Universiteit Utrecht

Applied Data Science (MSc)

Utrecht University, the Netherlands

MSc Thesis Dominic Comerford (5972944)

TITLE: Simulating (Multiple) Imputation in Relational Event History (REH) data: Missingness in Time, Sender, and/or Receiver

7th of July 2024

Supervisor:

Dr. Mahdi Shafiee Kamalabad

Second grader:

Dr. Gerko Vink

Word count: 6,504

Abstract

Relational event history (REH) data is a specific type of dynamic network data containing the time-stamped interactions between nodes in a network. REH data is characterized by its high resolution compared to regular network data and is increasingly available due to technological advancements. Therefore, it can potentially be crucial in investigating complex social and behavioral phenomena. The state-of-the-art method to analyze REH data is through a relational event model (REM) and managing missing data within this context is crucial as it can significantly impact the validity of results. While (multiple) imputation methods are well-regarded for their reliability, they remain underexplored within the realm of dynamic social networks, particularly in REH data. This study aims to bridge this gap by focusing on REH data to improve the robustness of REM analyses involving missing data in social network research.

By simulation and imputation of missing data in part of the Apollo 13 mission data, this study compares REM analyses of imputed data to their true and complete case analysis counterparts. Multiple imputation was employed for missingness in sender and receiver nodes, while time values were interpolated in several ways. Bias, coverage, and confidence interval width are evaluated in reciprocity, in-degree sender, out-degree receiver, and the same location statistics.

Results revealed biases in the estimated statistics. Imputed analyses showed reduced absolute and relative bias, but incorrect statistical significance compared to complete case analysis. Multiple imputation improved effect size estimation compared to complete case analysis, suggesting its potential in REH data. This finding also highlights the need for refined methods specifically tailored to imputing time data, ensuring more accurate and reliable analyses in the study of dynamic social networks.

Key words: relational event history, relational event model, social network analysis, missing data, multiple imputation, interpolation

Contents

1. Introduction

Because of the widespread occurrence of missingness in social network data the necessity of addressing missing data problems in social network analysis (SNA) is generally accepted (Kossinets, 2006; Huisman, 2009). However, current SNA often utilizes incomplete data, resulting from the exclusion of nodes (actors) or edges (associations) between nodes (e.g., applying listwise deletion). Employing such approaches may lead to biased results in SNA, even when missingness is randomized (Kossinets, 2006; Huisman, 2009). Even randomized missingness can lead to bias because the architecture of the network might change drastically even at small proportions of missingness.

Typically, SNA originates from *static* social network data that merely allows for analyzing a snapshot of that network or from panel data that allows for analyzing the same network over time in similar static snapshots (e.g., Böhnke & Link, 2017). In recent decades, however, statistical and computational advances have made it possible to model more complex network dynamics, through analyzing the network's time-ordered interactions. This type of *dynamic* data is referred to as relational event history (REH) data and the state-of-the-art method to analyze such data is through the relational event model (REM; Butts, 2008).

Three reasons together legitimize addressing the missing data problem in REH data (and REMs) in favor of more traditional social network data (and models). Firstly, REH data is one of the highest resolution and precise network data, which allows a deeper understanding of how social interactions evolve over time and for the modelling of more complex social phenomena. By maintaining the order of interactions, it is possible to include the past in the prediction of future interactions rendering it more informative than traditional SNA. Secondly, REH data is becoming increasingly available due to data being more and more recorded in a time series fashion (e.g., digital communication is often stored automatically, and updated when a new communication is sent). And thirdly, there is even less research on the missingness problem in REH compared to traditional social network data.

There are numerous options for handling missing (network) data. In many statistical software packages, the default method ignores missing values and merely uses the measured observations – referred to as complete case analysis or listwise deletion. When missingness occurs randomly, this method may produce reliable means, regression coefficients and correlations.

However, it results in overestimated standard errors depending on the proportion of missing values (van Buuren, 2018). Unfortunately, applying complete case analysis often leads to a loss of information, reduction of statistical power, and more worryingly, bias in the coefficients (van Buuren, 2018; Schafer & Graham, 2002). Other treatments of missing values involve weighting, likelihood-based procedures, and single-value - or multiple imputation and much is known about how these treatments affect the harm inflicted by missingness in various types of data (e.g., Schafer & Graham, 2002; Newman, 2014). Consequently, it is also known that multiple imputation is often a reliable method to manage missingness in statistical analyses compared to other methods (van Buuren, 2018).

However, there still is a gap in the literature on the effects of missingness and treatment thereof in social network models in general, and in *dynamic* social network models such as the relational event model (REM) specifically. The current study simulates and imputes such missingness in a section of Apollo 13 REH data containing the timestamped, chronologically ordered communications sent among ground and space crew and compares resulting analyses to their true coefficients and the complete case analysis. Researching to what extent missingness in REH data (time, sender, and/or receiver of communications) introduces bias and to what extent imputation of missing values corrects for biases are the focal questions in this study.

In the subsequent sections, the theoretical background of social network analysis, REH data and REM, as well as the implications of missing data will be discussed first. Afterwards, the data and methods, as well as the analysis strategy will be elaborated upon and then the results of the various REMs will be discussed. In the fifth and final section, the study's results will be discussed along with the study's conclusions and directions for future research.

2. Theoretical Background

2.1 Social network analysis

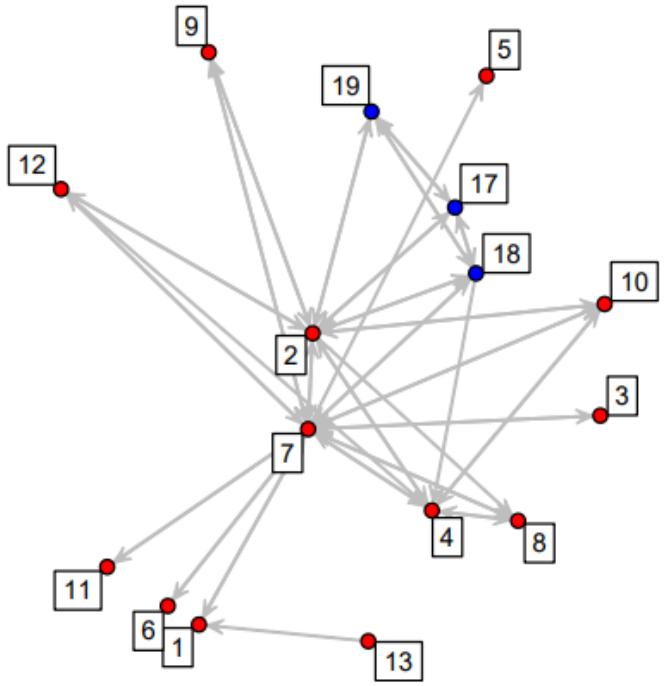
A (social) network can be defined as a collection of nodes connected through edges (Newman, 2018). The units of interest in social network analysis revolve around the relationships among nodes, such as the edges between individuals, between communities, or between other entities. An example of a social network is displayed in Figure 1, which shows the aggregated communication network among nodes in a section of the Apollo-13 mission communication data.

An arrow from one node to another indicates that the former has sent a message to the latter at least once.

Some informative network characteristics can be derived from such network graphs as it clearly shows node seven is a central unit that connects peripheral nodes and that the blue nodes form a triangle within the network. Considering node seven represents the flight director and the triplet constitutes the three astronauts, such an architecture seems plausible in this network (see Appendix A for the actor IDs and their roles).

Figure 1.

Network graph (directed) of communication between nodes in Apollo-13 data.



Note. Astronaut nodes are in blue, and ground control nodes are in red. An arrow represents whether a node communicated with a target node.

Consequently, because of its emphasis on interaction among nodes, social network analysis (SNA) requires data on the edges between nodes, and these edges can take various forms. For example, SNA may focus on friendships between colleagues or on communication instances between astronauts and ground control. Based on the characteristics of these edges and the research objectives, edges can either be directed or undirected (Newman, 2018). Undirected edges encompass mutual ties such as shared affiliations while directed edges involve a certain flow or

direction in the relationship such as a communication sent from ground control to an astronaut at a certain time point.

