact on the methods' ability to recover communities from background. We do not consider setting $(\xi, \phi) = (1, 1)$ as this setting is not guaranteed to converge. Should the user wish to set $(\xi, \phi) = (1, 1)$, we suggest considering a maximum number of iterations fewer than the number of nodes in the network.

Homogeneous Community Structure

In Chapter 4, we present each method's ability to identify the red community when $r_{11} \in (0.20, 0.25, 0.30)$, $r_{22} = 0.25$, and $r_{12} = 0$; however, a broader range of simulated conditions are constructed and explored. We present results for identifying both the red and blue community for general $r_{11} \in (0.15, 0.20, 0.25, 0.30)$ where $i \in \{1, 2\}$ before exploring the effects of ξ and ϕ. As in Chapter 4, one hundred networks are generated at each simulated condition, referred to as replicates. The median maximum Jaccard similarity measure is plotted at each simulated condition with uncertainty reflected by the range of first and third quartile.

Figure C.7 compares each method's ability to identify the red community at a broader range of simulated conditions. Each point represents the median maximum Jaccard at each simulated condition. The vertical range represents the middle 50% of observed maximum Jaccard measures. As the red to red density increases within the HCB (along the y-axis facets), ECoHeN can identify ECoHeN can identify the homogeneous community composed of red nodes with increasingly better precision, featuring marked improvements for small, homogeneous communities. There are no such improvements from ZCmod which partitions a network into modules each of which must maintain at least one node of each node type. Notably, the blue to blue density does not have an impact on any method's ability to identify the homogeneous community composed of red nodes.

While ECoHeN consistently outperforms ZCmod, ESSC and Walktrap consistently outperform ECoHeN and ZCmod at uncovering homogeneous community structure. This is not surprising considering these methods identify communities irrespective of node type. By construction, the within-block, between-type density is no different from the background density for networks under study. While ECoHeN is designed to identify both homogeneous and heterogeneous community structure, the tradeoff for this functionality is a reduction in power for uncovering homogeneous communities since there is no information gained through a comparison of the within-block, between-type density to the background. Notably, ESSC performs better or similarly to Walktrap at uncovering dense, homogeneous communities, and Infomap continues to place each node into it's own community, incapable of identifying a community amongst background noise.

Lastly, Figure C.8 compares each method's ability to identify the blue community at the same range of simulated conditions. The resulting conclusions are the same.

Effects of ξ and ϕ We consider setting (ξ, ϕ) to $(0, 0)$, and $(1, \phi)$ for $\phi = 0, 0.33, 0.66$, and 0.99 and running ECoHeN and ESSC to see the effect the parameter settings have on the ability for the methods to recover the homogeneous communities. The results are provided in Figure C.9 and C.10. As previously founded, it appears setting the maximal allowance to one for each iteration with $\xi = 0$ and $\phi = 0$ provides the best resolution for uncovering the homogeneous communities. When $\xi = 1$, a larger ϕ provides the best resolution, suggesting that if we are to speed up the algorithm by allowing a larger maximal allowance for early iterations, it is best to allow for a larger maximal allowance for many early iterations. We again do not consider setting $(\xi, \phi) = (1, 1)$ as this setting is not guaranteed to converge to a solution. Should the user wish to set $(\xi, \phi) = (1, 1)$, we suggest considering a maximum number of iterations fewer than the number of nodes in the network.

No Community Structure

In this section, we investigate each method's ability to identify background nodes: nodes which are not preferentially attached to any well-defined community. We generate networks with no community structure by fixing $R = R(0, 0, 0)$ and letting $P = P(b)$ where $b \in (0.05, 0.10, \ldots, 0.35)$. The proportion of nodes assigned to the HCB, p, is meaningless provided the choice of R, so we fix $p = 0.45$ as in Figure C.3c. The random networks under this setting are Erdős-Rényi networks with rate b, so no community structure exists. Equivalently, the set $D = \emptyset$. For each b, we produce twenty replicates.

We compare each method's ability to identify background. Both ESSC and ECoHeN are capable of assigning nodes to background; however, since ZCmod, Infomap, and Walktrap are partitioning methods, they are at an innate disadvantage to identify background nodes. For a fair comparison, *we consider the largest identified community to reflect the background of each partitioning method*. If a trivial partition is assigned (i.e., all nodes are assigned to one community

or each node is assigned to its own community) then the partition is assumed to identify each node as a background node. For each replicate, we compute the number of communities found by each method and the proportion of nodes identified as background nodes. The results are provided in Figure C.11.

As discussed in "Initialization" subsection of Chapter 4, ECoHeN is initialized at the neighborhood of each node. By nature, the neighborhoods in an Erdős-Rényi network have low conductance when compared to a random set of nodes. When the extraction procedure is parameterized to a maximal allowance of one for each iteration (i.e., $\xi = 0$ and $\phi = 0$), the neighborhoods are updated one node at a time, resulting in a small set of densely connected nodes. Thus, ECoHeN returns many small communities (see Figure C.11a). As the density of the Erdős-Rényi network gets larger, the number of communities found by ECoHeN tends to zero and the proportion of nodes assigned to background approaches one (see Figure C.11c). We demonstrate in the next subsection that these small communities are indeed particularly dense, at times ten to twenty times more connected internally than connected to the rest of the network.

To avoid finding communities in a random network, we can parameterize the extraction procedure such that the maximal allowance is one on the first iteration and tends to one with each passing iteration by setting the learning rate, $\xi = 1$, and the decay rate, $\phi < 1$. In this case, the algorithm is guaranteed to converge, and the number of communities found by ECoHeN is negligible for small b and quickly tends to 0 as the b gets larger (see Figure C.11b). In all cases, the proportion of nodes assigned to background is near one (see Figure C.11d). If one wishes to avoid finding dense subsets of nodes in an Erdős-Rényi graph, then it suffices to set $\xi = 1$ regardless of choice of ϕ. When $\xi = 1$, a larger choice of ϕ tends to yield slightly better performance at recovering simulated community structure; however, the impact of ϕ is minimal when $\xi = 1$.

Investigation of Identified Communities We previously demonstrated that the extraction procedure parameterized by $\xi = 0$ and $\phi = 0$ provides the best resolution for extracting simulated heterogeneous and homogeneous communities from background. At the same time, we demonstrated that such a parameterization results in many small communities in a heterogeneous Erdős-

Rényi network. Finding communities in an Erdős-Rényi network is not ideal behavior, so we investigate the characteristics of these identified communities. Considering the heterogeneous and homogeneous simulations were conducted at a background rate of 0.05, we will be examining the communities found by ECoHeN when $b = 0.05$.