This focus on edges and nodes contrasts with traditional studies that focus on individual attributes to understand behavior (McGloin & Kirk, 2014). An individual attribute could be a measure of a person's communication skills and researchers are then interested in associations between variables. In contrast, the occurrence or intensity of communication between individuals represents a relational event which can be modelled to analyze the evolution of interaction within a network.

SNA based on such relational events implicitly suggests that the quantity and type of connections among nodes can be informative explanatory factors in predicting future events (McGloin & Kirk, 2014). In other words, the relational history in the data may be predictive of further network dynamics. In the context of the Apollo-13 mission, past communication may be an informative characteristic in predicting further communication dynamics within the Apollo 13 network. For the dynamics in these relational event data to be analyzed a specific type of network model needs to be employed – the relational event model.

2.2 Relational event history data and relational event models

Relational events can be understood as actions that occur as discrete events at a certain point in time where one node exhibits a behavior targeted at one or multiple other nodes in the network (Bates & Harvey, 1975; Butts, 2008). A sequence of those discrete events, in continuous time, is then described as relational event history (REH) data and encompasses at least the times or order of events, and dyads of sender and receiver nodes (Butts, 2008). Table 1 entails the first two and last two cases of the Apollo 13 REH data, as each row represents a discrete time-stamped event where a message is sent from a sender to a receiver node. Here, the sender column shows what node sent the message, while the receiver column contains the target of that message.

Table 1.

Relational events of Apollo-13 communication.

Time	Sender ID	Receiver ID
11849.2	18	2
11854.2	2	18
…	…	…
50012.8	7	4
50014.8	4	7

Note. Total number of recorded events is 3882 among 16 nodes. Time is in seconds from the onset of the mission.

The relational event model (REM) is a gold standard for analyzing REH data and provides a framework for modelling the predictors (statistics) that explain how interactions or relationships evolve. In this model, events occur at discrete moments in time and thus have a well-defined duration, and ties between nodes exist in these short moments in continuous time and dissolve after. These points imply that there is a clear understanding of the order and duration of interaction over time and allow for examining the expected dyad of nodes that will communicate, and the expected time till a relational event occurs (Butts, 2008). Note that a static (panel) network model differs herein as in the REM ties are short-lasting and exist at exact time points.

In a REM the time between events with exponential distribution is modelled with rate parameter λ. The events rate, λ, shows the propensity of an event to occur which determines which nodes will interact *and* when this interaction will occur (Butts, 2008; Meijerink-Bosman et al., 2023). It is assumed that λ is a log-linear function of exogenous and endogenous statistics. Exogenous statistics entail characteristics such as ‘age’ or ‘location’ of individual nodes or edges and allow for researching to what extent certain attributes determine the event rate. In contrast, endogenous statistics encompass the likelihood of potential subsequent events conditional on past events, such as a dyad's prior communication. The event rate can then be modelled as the outcome, regressed on by predetermined statistics in a log-linear function (Meijerink-Bosman et al., 2023):

$$\log \lambda\,(s,r,t) = \sum \beta_p X_p(s,r,t),$$

where β_p refers to the impact of the *p*-th statistic $X_p(s,r,t)$ on the event rate. Consequently, by estimating the model parameters, β_p, linked to exogenous and endogenous statistics inferences can be made on the occurrences and dynamics of communication within a network over time (Meijerink-Bosman et al., 2022).

To estimate the event rates, it is first necessary to construe a risk set entailing all possible events that might occur, resulting in a matrix of all conceivable dyads. In the context of *directed* edges of a sending node, defined as *s* and a receiving node, defined as *r* at time-point *t*, the matrix *s* times *r* represents all possible relational events at time-point *t*. Thus, the Apollo-13 communication risk set comprises *N* (*N* - 1), or (16 x 15 =) 240 potential events at each time-point, where *N* represents the number of nodes in the network.[1]

Second, the likelihood of an event (*s*, *r*, *t*) to occur is equal to the occurrence rate of that event relative to the sum of rates for all events in the risk set at that time point (Butts, 2008). This rule ensures more common events are assigned higher event rates compared to the less common events, and can be defined as:

$$P\big((s,r)\big|t\big) = \frac{\lambda(s,r,t)}{\sum\lambda(s,r,t)}.$$

2.3 Missing data

There could be numerous reasons that social network data, including REH data, are incomplete and these include but are not limited to respondent inaccuracy, non-response, and technological failures (Kossinets, 2006; Kiang et al., 2021). For example, nodes might falsely portray the absence of edges to other nodes, nodes might not respond at all, or data might go missing due to electronic malfunctioning. The mechanisms by which missingness occurs can vary too and in the literature are described as Not Data Dependent (NDD), Seen Data Dependent (SDD), and Unseen Data Dependent (UDD) (Rubin, 1976; van Buuren, 2018).[2]

[1] For practical reasons it is assumed in the current study that a node cannot send messages to multiple other nodes simultaneously however it is possible to model such interaction in a REM.

[2] NDD, SDD and UDD are typically referred to as Missing Completely at Random (MCAR), Missing at Random (MAR) and Missing Not at Random (MNAR), respectively. However, in the current study, the 'data dependent' terminology from Hand (2020) is used as it directly conveys the missingness mechanism at play.

NDD describes situations where the probability of missingness is equal across all cases, meaning that the research questions we pose to answer are unrelated to the distribution of the missing values. Consequently, beyond the loss of information, various complexities stemming from such missing data may be overlooked. In contrast, in situations where missingness is affected by either observed (SDD), or unobserved (UDD) characteristics of the data, the research questions of interest *are* related to the missingness. Hence, making inferences from subsequent analyses requires more critical evaluation than in an NDD context (van Buuren, 2018).

Because the current article is exploratory in terms of imputing missing values in relational event history (REH) data, only NDD is further described as it serves as the benchmark against which imputation should be evaluated. In other words, if imputation is not satisfactory in the more convenient NDD context, it will likely also not be in one defined by the more problematic contexts of SDD or UDD (van Buuren, 2018). Mathematically, the NDD situation can be formulated as:

$$\Pr(R = 0|Y_{obs}, Y_{mis},\psi) = \Pr(R = 0|\psi).$$

Here, Y is a matrix composed of Y_{obs} and Y_{mis}, or the observed and missing values, R represents a missingness matrix in which each cell indicates whether the aligning cell in Y is missing (0) or observed (1), while ψ encompasses the missing data model parameters such as the probability for a missing value to occur. So, the probability of data being missing in an NDD context depends on ψ, the general missingness probability, as each value has an equal chance to be missing, rather than on Y_{obs} or Y_{mis}. In sum, NDD is a mechanism resulting in missingness to occur randomly across the data.

Most social network analyses ignore the problem of missingness by analyzing complete cases while others transform missing edges between nodes to be non-existing edges, which can lead to biased inferences (Gile & Handcock, 2017). A more truthful method to handle missing data is through multiple imputation as it acknowledges the uncertainty and variance surrounding missing values. By creating multiple versions of the data through the replacement of a missing value by a plausible one, multiple imputation allows for analyzing each imputed dataset individually before merging the estimates (van Buuren, 2018).

2.4 This study

This study aims to analyze how (multiple) imputation of missing values affects analyses of REH data. Missingness is simulated through the NDD mechanism in part of the Apollo-13

communication data and is allowed to occur in the time, sender, or receiver information or in any combination of this information. Subsequently, missing values are imputed through (multiple) imputation. Next, REMs of imputed datasets are compared to the analysis utilizing the fully observed communication data (the true estimates) and the complete case analysis (the default method to handle missing values in many applications) In doing so, the current research improves our understanding of 1) to what extent missing data occurring randomly, specifically missingness in time, biases results in REMs, and 2) to what extent current techniques for imputing missing data may help correct for bias.