In each of twenty replicates, ECoHeN uncovers between 300 and 400 communities (see Figure C.12). To gauge the assortativity of each identified community, we compute the ratio of densities for each. In an Erdős-Rényi network, the expected ratio of densities is one, so any set with a ratio of densities sufficiently greater than one can arguably be called a community.

Each of the communities found by ECoHeN is particularly dense where density scales naturally with the size of the community (see Figure C.13). The majority of communities found by ECoHeN are small, and these communities feature a large ratio of densities, sometimes proving twenty times more dense internally than to the rest of the network. The larger communities found by ECoHeN tend to have relatively smaller (albeit large) ratio of densities, proving at least five times more dense internally than to the rest of the network.

To gauge how unlikely it would be to attain the observed ratio of densities, we gather 1000 snowball samples for each community, compute the ratio of densities for each sample, and record the 95% quantile. Any observed ratio of densities larger than the respective 95% quantile is deemed sufficiently dense to be considered a community. For computational feasibility, we isolate the largest community found by ECoHeN at each of the twenty replicates, plotted in Figure C.14 in relation to the observed ratio of densities. Notice, as the community size increases, the threshold ratio of densities required decreases as it becomes increasingly unlikely to observe large communities with a large ratio of densities than a small community with an equally large ratio of densities. Each of the largest ECoHeN communities is sufficiently dense compared to the respective threshold, indicating that while these communities were attained from an Erdős-Rényi network, they are still sufficiently dense.

To describe the snowball sampling routine, consider an ECoHeN community of size n_C. One snowball sample is attained by first picking a node uniformly at random and recording its neigh-

bors. There are $n_C - 1$ nodes left to select. Should there be more neighbors than left to select, the remaining nodes left to select are chosen from the recorded neighbors uniformly at random. Otherwise, all of the neighbors are recorded, and the number of nodes left to select is updated accordingly. The process continues until n_C nodes have been selected. Since the snowball sampling routine results in dense, well-connected sets of nodes, it is a reasonable (albeit computationally burdensome) null model when assessing community structure in an Erdős-Rényi network.

C.4 Extraction Routines

The "ECoHeN Algorithm" section of Chapter 4 describes the ECoHeN algorithm in depth. The extraction routines AddWellConnected and RemoveLooselyConnected from Algorithm 4.1 are provided in pseudocode in Algorithm C.1. For the most up-to-date R implementation of ECoHeN (with a `C++` backend), see the following GitHub url: https://github.com/ConGibbs10/ECoHeN.

Extraction Routines

AddWellConnected

Inputs: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, node subset $\mathcal{B}_i$, and maximal allowance μ.

Return: The node subset with any external nodes which are well-connected $\mathcal{B}_i^+$.

Let $\mathcal{B}_i^c = \mathcal{V} - \mathcal{B}_i$, and $q = |\mathcal{B}_i^c|$ denote the number of external nodes.

For each $u \in \mathcal{B}_i^c$ **do :**

Denote the node type of node u as l.

Compute $p_{\mathcal{B}i}(u) = \prod_{k=1}^{K} \mathbb{P}\left(Y_l^{[k]}(u, \mathcal{B}_i) \geq x^{[k]}(u : \mathcal{B}_i)\right)$ using Corollary 3.1 where $Y_l^{[k]}(u, \mathcal{B}_i) \sim \text{Binom}(d^{[k]}(u), p_l^{[k]}(u, \mathcal{B}_i))$ and

$$p_l^{[k]}(u, \mathcal{B}_i) = \frac{\left[\sum_{w \in \mathcal{B}_i^{[k]}} d^{[l]}(w)\right] - \mathbb{I}(k = l)\mathbb{I}(u \in \mathcal{B}_i)d^{[l]}(u)}{2^{\mathbb{I}(k=l)}|E^{[kl]}| - \mathbb{I}(k = l)d^{[l]}(u)}.$$

Order the q external nodes so that $p_{\mathcal{B}_i}(u_1) \leq \cdots \leq p_{\mathcal{B}_i}(u_q)$.

Perform an FDR correction on $\{p_{\mathcal{B}_i}(u_j)\}_{j=1}^{q}$, and let $\{\tilde{p}_{\mathcal{B}_i}(u_j)\}_{j=1}^{q}$ denote the set of adjusted p-values.

Let $\tilde{\mu} = \min(\mu, q)$ denote the enforced maximal allowance.

Let $\mathcal{B}_i^+ = \mathcal{B}_i \cup \{u \in \{u_j\}_{j=1}^{\tilde{\mu}} : \tilde{p}_{\mathcal{B}_i}(u) \leq \alpha\}$ denote the set $\mathcal{B}_i$ and any well-connected external nodes.

RemoveLooselyConnected

Inputs: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, node subset $\mathcal{B}_i^+$, and maximal allowance μ.

Return: The node subset without any internal nodes which are not well-connected $\mathcal{B}_{i+1}$.

Let $q = |\mathcal{B}_i^+|$ denote the number of internal nodes.

For each $u \in \mathcal{B}_i^+$ **do :**

Denote the node type of node u as l.

Compute $p_{\mathcal{B}i}(u) = \prod_{k=1}^{K} \mathbb{P}\left(Y_l^{[k]}(u, \mathcal{B}_i) \geq x^{[k]}(u : \mathcal{B}_i)\right)$ using Corollary 3.1 where $Y_l^{[k]}(u, \mathcal{B}_i) \sim \text{Binom}(d^{[k]}(u), p_l^{[k]}(u, \mathcal{B}_i))$ and

$$p_l^{[k]}(u, \mathcal{B}_i) = \frac{\left[\sum_{w \in \mathcal{B}_i^{[k]}} d^{[l]}(w)\right] - \mathbb{I}(k = l)\mathbb{I}(u \in \mathcal{B}_i)d^{[l]}(u)}{2^{\mathbb{I}(k=l)}|E^{[kl]}| - \mathbb{I}(k = l)d^{[l]}(u)}.$$

Order the q internal nodes so that $p_{\mathcal{B}_i^+}(u_1) \geq \cdots \geq p_{\mathcal{B}_i^+}(u_q)$.

Perform an FDR correction on $\{p_{\mathcal{B}_i^+}(u_j)\}_{j=1}^{q}$, and let $\{\tilde{p}_{\mathcal{B}_i^+}(u_j)\}_{j=1}^{q}$ denote the set of adjusted p-values.

Let $\tilde{\mu} = \min(\mu, q)$ denote the enforced maximal allowance.

Let $\mathcal{B}_{i+1} = \mathcal{B}_i^+ - \{u \in \{u_j\}_{j=1}^{\tilde{\mu}} : \tilde{p}_{\mathcal{B}_i^+}(u) > \alpha\}$ denote the set $\mathcal{B}_i^+$ without any loosely connected internal nodes.