3. Data & Methods

3.1 Data

Following Shafiee Kamalabad et al. (2023), communication data from the Apollo 13 mission, as REH data, was used for the empirical analyses, specifically from the time surrounding the iconic phrase "Houston, we've had a problem."[3] This 'problem' occurred fifty-six hours into the mission and referred to an exploded oxygen tank. At that moment, the mission turned from a routine journey destined to the moon into a mission to solve life threatening issues and safely return the astronauts to Earth. Luckily, they did in the end - aided by clear communication within the network.

In this study part of the Apollo 13 communication data is analyzed, focusing on the sequence of communications that occurred within the network. As such, the relational events are time-stamped directed communications from a sender to a receiver node (see Table 1). Note that time is in seconds from the onset of the mission, and the sender and receiver columns contain the ID rather than the role of the nodes (see Appendix A for more information). After selecting the Apollo 13 communications from around an hour before to approximately six hours after the moment the tank exploded, a dataset of 3882 relational events among 16 nodes remained.

3.2 Measures

In this study, we focus on the impact of missing data and how to make valid inferences through (multiple) imputation in REMs rather than analyzing the exact content of communications.

[3] The complete communication transcript can be retrieved from http://apollo13realtime.org/. The used subset of Apollo 13 is web-scraped and does not have any privacy or ethical limitations.

Therefore, the content of the communications is excluded but like Shafiee Kalamabad et al. (2023 the endogenous statistics; reciprocity, indegree sender, and outdegree receiver are included. Also, as an exogenous statistic, it is considered whether a sender and receiver of a communication are in the same or different locations. Including these statistics allows for modelling network dynamics based on past relational events as well as disentangling location effects on communication and comparing the extent to which results from various models differ.

Reciprocity. The reciprocity statistic assumes there is a tendency for node 1 to reciprocate communication to node 2 if node 2 has contacted node 1 in the past. Indeed, studies have shown that nodes that have received communications are likely to reciprocate these in the future (Stadfeldt & Block, 2017; Shafiee Kalamabad et al., 2023).

In-degree sender. The in-degree of the sender statistic refers to the number of communications a node has received and assumes that those with a higher in-degree have a higher likelihood of initiating contact in the future (Butts & Marcum, 2017). For instance, when node 1 receives relatively many communications up to a certain time-point, it is expected that node 1 has a relatively high probability of initiating contact in the future compared to a node that has received fewer communications.

Out-degree receiver. The out-degree of the receiver statistic refers to the number of communications a receiver node has transmitted, and it is assumed that nodes with a higher out-degree have a higher likelihood of being contacted in the future (Butts & Marcum, 2017). For instance, when node 2 sends relatively many communications up to a certain time-point, it is expected that node 2 has a relatively high probability of being contacted in the future themselves compared to a node that has sent fewer communications.

Same location. The same location statistic reflects an exogenous attribute that a sender and receiver's rate of interacting is determined by whether the dyad shares the same location. In other words, whether the nodes are in the same group – astronauts or ground control. A binary variable was coded where intragroup communication among the space crew (node 17, node 18, and node 19) and ground control (the other actors) was assigned a 1, and intergroup communication a 0. A negative effect of 'same location' reflects that nodes who are in the same location initiate future communication with a lower event rate compared to nodes in different locations. Because there is a clear structure and hierarchy before sending a message from ground control to the astronauts it

is expected that 'same location' will have a negative effect on the event rate. Ground crew, as well as the astronauts, are likely to communicate off the record before sending a coordinated message via the Apollo 13 channel.

3.3 Analysis strategy

The analysis can be divided into three main components, and these are 1) missingness amputation, 2) missingness imputation, and 3) the REM analyses.

Missingness amputation. The first step in the analysis was to create missingness through the NDD mechanism in the Apollo 13 data in 100 simulations. To include every node in each simulated dataset the same 1500 relational events were drawn and preserved from the complete data and included in each simulation. Part of the data was preserved because some nodes occurred rarely and would have been removed altogether in some simulations by amputation over all relational events. Therefore, preserving some communication resulted in the same number of nodes (16) in each simulation, and thus equal risk set sizes (240), a requirement for comparisons of different REMs.

For robustness, the amputation was done 100 times by creating various versions of Apollo 13 data with 40% of rows containing at least one missing value in the remaining data. So, 40% of these rows had either one, two or three missing values in time, sender and/or receiver columns. This proportion allowed for stable enough analyses under a substantial amount of random missingness.

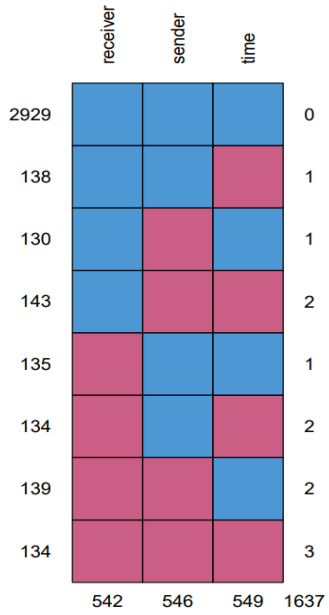
As an example, Figure 2 contains the missingness patterns of the first simulated dataset. Blue cells indicate that a value has been observed, whereas red represents a missing value. The stacked rows then represent the various missingness patterns. The patterns show that in the first simulation, 2929 relational events are completely observed and seven patterns of missingness exist that occur from 130 to 143 times each.[4] Note that a pattern where all columns are missing – the bottom pattern - could occur in REH data. Communications with missing time, as well as missing sender, and receiver nodes, are still transmitted and logged as a relational event. Here, it is unknown what the time and sender and receiver nodes of this communication are but that an event

[4] The *MICE* package in R (van Buuren & Groothuis-Oudshoorn, 2011) was used to create missingness as well as to impute missingness in the data in the next step.

occurred *is* known because a relational event is stored automatically. In the first simulated dataset the number of communications with completely missing information except for the rank is 134 times.

Figure 2.

Missingness patterns of the first simulated dataset



Note. Rows represent different missingness patterns whereas columns represent receiver, sender, and time information. The number on the left stands for how often a pattern occurred, the number on the right how many missing variables occur in the corresponding patterns, and the numbers below how often that column is missing in the data in total. Blue cells indicate an observed while red cells indicate missing values.

Missingness imputation. The second step of the analysis was to impute the missing values through (multiple) imputation. This required considerations regarding the number of imputations, the number of iterations, the methods employed and the specification of predictors for the imputation model. Imputation refers to how many times the missing values are to be imputed, while the number of iterations refers to how often each imputation is updated to calculate the eventual imputation. It was opted to set the number of imputations and iterations both to five for computational efficiency. Furthermore, research shows this number is often sufficient for reliable imputations (van Buuren, 2018).

The employed methods differed per missing variable, as time was imputed by interpolation using the values before and after the missing times rather than multiple imputed. This was done in

three ways, as time was interpolated linearly at first, but in two additional analyses time was interpolated using the spline and Stineman methods. Spline interpolation fits a polynomial function to the data, whereas the Stineman algorithm interpolates by using linear segments to adjust the smoothness of interpolation rather than polynomials (Stineman, 1980). Interpolation of time might in some cases be a realistic option for REH data with missingness in times as specific times may be lost but due to data being stored chronologically, the order might still be preserved.[5]

In contrast, missing sender and receiver data were multiple imputed through predictive mean matching. Predictive mean matching interpolates from a pool of candidate donors that are most like the missing observation in other variables and one or more of such observations are used to determine the imputed value (van Buuren, 2018). Consequently, such 'regular' predictive mean matching can result in an imputed node seemingly communicating to themselves whereas the Apollo 13 data does not allow for such communications. Therefore, a custom method was employed that prevents imputed data from containing nodes that communicate with themselves (Vink, 2023). In all multiple imputation models, the target variables were regressed on both remaining variables. For instance, in the sender imputation model both time and receiver were used as predictors of missing sender values.

Analyses. The third step of the analysis was conducting the REMs and comparing their performance. The statistics for the models were computed using the package *remstats* (Meijerink-Bosman et al., 2024), while the REMs were conducted with *remify* (Arena et al., 2024) using the Cox proportional hazard function from the *survival* package (Therneau, 2023). The REMs in combination with the obtained statistics output the number of events and nodes, the amount of time that passed from the first to the last event and well as the estimated parameters for each statistic of interest.

The Apollo 13 mission data used in this paper is assumed to be the complete population. Therefore, the variance around the statistics in the simulations is calculated in alignment with Vink and van Buuren (2014), who would argue that because we have a finite population from which we draw samples (the sub section of Apollo 13 data) we can treat the fully observed data as the 'truth'. Subsequently, this finite population feature removes sampling variance in calculating the total

[5] Interpolation was conducted through the *imputeTS* package in R (Moritz & Bartz-Beielstein, 2017).

variance in the statistics, resulting in substantially smaller estimates for standard errors in REMs that utilize the multiple imputed data.

That said, the REM on the fully observed data was conducted first as this constitutes the 'truth'. Thereafter, the REM was conducted as a complete case analysis pooled over the 100 simulations and compared to its fully observed counterpart. Finally, the REM was analyzed over the imputed datasets and its pooled result was compared to the previous analyses.

Performance evaluation of the imputed simulations compared to the true coefficients was done based on Oberman and Vink's (2023) recommendations. These recommendations include analyzing the amount of bias in the estimate, and the coverage rate (CR). The bias of a coefficient represents how far off from the true coefficient the estimation is, while coverage represents how often the true coefficient is in the range of the estimate's confidence interval. The proportion of bias (PB) in the coefficients should ideally be lower than five percent, while the coverage should be at least 95%. A small amount of bias with sufficient coverage would indicate that the estimation method is a viable method for imputation and yields valid inferences. Moreover, the average width (AW) of the 95% confidence intervals is used for evaluation, where more narrow intervals in combination with sufficient coverage suggest less uncertainty in the estimates than wider ones (Oberman & Vink, 2014).

4. Results

4.1 Descriptive statistics

Table 2 shows descriptive statistics of the static Apollo 13 network; so, for now, the emphasis is on whether a dyad communicated at all and not on the intensity of contact. The number of nodes is 16 of which 3 are astronauts. Also, the number of communications is 3882 of which 1813 are sent to the same location and 2069 between locations. Furthermore, characteristics derived from the network in Figure 1 are included. The density, or the number of connected nodes over all possible nodes, is .21, while the longest shortest path spans 4 nodes. The average closeness centrality is .68, implying most nodes seem to be close to other nodes while the average eigenvector centrality or the average centrality based on neighbors' influence is also high, reflected in the score of .61. The average shortest path in the network is 1.92, meaning that any pair of nodes, on average, is separated by less than two nodes. Relatedly, the average betweenness, or the

extent to which nodes fall on another dyad's shortest path is .57, while transitivity, or the overall closure of triplets into connected triangles is .33.

Table 2.

Network characteristics of complete Apollo 13 data.

	Amount	Network-level	
Nodes	16	Density	.21
Astronauts	3	Diameter	4
Ground control	13	Closeness	.68
Communications	3882	Eigenvector	.61
Same locations	1813	Average shortest path	1.92
Different locations	2069	Betweenness	.57
		Transitivity	.33

4.2 Fully observed data

The results of the REM utilizing the fully observed Apollo 13 data can be found in Table 3. Firstly, it can be derived that the total events in the data used for the Cox proportional hazard functions is 931.680. This number amounts to the product of the number of events, 3882, multiplied by the number of possible pairs, or the risk set at each timepoint (16 x 15). Secondly, it shows that reciprocity has a small positive effect on the event rate although this is not statistically significant ($\beta = 2.332^{-02}$, $p = .209$). Nodes do not seem to return past communications in this network. Perhaps the hierarchical nature of this network with strict communication guidelines inhibits reciprocal communication. Thirdly, a sender's past in-degree positively affects the likelihood of an event happening and this effect is statistically significant ($\beta = 4.314^{-04}$, $p < .001$). Receiving a larger number of communications results in a higher likelihood of becoming a future sender. Fourth, the out-degree of the receiver has a small negative and statistically insignificant effect on communication happening ($\beta = -9.023^{-05}$, $p = .225$). This coefficient implies that the number of communications a node has sent does not determine whether that node will be a receiver of future communications.

Finally, there is a tendency to engage in contact with nodes that are in a different location. Whether a pair of nodes are in a different location proves to be a strong predictor of future communication, as those in a different location are more likely to engage in contact with each other through the Apollo 13 channel (β = -.863, p < .001). This might be explained considering ground crew possibly talks to each other outside of the mission's channel before a final message is sent to the astronauts, and vice versa.

Table 3.

REM Results for the fully observed Apollo 13 data.

Statistic	β	p-value
Reciprocity	2.332^{-02}	.209
	(1.856^{-02})	
In-degree sender	4.314^{-04}	< .001
	(7.398^{-05})	
Out-degree receiver	-9.023^{-05}	.225
	(7.437^{-05})	
Same location	-.863	< .001
	(.032)	

Note. Number of possible events = 931.680, number of events = 3882. Standard errors in parentheses. BIC = 98241. The number of simulations is 100.

4.3 Complete case analysis

Table 4 contains the results of the aggregated complete case analysis over 100 simulations. Compared to their true counterparts the coefficients for reciprocity ($\beta = 2.773^{-02}$, p = .196), in-degree sender ($\beta = 5.679^{-04}$, p < .001) are somewhat overestimated whereas out-degree receiver ($\beta = -1.208^{-04}$, p = .293) is somewhat underestimated. Also, same location seems to be relatively biased as its estimate is severely underestimated in the complete case analysis (β = -1.349, p < .001). Unsurprisingly in this complete case analysis where missingness occurred randomly; all

standard errors are larger than their true counterparts, which implies there is more uncertainty in the estimates.

Table 4.

Aggregated REM results for complete case analysis.

Statistic	β	*p*-value	Bias
Reciprocity	2.773^{-02}	.196	4.413^{-03}
	(2.121^{-02})		
In-degree sender	5.679^{-04}	< .001	1.364^{-04}
	(1.126^{-04})		
Out-degree receiver	-1.208^{-04}	.293	-3.054^{-05}
	(1.129^{-04})		
Same location	-1.349	<.001	-.486
	(.039)		

Note. The total risk set size ranges from 690.720 to 717.360, and number of relational events ranges from 2878 to 2989 across simulations. Standard errors in parentheses. The average BIC is 71786. Number of simulations is 100.

4.4 Imputed simulations

Table 5 shows the aggregated REM results over 100 simulations where time was imputed via linear interpolation and sender and receiver through multiple imputation. Results indicate that reciprocity has a small positive effect, and this is statistically significant (β = 2. 2.516^{-02}, $p < .001$, 95% CI = [2.217^{-02},2.814^{-02}]). Similarly, receiving more communications now positively predicts sending communications in the future ($\beta = 4.177^{-04}$, $p < .001$, 95% CI = [3.901^{-04}, 4.453^{-04}]), while sending more communications has a small negative and statistically significant effect on being a receiver of future communications ($\beta = -9.313^{-05}$, $p < .001$, 95% CI = [1.044^{-05}, -8.187^{-05}]). The

coefficient for same location is still underestimated but the bias is smaller than in the complete case analysis (β = -.910, p < .001, 95% CI = [-.943, -877]).[6]

Table 5.

Aggregated REM results after imputation of missingness.