Algorithm C.1: Pseudocode for the ECoHeN extraction routines AddWellConnected and RemoveLooselyConnected.

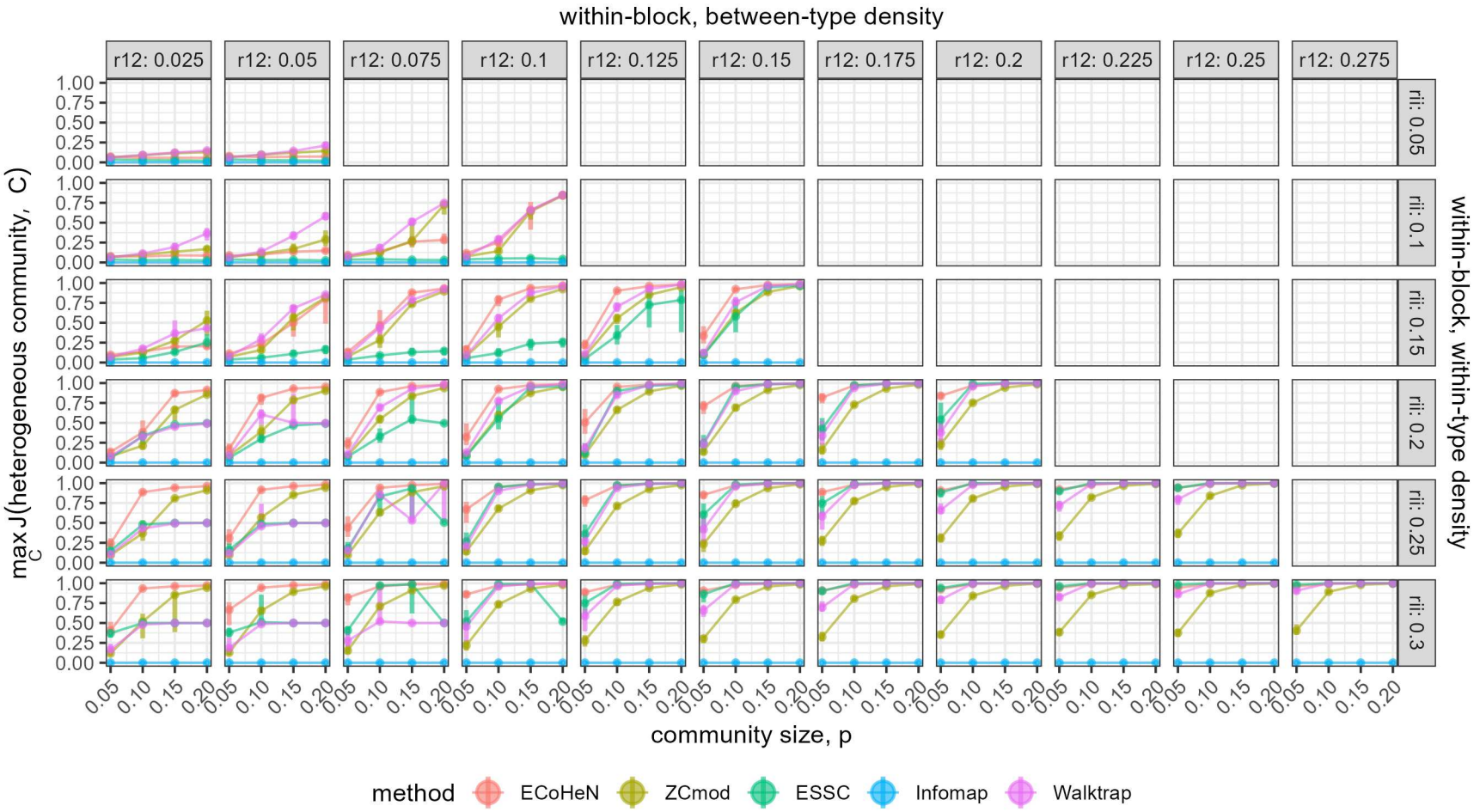


Figure C.4: Each method's ability to identify the heterogeneous community along with the size of the heterogeneous community. The vertical interval around the median represents the middle 50%. ECoHeN and ZCmod's ability to recover the heterogeneous community notably improves as the within-block, within-type density (i.e., $b + r_{ii}$) increases *and* the within-block between-type density (i.e., $b + r_{ij}$) increases. Each method poorly recovers the heterogeneous community $r_{ii} < 0.15$. When $r_{ii} < 0.15$, Walktrap generally seems preferable, while ECoHeN and ZCmod perform similarly in these conditions. When $r_{ii} \geq 0.15$, ECoHeN performs relatively better than all methods at recovering small heterogeneous communities (i.e., when p is small), especially for relatively small r_{12}.

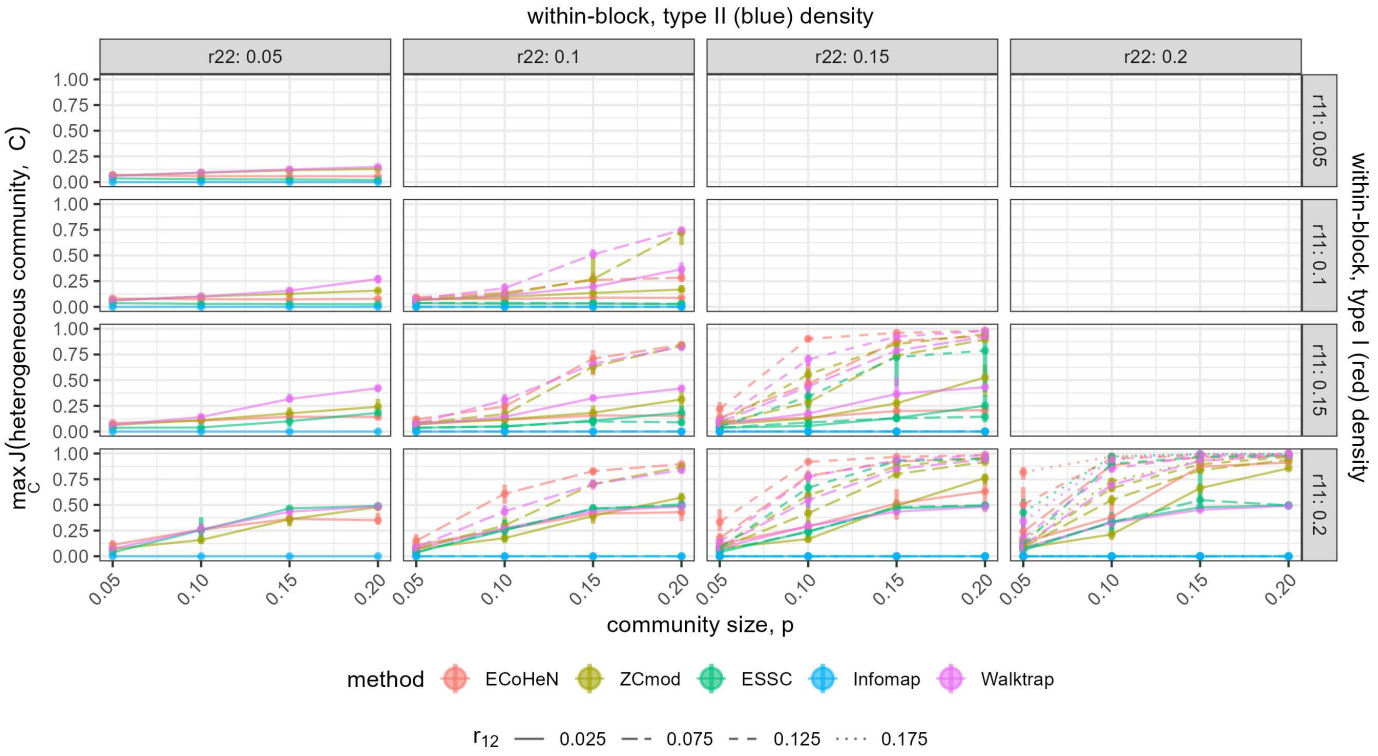


Figure C.5: The within-block, within-type (between-type) densities are provided in the facets (line type). For each community size, we show the maximum Jaccard for increasing values of r_{12} which range from 0.025 to r_{22}. ECoHeN's ability to identify the heterogeneous community drastically improves for small increases to r_{12}, outperforming each competing method at recovering small, heterogeneous communities. By comparison, ZCmod struggles to identify small, heterogeneous communities, performing well as the community size increases.

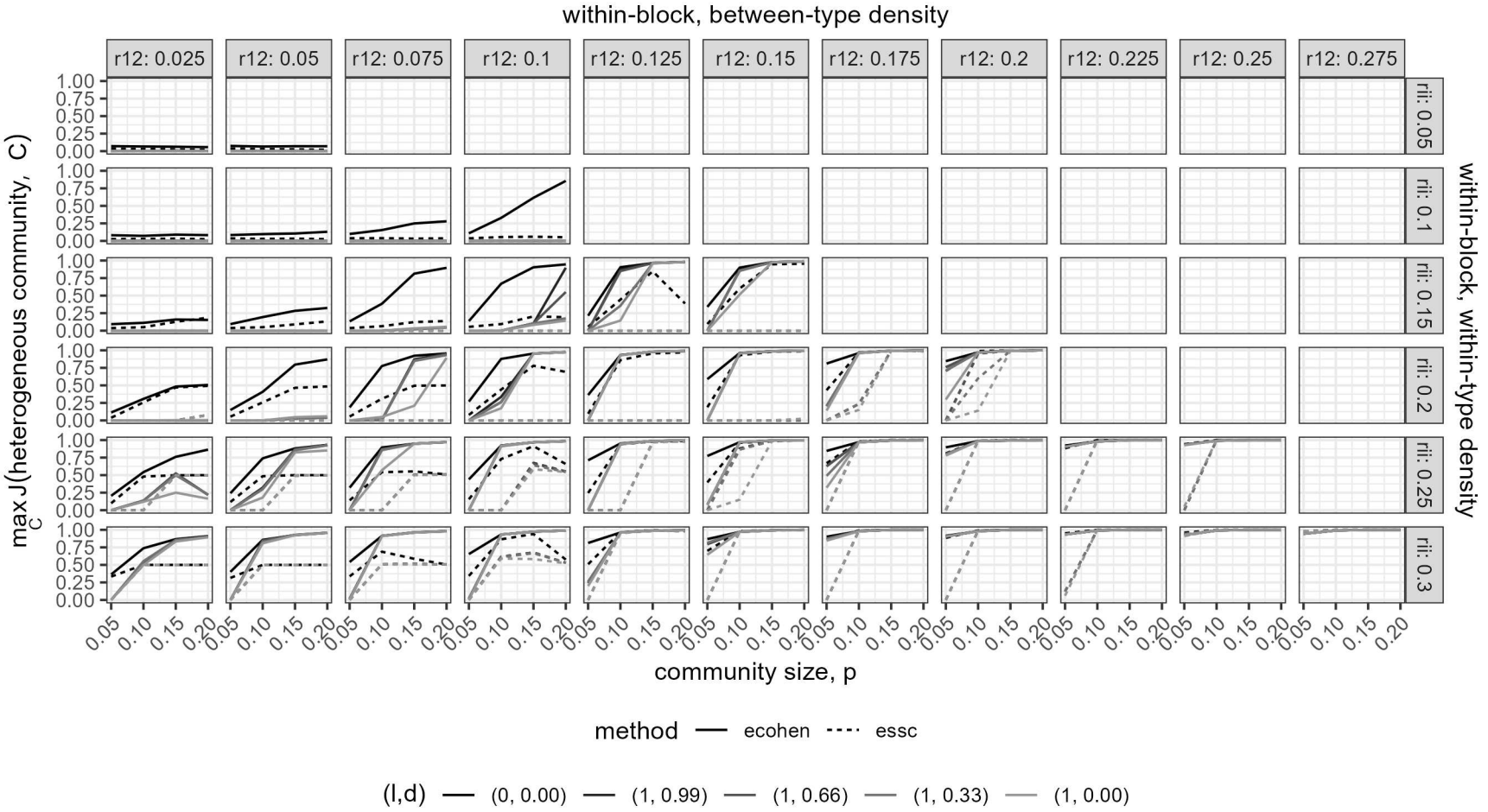


Figure C.6: Setting a maximal allowance to one for each iteration with $\xi = 0$ and $\phi = 0$ provides the best resolution for uncovering the heterogeneous community at a wide range of simulated conditions. When $\xi = 1$, a larger ϕ provides the best resolution, suggesting that if we are to speed up the algorithm by allowing a larger maximal allowance for early iterations, it is best to allow a larger maximal allowance for many early iterations. However, the effect is relatively minute for most simulated conditions, suggesting that the choice of ϕ when $\xi = 1$ will have minimal impact on the methods' ability to recover communities from background.