Statistic	β	p-value	*CI-95%* [LB, UB]	CR	Bias	PB	AW
Reciprocity	2.516^{-02} (1.076^{-03})	< .001	[2.217^{-02}, 2.814^{-02}]	.75	1.834^{-03}	7.866	5.975^{-03}
In-degree sender	4.177^{-04} (9.946^{-06})	< .001	[3.901^{-04}, 4.453^{-04}]	.88	-1.373^{-05}	3.306	5.523^{-05}
Out-degree receiver	-9.313^{-05} (4.053^{-06})	< .001	[-1.044^{-04}, -8.187^{-05}]	.89	-2.894^{-06}	4.523	2.251^{-05}
Same location	-.910 (1.181^{-02})	< .001	[-.943, -.877]	.17	-4.761^{-02}	5.517	6.560^{-02}

Note. Number of possible events = 931.680, number of events = 3882. 'Time' is imputed as a single value by interpolation. 'Sender' and 'Receiver' are imputed through multiple imputations in MICE. Standard errors in parentheses. The number of imputations is 5 in each simulation. The number of simulations is 100.

There are both similarities and differences when comparing the imputed simulation results to the fully observed REM. The effect sizes are like the fully observed REM, leading to only marginal absolute and acceptable relative bias as the estimated coefficients are close to the truth. Considering the missingness mechanism was NDD, only a relatively small amount of bias was anticipated because the missingness occurred randomly across the data and was not associated with any observed or unobserved characteristics of the data. The bias introduced by these simulations is smaller than in complete case analysis because the latter discards around 20% of the

[6] As a sensitivity analysis, the REM was conducted on simulations where 'same location' was used as an additional predictor in multiple imputation models. This analysis yielded similar conclusions, but coverage was substantially better for same location at .51. See Table 7 in Appendix B.

data resulting in less accurate estimates in the latter method. Furthermore, the relative biases range from 3.306% to 7.866% which would imply an acceptable amount of bias.

However, the standard errors in the imputed simulations are substantially smaller than in the fully observed scenario, resulting in statistically significant effects for reciprocity and out-degree receiver statistics that were not found in the fully observed REM (and not in the complete case analysis either). Using the finite population results in a smaller variance because sampling variance is not included in estimating the total variance. Absence of sampling variance results in substantially smaller stander errors. Consequently, the confidence intervals of the statistics in the imputed data analysis are narrow as well, as showcased by the narrow average widths (AW) across the simulations. Unfortunately, the coverage rates (CR), or the proportion of times the 95% confidence intervals include the 'true' value are suboptimal. In only 75% of the confidence intervals for reciprocity does the truth fall within the boundaries, while this reaches 88% and 89% for the in-degree sender and out-degree receiver statistics. Same location only contains the true coefficient in 17% of confidence intervals because of the severe bias in this statistic and small standard error. Ideally, these rates should at least be 95% (Oberman & Vink, 2023). This underperformance implies that the current imputation procedure may lead to invalid inferences when caution is not preserved regarding the standard errors of the effect sizes. Although absolute (and relative) bias is smaller than in the complete case analysis the smaller standard error may falsely suggest statistically significant results.

4.5 Time imputed with spline and Stineman interpolation

Two additional REMs were conducted where time was first interpolated through spline interpolation and then according to the Stineman algorithm instead of the previous linear interpolation of time. Table 6 contains the results of these REMs and conclusions remain like the main analysis. Reciprocity and out-degree of the receiver become statistically significant while all standard errors become substantially smaller. The small standard errors also result in similar confidence intervals and their average widths, as well as comparable absolute and relative bias to the main analysis with linear interpolation. Using the spline method improves reciprocity and decreases out-degree receiver and same location coverage while in-degree sender remains the same. At the same time, the Stineman algorithm slightly improves the coverage rates for reciprocity and out-degree receiver but decreases it for in-degree sender while same location is the same.

Table 6.

Aggregated REM results after imputation of time spline and Stineman interpolation.

Statistic	β	p-value	*CI-95%* [LB, UB]	CR	Bias	PB	AW
			Spline				
Reciprocity	2.514^{-02} (1.057^{-03})	< .001	[2.220^{-02}, 2.807^{-02}]	.78	1.817^{-03}	7.826	5.869^{-03}
In-degree sender	4.184^{-04} (1.062^{-05})	< .001	[3.887^{-04}, 4.478^{-04}]	.88	-1.310^{-05}	3.194	5.897^{-05}
Out-degree receiver	-9.320^{-05} (4.188^{-06})	< .001	[-1.048^{-04}, -8.157^{-05}]	.87	-2.965^{-06}	4.928	2.326^{-05}
Same location	-.910 (1.081^{-02})	< .001	[-.940, -.880]	.13	-4.740^{-06}	5.493	6.000^{-02}
			Stineman				
Reciprocity	2.517^{-02} (1.090^{-03})	< .001	[2.214^{-02}, 2.819^{-02}]	.80	1.844^{-03}	7.957	6.050^{-03}
In-degree sender	4.180^{-04} (1.009^{-05})	< .001	[3.900^{-04}, 4.461^{-04}]	.82	-1.342^{-05}	3.325	5.604^{-05}
Out-degree receiver	-9.324^{-05} (3.891^{-06})	< .001	[1.040^{-04}, -8.244^{-05}]	.90	-3.006^{-06}	4.524	2.161^{-05}
Same location	-.910 (1.173^{-02})	< .001	[-.943, -.877]	.17	-4.734^{-02}	5.486	6.516^{-02}

Note. Number of possible events = 931.680, number of events = 3882. 'Time' is imputed as a single value by interpolation. 'Sender' and 'Receiver' are imputed through multiple imputations in MICE. Standard errors in parentheses. The number of imputations is 5 in each simulation. The number of simulations is 100.

5. Discussion and Conclusions

This study aimed to explore the extent to which (multiple) imputation of missing values affects analyses of REH data. Missingness was simulated 100 times through the NDD mechanism in part of the Apollo 13 data and was allowed to occur in any combination of time, sender, and receiver information. In subsequent imputation models missing values in sender and receiver nodes in the simulated dataset were imputed via multiple imputation, while missing time values were interpolated by the prior and subsequent values. Next, the results of the REMs in the simulations were aggregated and compared to the fully observed analysis and complete case analysis based on the bias introduced in the statistics of reciprocity, in-degree sender, out-degree receiver and whether the dyad shared the same location.

In the fully observed analysis – the truth – the in-degree of the sender and same location were statistically significant predictors of the event rate as higher in-degree was associated with being a sender in the future and nodes were more likely to engage with receiver nodes who were in a different location. The results of the subsequent analyses indicated that missingness in relational event history (REH) data, even in a randomized fashion, biased the estimates. This conclusion is in line with studies conducted by Kossinets (2006) and Huisman (2009) which utilized static network data. In line with the literature on the NDD mechanism and complete case analysis (e.g., van Buuren, 2018), complete case analysis seemed moderately reliable in an NDD context with a substantial amount of data remaining. Although coefficients for reciprocity and in-degree sender were overestimated, while out-degree receiver and same location were underestimated, the statistical significance of the coefficients was not altered, which speaks in favor of complete case analysis.

In the analysis involving the imputed simulations, coefficients were less biased compared to complete case analysis in how far from the 'truth' the coefficients were estimated to be. However, smaller standard errors of the coefficients led to the reciprocity and out-degree receiver statistics incorrectly being identified as statistically significant predictors of the event rate. Based on the criteria of relative bias of the estimates and narrow average width of the confidence intervals these simulated imputations seemed acceptable, but the suboptimal coverage rates for the statistics warrant caution for making inferences from these imputations. Additional analyses where time was interpolated through the spline and Stineman algorithms (Stineman, 1980) yielded somewhat

different results in the statistics, but overall conclusions were alike. In sum, while the REM after imputation does estimate the effect size more accurately than complete case analysis, the former falsely detected statistically significant results in reciprocity and out-degree of the receiver whereas the latter did not.

The are several limitations in the current study that should be mentioned. First, missing times in the current study were interpolated from the timepoint before and after those missing values rather than through multiple imputation. This was done because the challenge of multiple imputations of time in REH data proved to be more complex than expected because it is essential to preserve the chronological order while adding sufficient noise to the multiple imputations to obtain reliable inferences. The noise in some imputations of missing times caused the order to shift in some cases resulting in a violation of one of REH's assumptions – that there is at least an order in the relational events (Butts, 2008). Despite this limitation of employing single imputation, the single imputation of missing times could perhaps be worthwhile to explore further too as indicated by the to some extent varying results in the REMs where interpolation of time differed (linear, spline, and the Stineman method). Because REH data is stored chronologically, in some contexts missing values in time may be imputed with relative certainty through a single imputation as the order may have been retained even though some of the exact times are missing. However, when the order of social interaction in REH data is missing too, multiple imputation should most likely be the baseline for imputation of time. Therefore, correct methods for multiple imputation of time in REH data should be explored further as the element of time is so pivotal.

Second, the content of the communications is overlooked in imputing missingness, but it is reasonable to assume the messages themselves could provide valuable information for more reliable imputation models. For instance, some communications may have a more positive sentiment whereas others have a more negative sentiment, and including such exogenous statistics could be an exciting direction for subsequent studies.

Third, the current study simulates missingness through the 'simpler' NDD mechanism as it serves as a baseline to explore the missingness problem in REH data and a logical step would be to perform similar analyses in an SDD (or UDD) context too.

Fourth, because samples were drawn from a finite population the variance and resulting standard errors of the statistics were substantially smaller in the imputed simulations than in the

fully observed and complete case analyses. However, had sampling variance been included in the simulations this would have led to overestimation of standard errors, and would have been incorrect as we had the finite population at our disposal. Therefore, an important task for future research lies in finding a balance between an appropriate amount of variance and conserving part of the data in each simulation.

Despite these limitations, the current study does indicate that multiple imputation of sender and receiver data and single imputation of time performs better than complete case analysis in an NDD context when looking at effect sizes. However, caution is required when making further inferences from these imputation analyses based on the extremely small standard errors – and resulting statistically significant *p*-values.

References

Apollo 13 Real-time. (n.d.). Retrieved from http://apollo13realtime.org/

Bates, F. L., & Harvey, C. C. (1975). *The Structure of Social Systems*. New York: Gardner Press.

Böhnke, P., & Link, S. (2017). Poverty and the dynamics of social networks: An analysis of German panel data. *European Sociological Review*, *33*(4), 615-632. https://doi.org/10.1093/esr/jcx063

Butts, C. T. (2008). A relational event framework for social action. *Sociological methodology*, *38*(1), 155-200. https://doi.org/10.1111/j.1467-9531.2008.00203.x

Gile, K. J., & Handcock, M. S. (2017). Analysis of networks with missing data with application to the National Longitudinal Study of Adolescent Health. *Journal of the Royal Statistical Society Series C: Applied Statistics*, *66*(3), 501-519. https://doi.org/10.1111/rssc.12184

Huisman, M. (2009). Imputation of missing network data: Some simple procedures. *Journal of Social Structure*, *10*(1), 1-29. https://doi.org/10.21307/joss-2019-050

Kamalabad, M. S., Leenders, R., & Mulder, J. (2023). What is the Point of Change? Change Point Detection in Relational Event Models. *Social Networks*, *74*, 166-181. https://doi.org/10.1016/j.socnet.2023.03.004

Kiang, M. V., Chen, J. T., Krieger, N., Buckee, C. O., Alexander, M. J., Baker, J. T., ... & Onnela, J. P. (2021). Sociodemographic characteristics of missing data in digital

phenotyping. *Scientific reports*, *11*(1), 15408. https://doi.org/10.1038/s41598-021-94516-7

McGloin, J. M., & Kirk, D. S. (2014). An overview of social network analysis. *Advancing Quantitative Methods in Criminology and Criminal Justice*, 67-79.

Meijerink-Bosman, M., Leenders, R., & Mulder, J. (2022). Dynamic relational event modeling: Testing, exploring, and applying. PLoS One, 17(8), e0272309. https://doi.org/10.1371/journal.pone.0272309

Meijerink-Bosman, M., Back, M., Geukes, K., Leenders, R., & Mulder, J. (2023). Discovering trends of social interaction behavior over time: An introduction to relational event modeling: Trends of social interaction. *Behavior Research Methods*, *55*(3), 997-1023. https://doi.org/10.3758/s13428-024-02423-2

Moritz S, Bartz-Beielstein T (2017). “imputeTS: Time Series Missing Value Imputation in R.” _The R Journal_, *9*(1), 207-218.

Newman, D. A. (2014). Missing data: Five practical guidelines. *Organizational Research Methods*, *17*(4), 372-411. https://doi.org/10.1177/1094428114548590

Newman, M. F. (2018). Networks. In Oxford University Press eBooks. https://doi.org/10.1093/oso/9780198805090.001.0001

Oberman, H. I., & Vink, G. (2023). Toward a standardized evaluation of imputation methodology. Biometrical Journal. https://doi.org/10.1002/bimj.202200107

Rubin, D. B. (1976). Inference and missing data. *Biometrika*, *63*(3), 581–590. https://doi.org/10.1093/biomet/63.3.581

Salgado CM, Azevedo C, Proença H, Vieira SM. *Missing Data. In: Secondary Analysis of Electronic Health Records.* Springer, Cham (CH); 2016. PMID: 31314252.

Schafer, J. L., & Graham, J. W. (2002). Missing data: our view of the state of the art. *Psychological methods*, *7*(2), 147. https://doi.org/10.1037/1082-989X.7.2.147

Stadtfeld, C., & Block, P. (2017). Interactions, actors, and time: Dynamic network actor models for relational events. *Sociological Science*, *4*, 318-352. https://doi.org/10.15195/v4.a14

Stineman, R. W. (1980). A consistently well-behaved method of interpolation. *Creative Computing*, *6*(7), 54-57.

Therneau, T. M. (2023, March 12). Survival Analysis [R package survival version 3.5-5]. https://cran.r-project.org/package=survival

van Buuren, S. (2018). *Flexible Imputation of Missing Data* (2nd ed.). Boca Raton: CRC Press.

van Buuren, S., & Groothuis-Oudshoorn, K. (2011). mice: Multivariate imputation by chained equations in R. *Journal of statistical software*, *45*, 1-67.

https://10.18637/jss.v045.i03

Vieira, F., Leenders, R., & Mulder, J. (2024). Fast meta-analytic approximations for relational event models: applications to data streams and multilevel data. *Journal of Computational Social Science*, 1-37. https://doi.org/10.1007/s42001-024-00290-7

Vink, G., & van Buuren, S. (2014). Pooling multiple imputations when the sample happens to be the population. *arXiv preprint arXiv:1409.8542*. https://doi.org/10.48550/arXiv.1409.8542

Vink G., (2023). mice::mice.impute.pmm.conditional() https://github.com/gerkovink/mice/tree/match_conditional_current

Appendix A: Apollo 13 Actors and IDs

- AFD: Assistant Flight Director from Flight directors (1)
- CAPCOM: Capsule Communicator from Flight directors (2)
- CONTROL: Control Officer from Flight directors (3)
- EECOM: Electrical, Environmental and Consumables Manager from Flight directors (4)
- All : Ground control team (without flight directores) (5)
- FDO : Flight dynamics officer (FDO or FIDO) (6)
- FLIGHT: Flight Director from Flight directors (7)
- GNC: The Guidance, Navigation, and Controls Systems Engineer from Flight directors (8)
- GUIDO: Guidance Officer from Flight directors (9)
- INCO: Integrated Communications Officer from Flight directors (10)
- NETWORK: Network of ground stations from Flight directors (11)
- TELMU: Telemetry, Electrical, and EVA Mobility Unit Officer from Flight directors (12)
- RECOVERY: Recovery Supervisor from Flight directorsc (13)
- PROCEDURES: Organization & Procedures Officer from Flight directors (14)
- FAO: Flight activities officer from Flight directors (15)
- RETRO: Retrofire Officer from Flight directors (16)
- CDR: Commander James A. Lovell Jr. crew (astronauts) (17)
- CMP: Command Module Pilot John (Jack) L. Swigert Jr. crew (astronauts) (18)
- LMP: Lunar module pilot Fred W. Haise Jr. crew (astronauts) (19)

Appendix B: Sensitivity Analysis

In this analysis same location was used in imputation models of sender and receiver as additional predictor. Conclusions are similar but note that coverage rates are substantially lower in this REM – except for reciprocity.