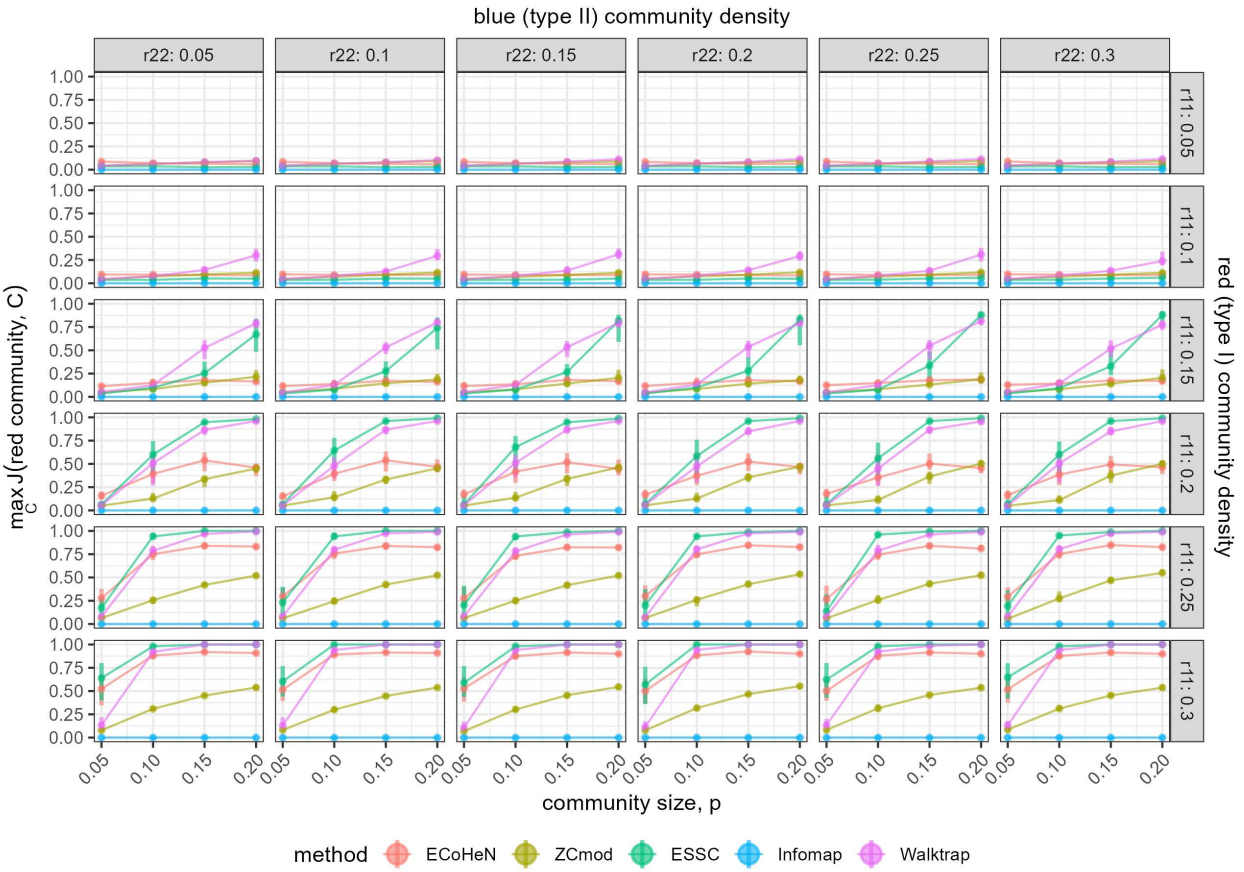


Figure C.7: The blue to blue (red to red) community density is provided on the x-axis (y-axis) facets. Each method's ability to identify the red community is provided alongside the red community size. As the red to red density increases, ECoHeN can identify the red community with increasingly better precision, marked improvements for small, homogeneous communities. There are no such improvements for ZCmod. ESSC and Walktrap consistently outperform ECoHeN and ZCmod since the between-type density is no larger than the background. In general, ECoHeN has less power in identifying homogeneous community structure than ESSC and Walktrap: a tradeoff for the flexibility of finding both homogeneous and heterogeneous community structure. Notably, the density of blue to blue connections does not impact any method's ability to identify the red community.