Table 7.

Aggregated REM results of 100 simulations after imputation of missingness with same location as additional predictor.

Statistic	*β*	*p*-value	*CI-95%* [LB, UB]	CR	Bias	PB	AW
Reciprocity	2.483^{-02} (1.001^{-03})	< .001	[2.205^{-02}, 2.761^{-02}]	.85	1.509^{-03}	6.702	5.560^{-03}
In-degree sender	4.072^{-04} (1.047^{-05})	< .001	[3.781^{-04}, 4.363^{-04}]	.66	-2.424^{-05}	5.612	5.815^{-05}
Out-degree receiver	-9.340^{-05} (3.888^{-06})	< .001	[-1.042^{-04}, -8.261^{-05}]	.92	-3.167^{-06}	4.890	2.159^{-05}
Same location	-.886 (8.795^{-03})	< .001	[-.911, -.862]	.51	-2.330^{-02}	2.700	4.884^{-02}

Note. Number of possible events = 931.680, number of events = 3882. 'Time' imputed as single value by interpolation. 'Sender' and 'Receiver' imputed through multiple imputation in MICE. Standard errors in parentheses. The number of imputations is 5 in each simulation. The number of simulations is 100.

Appendix C: GitHub Repository and R Syntax

Comerford D., Dominic_2024 (2024). GitHub repository,

https://github.com/mshafieek/ADS-Missing-data-social-network/tree/main/Dominic_2024

```
##Loading Packages and Data
##### Load packages
packages_to_install <- c("purrr", "furrr", "magrittr", "dplyr",
                         "tibble", "survival", "tidyverse",
                         "devtools", "igraph", "sna", "imputeTS", "stats")

for (pkg in packages_to_install) {
  if (!require(pkg, character.only = TRUE)) {
    # If not, install the package
    install.packages(pkg)
  }
}

library(purrr, warn.conflicts = FALSE)      # for functional programming
library(furrr, warn.conflicts = FALSE)      # for functional futures
library(magrittr, warn.conflicts = FALSE)   # for pipes
library(dplyr, warn.conflicts = FALSE)      # for data manipulation
library(tibble, warn.conflicts = FALSE)     # for tibbles
library(survival, warn.conflicts = FALSE)   # for REM analysis
library(tidyverse, warn.conflicts = FALSE)  # tidyverse
library(tidyr, warn.conflicts = FALSE)      # tidyr
library(igraph, warn.conflicts = FALSE)     # network graphs
library(sna, warn.conflicts = FALSE)        # social network analysis
library(imputeTS, warn.conflicts = FALSE)   # time interpolation
library(stats, warn.conflicts = FALSE)      # for a range of model stats

devtools::install_github("TilburgNetworkGroup/remify")
devtools::install_github("gerkovink/mice@match_conditional_current")
devtools::install_github("TilburgNetworkGroup/remstats")

library(mice, warn.conflicts = FALSE)      # for imputation and amputation
library(remstats, warn.conflicts = FALSE)  # for REM statistics
library(remify, warn.conflicts = FALSE)    # for converting

##### Load data
con <- url("https://github.com/mshafieek/ADS-Missing-data-social-
network/raw/main/literature_%20REM/Tutorial_REM_REH_DATA/UUsummerschool.Rdata
")
load(con)
apollo <- PartOfApollo_13 %>%
  rename(
    actor1 = sender,
    actor2 = receiver
  )
rm(Class, PartOfApollo_13, Twitter_data_rem3, WTCPoliceCalls, ClassIntercept,
   ClassIsFemale, ClassIsTeacher, WTCPoliceIsICR, con, pkg)

## Same location dummy
apollo$s_ast <- ifelse(apollo$actor1 > 16, 1, 0)
```

```
apollo$r_ast <- ifelse(apollo$actor2 > 16, 1, 0)
apollo$same_location <- ifelse(apollo$s_ast == apollo$r_ast, 1, 0)
apollo$s_ast <- NULL
apollo$r_ast <- NULL

head(apollo)
tail(apollo)
summary(apollo)
str(apollo)

#Descriptive network analysis
edges_apollo <-
data.frame(from=c(as.character(apollo[,2])),to=c(as.character(apollo[,3])))
graph_apollo <- graph_from_data_frame(edges_apollo,directed = TRUE)

net <- simplify(graph_apollo, remove.multiple = T) # remove multiple edges
for snapshot network analysis
edge_density(net)

centr_clo(net, mode="all", normalized=T)$centralization
centr_betw(net, directed=T, normalized=T)$centralization
centr_eigen(net, directed=T, normalized=T)$centralization
transitivity(net, type="global")

diameter(net, directed=T)
mean_distance(net, directed=T)
sum(apollo$same_location)

set.seed(0)
ApolloNet <- as.sociomatrix.eventlist(apollo[1:3], 19)
Figure_1 <- gplot(ApolloNet, jitter = TRUE, pad = .075,
              mode = "target",
              displaylabels = TRUE, label.pos = 0, label.cex = .75,
              boxed.labels = TRUE, label.pad = .5,
              displayisolates = FALSE, vertex.cex=.6,
              arrowhead.cex = .75, edge.lwd = -.75, edge.col = "gray",
              vertex.col = ifelse(seq_along(ApolloNet) %in% c(17, 18, 19),
"blue", "red")) # astronauts as blue

##Sufficient set & Missingness function

set.seed(123) # fix seed to realize a sufficient set
apollo <- apollo |> as_tibble()
indic <- sample(1:nrow(apollo), 1500)
remify(apollo[indic, ], model = "tie") %>% dim() # check if 16 nodes
#### Combine the sufficient set and the incomplete set
make_missing <- function(x, indic) {
  x$time_index <- seq_len(nrow(x))
  sufficient <- x[indic, ]
  miss <- x[-c(indic), ] |>
    ampute(prop = 0.4,
           mech = "MCAR", # (4th column is same location, 5th column is
time_index)
           patterns = matrix(c(1,1,0,1,1, # missing in actor 2
                               1,0,1,1,1, # missing in actor 1
                               0,1,1,1,1, # missing in time
                               0,0,0,1,1, # missing in all
```

```
                                  1,0,0,1,1, # missing in actor 1 + 2
                                  0,1,0,1,1, # missing in time + actor 2
                                  0,0,1,1,1  # missing in time + actor 1
                                  ), nrow=7,
                                byrow=TRUE)) %>%
    .$amp
  combined <- rbind(sufficient, miss)
  combined <- combined[order(combined$time_index), ]
  combined <- combined[-5] # remove time_index
  return(combined)
}

##Interpolate 'time' before MICE
set.seed(123)
mbased_finite_apollo_miss <-
  furrr::future_map(1:100, ~ {   # Create 100 simulated datasets with
missingness
    make_missing(apollo, indic) }, .options = furrr_options(seed = 123))
mbased_finite_apollo_miss_cc <- mbased_finite_apollo_miss # for later
complete case analysis before time gets imputed

Figure_2 <- md.pattern(mbased_finite_apollo_miss[[1]], rotate.names = TRUE)

##Impute time with interpolation (single imputation)
for (i in 1:length(mbased_finite_apollo_miss)) {
  # Impute missing values in the 'time' column using na_interpolation
  mbased_finite_apollo_miss[[i]]$time <-
na_interpolation(mbased_finite_apollo_miss[[i]]$time)
}
#### When using the spline and stineman methods for interpolation of time:
# for (i in 1:length(mbased_finite_apollo_miss)) {
#   # Impute missing values in the 'time' column using na_interpolation
#   mbased_finite_apollo_miss[[i]]$time <-
na_interpolation(mbased_finite_apollo_miss[[i]]$time, option = "spline")
# }
# for (i in 1:length(mbased_finite_apollo_miss)) {
#   # Impute missing values in the 'time' column using na_interpolation
#   mbased_finite_apollo_miss[[i]]$time <-
na_interpolation(mbased_finite_apollo_miss[[i]]$time, option = "stine")
# }

##Impute sender and receiver through MICE

##Multiple imputation specification
whichcol <- c("", "actor2", "actor1", "") # Ensure that actor 1 != actor 2 in
imputations
names(whichcol) <- colnames(apollo)