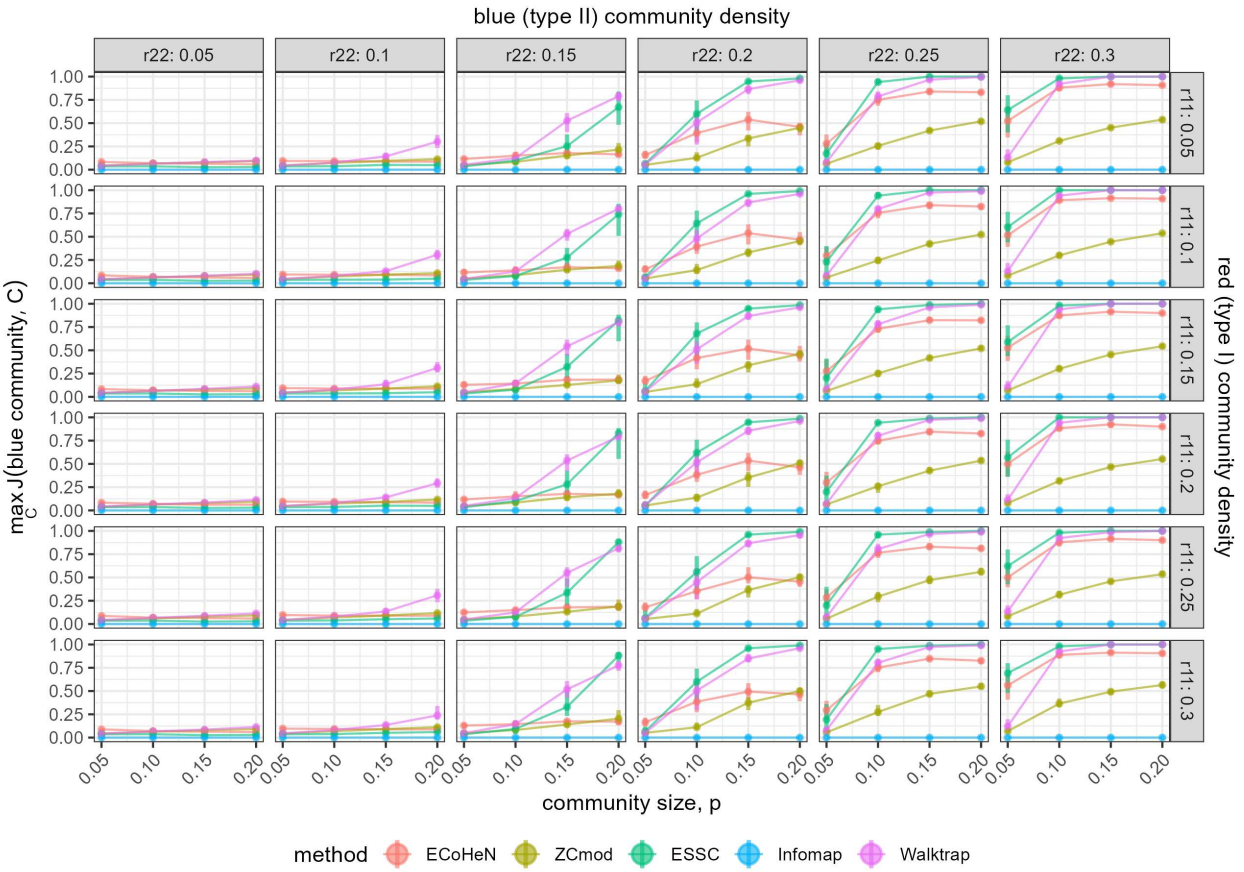


Figure C.8: The blue to blue (red to red) community density is provided on the x-axis (y-axis) facets. Each method's ability to identify the blue community is provided alongside the blue community size. As the blue to blue density increases, ECoHeN can identify the blue community with increasingly better precision, marked improvements for small, homogeneous communities. There are no such improvements for ZCmod. ESSC and Walktrap consistently outperform ECoHeN and ZCmod since the between-type density is no larger than the background. In general, ECoHeN has less power in identifying homogeneous community structure than ESSC and Walktrap: a tradeoff for the flexibility of finding both homogeneous and heterogeneous community structure. Notably, the density of red to red connections does not impact any method's ability to identify the blue community.

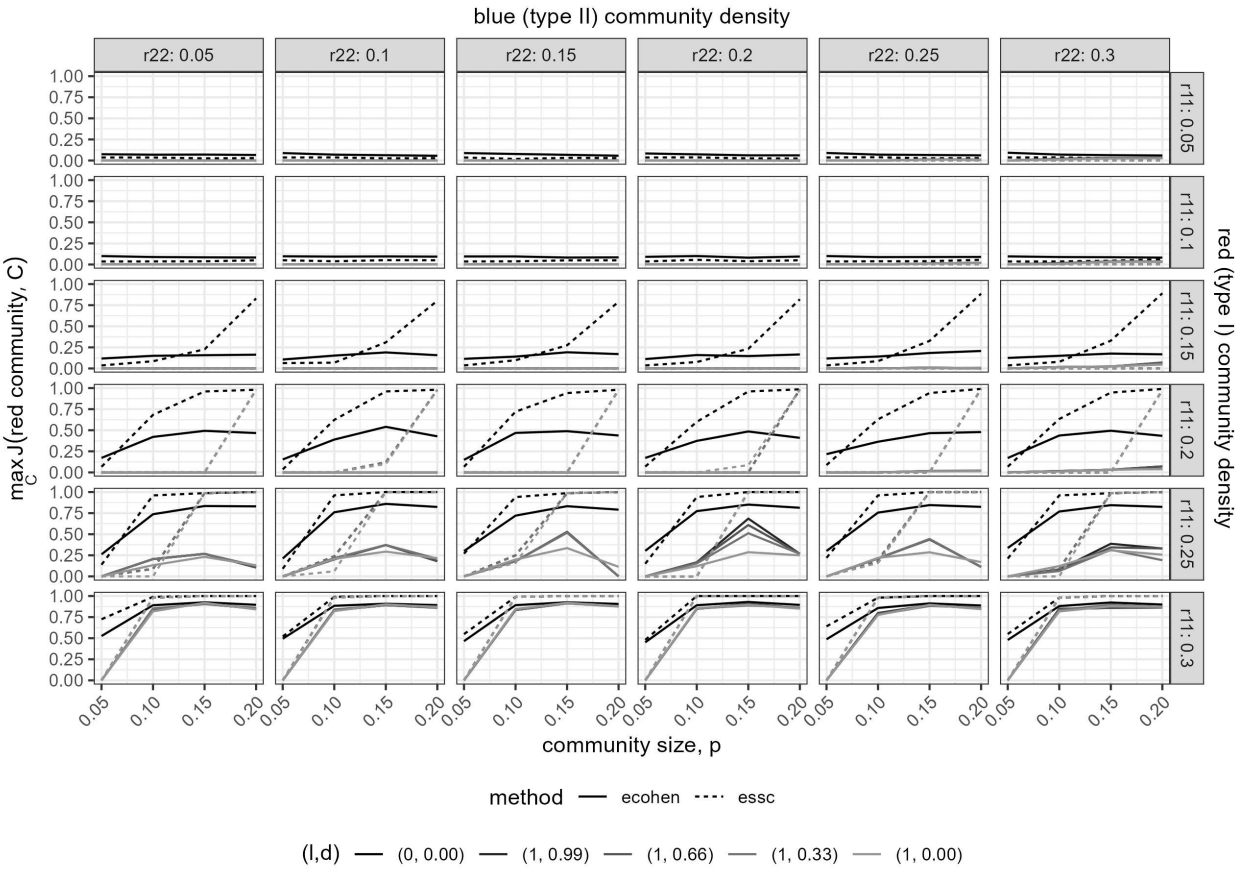


Figure C.9: The parameter setting $(0, 0)$ provides the best resolution for uncovering the homogeneous communities. When $\xi = 1$, a larger ϕ provides the best resolution, suggesting that if we are to speed up the algorithm by allowing a larger proportion of the node set to transition into and out of the candidate set, it is best to allow a large proportion to transition for more iterations.

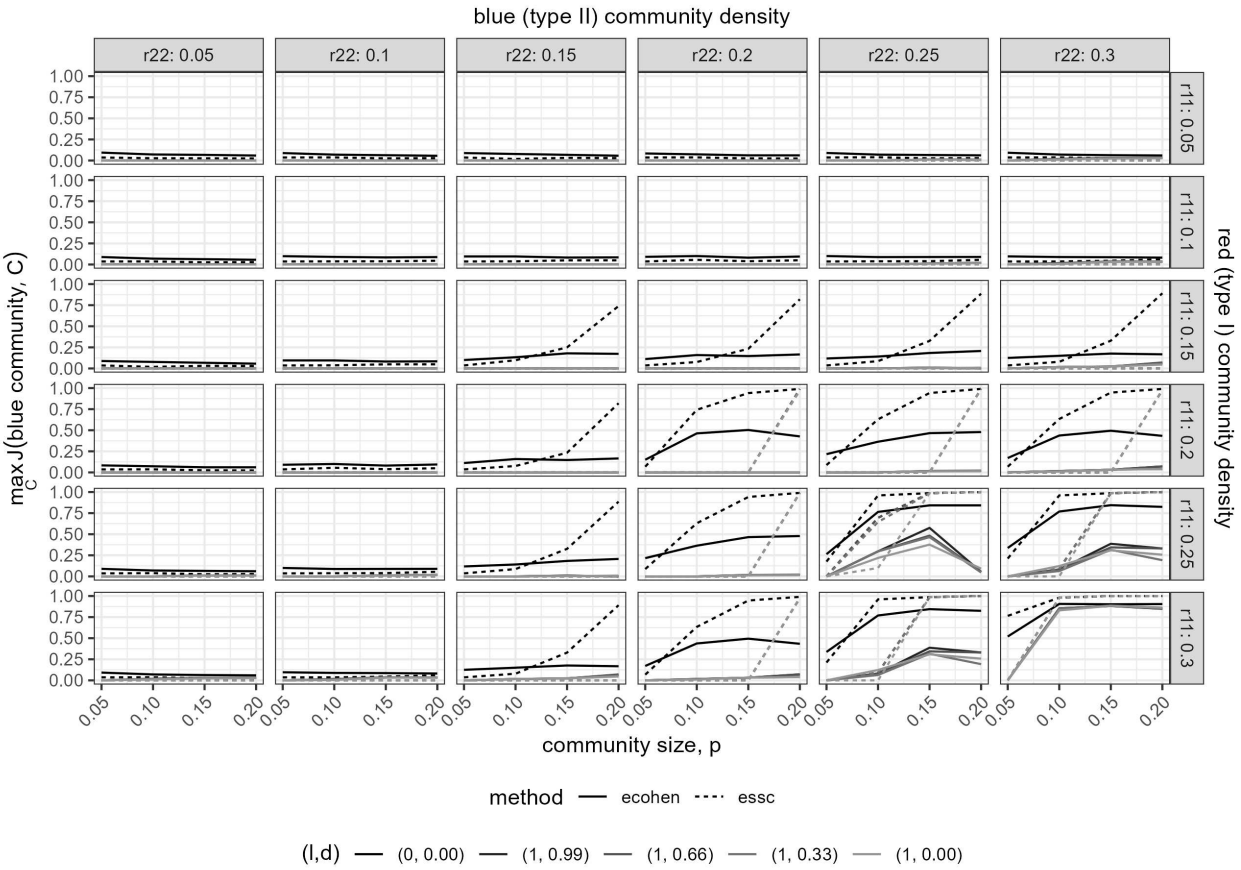
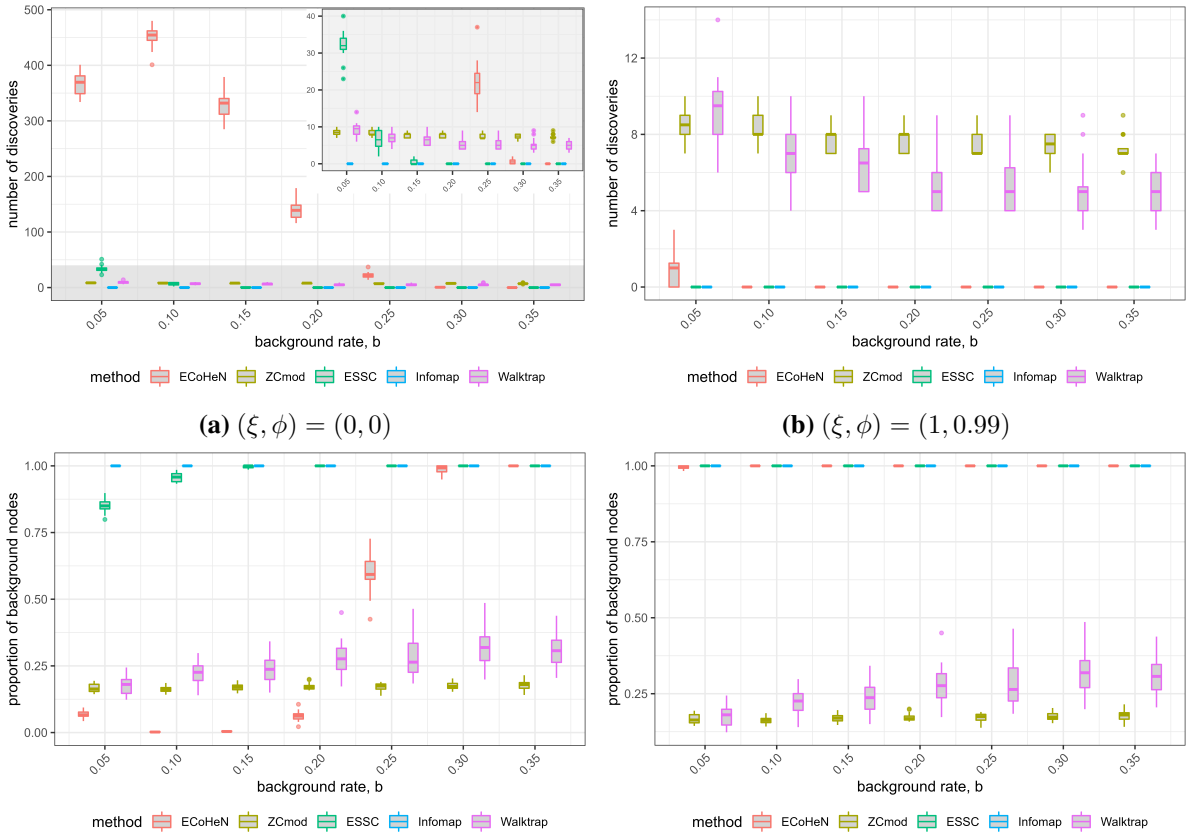


Figure C.10: The fundamental conclusions are the same as founded and discussed in Figure C.9 as they pertain to block type II (blue) nodes.



(a) $(\xi, \phi) = (0, 0)$ **(b)** $(\xi, \phi) = (1, 0.99)$

(c) $(\xi, \phi) = (0, 0)$ **(d)** $(\xi, \phi) = (1, 0.99)$

Figure C.11: The learning, ξ, and decay, ϕ, rate control the scale of changes when updating a candidate set from initialization to convergence. While the parameter setting $\xi = 0$ and $\phi = 0$ (microscopic changes) provides the best resolution for uncovering a community amongst background noise, it results in many small densely connected communities in a random network (see panels (a) and (c)). On the other hand, when $\xi = 1$ and $\phi < 1$ (macroscopic changes), ECoHeN identifies each node in a random network as background. Early macroscopic changes allow ECoHeN to break out of the low conductance initializations (see panels (b) and (d)). A practitioner's guide to selecting ξ and ϕ are provided in the "Parameter Choices" subsection of Chapter 4.

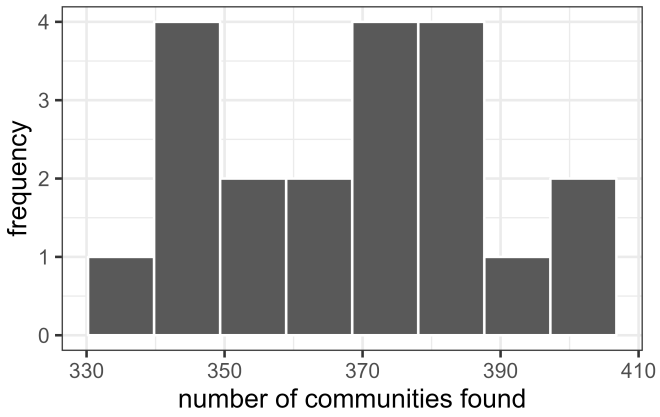


Figure C.12: When $b = 0.05$, ECoHeN uncovers between 300 and 400 communities for each of the 20 replicates.

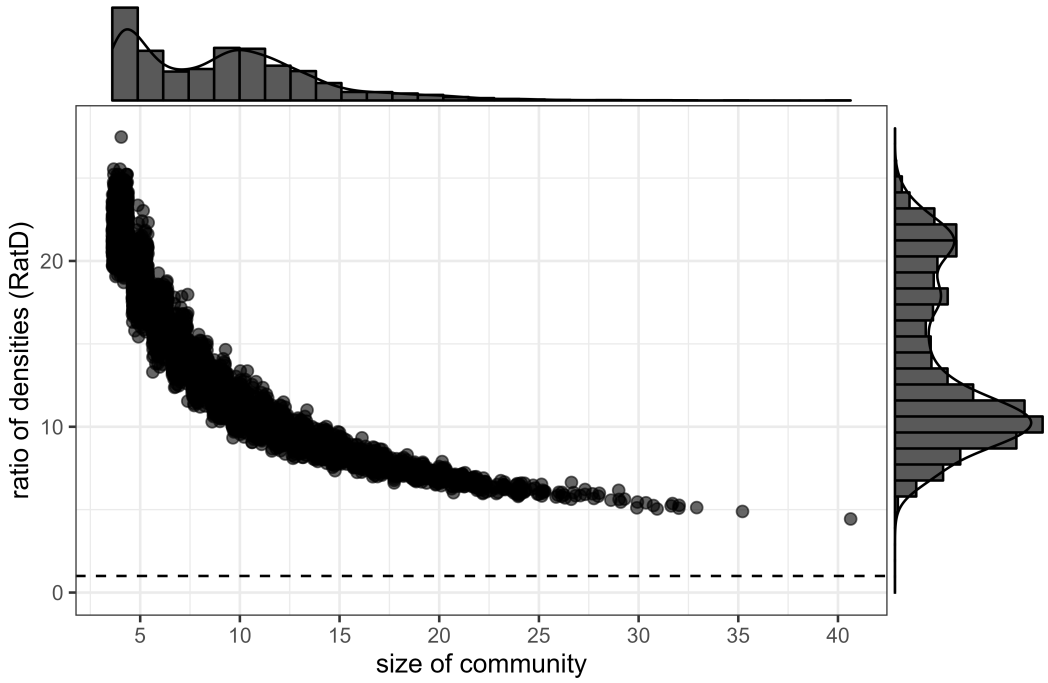


Figure C.13: Each of the communities found by ECoHeN across the 20 replicates is particularly dense. Small communities necessitate a higher ratio of densities to be deemed a community, sometimes twenty times more dense internally than to the rest of the network. The larger communities found by ECoHeN tend to have relatively smaller, albeit still large, ratio of densities, proving to be at least five times more dense internally than to the rest of the network.

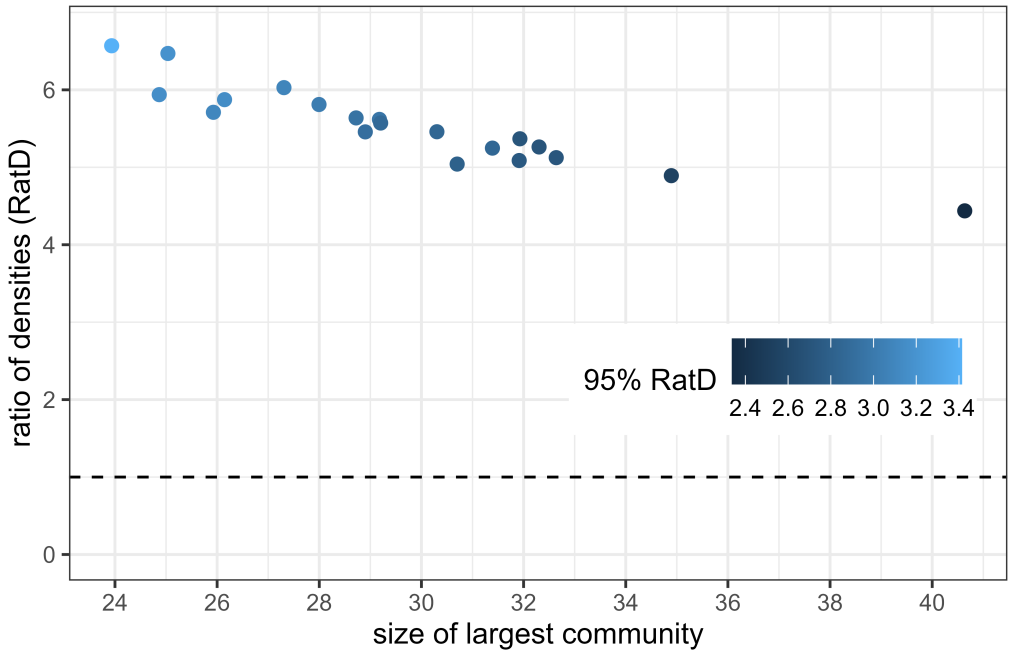


Figure C.14: The largest communities across the twenty replicates are isolated and plotted with respect to the observed ratio of densities. To gauge how unlikely it would be to attain the observed ratio of densities, we gather 1000 snowball samples for each community, compute the ratio of densities for each sample, and record the 95% quantile. Any observed value larger than the 95% quantile is deemed sufficiently dense and arguably embodies community structure (even for a random network). Each of the largest ECoHeN communities maintains a ratio of density that is roughly twice as large as the estimated respective 95% quantile.