## predictor matrix
pred <- make.predictorMatrix(apollo)
pred[c("actor1", "actor2"), "same_location"] <- 0 # exclude same location as
predictor of actor 1 + 2. Comment out for sensitivity analyses (Table 7)

## use the pmm.conditional method
method <- make.method(apollo)
method[c(2,3)] <- "pmm.conditional"
mbased_finite_apollo <-
  furrr::future_map(1:100, ~ {
```

```
  mice(mbased_finite_apollo_miss[[.x]],
       m = 5,
       maxit = 5,
       method = method,
       whichcolumn = whichcol,
       predictorMatrix = pred,
       print = FALSE)
}, .options = furrr_options(seed = 123))

##Multiple imputation check
# ##### Missing data pattern of all simulations.
convergence <- plot(mbased_finite_apollo[[5]])
convergence
stripplot <- stripplot(mbased_finite_apollo[[5]])
stripplot

##Defining REM effects and preparing data for Cox function
##### Defining effects for REM
effects <- ~ -1 + reciprocity(scaling = ("std")) + indegreeSender() +
outdegreeReceiver()

##### Function to get the statistics of the previously defined effects.
stats_function <- function(data) {
  # remify the data
  reh <- remify::remify(edgelist = data, model = "tie")
  # calculate effect statistics
  statsObject_imp <- remstats(reh = reh, tie_effects = effects)
  # Return the statistics
  return(statsObject_imp)
}

##### Function for making the data compatible with coxph()
prepare_coxph_data <- function(statsObject, apollo) {
  risk_sets <- attr(statsObject, "riskset")
  risk_sets <- risk_sets %>% select(-'id')
  # Get the times
  time <- apollo$time
  # merge riskset with each timepoint
  combined <- merge(risk_sets, time, by = NULL)

  combined <- combined %>% rename("time" = "y")
  combined <- lapply(combined, as.numeric)
  combined <- as.data.frame(combined)

  # Create matrices for subtraction to make a status column for coxph
  combined_matrix <- data.matrix(combined)
  matrix_rows <- nrow(combined)

  repeated_df <- apollo[rep(seq_len(nrow(apollo)), each = 240), ]
  repeated_df <- repeated_df[, c(2,3,1)]
  apollo_matrix <- data.matrix(repeated_df)

  status_matrix <- apollo_matrix - combined_matrix

  # create a status column
  status <- as.integer(rowSums(status_matrix == 0) == ncol(status_matrix))
  status <- as.data.frame(status)
```

```
  # Add status to the combined set
  combined <- cbind(combined, status)

  # Extract statistics and add them to the dataframe
  reciprocity <- statsObject[,,1]
  indegreeSender <- statsObject[,,2]
  outdegreeReceiver <- statsObject[,,3]

  combined$reciprocity <- c(reciprocity)
  combined$indegreeSender <- c(indegreeSender)
  combined$outdegreeReceiver <- c(outdegreeReceiver)

  ## add same location
  combined$s_ast <- ifelse(combined$sender > 16, 1, 0)
  combined$r_ast <- ifelse(combined$receiver > 16, 1, 0)
  combined$same_location <- ifelse(combined$s_ast == combined$r_ast, 1, 0)
  combined$s_ast <- NULL
  combined$r_ast <- NULL
  return(combined)
}

##Fully observed data
###### TRUE ANALYSIS
true.reh <- remify(edgelist = apollo,
                   model = "tie")
# calculate stats
stats <- remstats(tie_effects = effects,
                  reh = true.reh)
# use the function to create the correct format of the dataframe
true.cox.set <- prepare_coxph_data(stats, apollo)

# fit cox model
true.cox.fit <- coxph(Surv(time, status) ~ reciprocity + indegreeSender +
                        outdegreeReceiver + same_location,
                      data=true.cox.set)
true <- coefficients(true.cox.fit) # save the true values
true.cox.fit
BIC(true.cox.fit)
##Analysis on time (interpolated), actor 1 + 2 Multiple imputed
###### Running the REM on  simulations
Results1 <-
  mbased_finite_apollo[1:10] %>%
  map(~.x %>% # for every simulation
        complete("all") %>%
        map(~.x %>% # for every imputation
              stats_function() %>% # do stats function
              prepare_coxph_data(apollo = .x) %$% # prepare cox ph
              coxph(Surv(time, status) ~
                      reciprocity +
                      indegreeSender +
                      outdegreeReceiver +
                      same_location)) %>%
        pool(custom.t = ".data$b + .data$b / .data$m") %>% # pool
coefficients
        .$pooled %>% # extract table of pooled coefficients
          mutate(true = true, # add true
```

```
                df = m-1,  # correct df
                riv = Inf, # correct riv
                std.error = sqrt(t), # standard error
                statistic = estimate / std.error, # test statistic
                p.value = 2 * (pt(abs(statistic),
                                  pmax(df, 0.001),
                                  lower.tail = FALSE)), # correct p.value
                `2.5 %` = estimate - qt(.975, df) * std.error, # lower bound
CI
                `97.5 %` = estimate + qt(.975, df) * std.error, # upper
bound CI
                cov = `2.5 %` < true & true < `97.5 %`, # coverage
                bias = estimate - true) %>% # bias
          select(term, m, true, estimate, std.error, statistic, p.value,
                 riv, `2.5 %`, `97.5 %`, cov, bias) %>%
          column_to_rownames("term") # `term` as rownames
    )

Results2 <-
  mbased_finite_apollo[11:20] %>%
  map(~.x %>% # for every simulation
        complete("all") %>%
        map(~.x %>% # for every imputation
              stats_function() %>% # do stats function
              prepare_coxph_data(apollo = .x) %$% # prepare cox ph
              coxph(Surv(time, status) ~
                      reciprocity +
                      indegreeSender +
                      outdegreeReceiver+
                      same_location)) %>%
        pool(custom.t = ".data$b + .data$b / .data$m") %>% # pool
coefficients
        .$pooled %>% # extract table of pooled coefficients
          mutate(true = true, # add true
                 df = m-1,  # correct df
                 riv = Inf, # correct riv
                 std.error = sqrt(t), # standard error
                 statistic = estimate / std.error, # test statistic
                 p.value = 2 * (pt(abs(statistic),
                                   pmax(df, 0.001),
                                   lower.tail = FALSE)), # correct p.value
                 `2.5 %` = estimate - qt(.975, df) * std.error, # lower bound
CI
                 `97.5 %` = estimate + qt(.975, df) * std.error, # upper
bound CI
                 cov = `2.5 %` < true & true < `97.5 %`, # coverage
                 bias = estimate - true) %>% # bias
          select(term, m, true, estimate, std.error, statistic, p.value,
                 riv, `2.5 %`, `97.5 %`, cov, bias) %>%
          column_to_rownames("term") # `term` as rownames
    )

Results3 <-
  mbased_finite_apollo[21:30] %>%
  map(~.x %>% # for every simulation
        complete("all") %>%
        map(~.x %>% # for every imputation
```

```
              stats_function() %>% # do stats function
              prepare_coxph_data(apollo = .x) %$% # prepare cox ph
              coxph(Surv(time, status) ~
                      reciprocity +
                      indegreeSender +
                      outdegreeReceiver+
                      same_location)) %>%
        pool(custom.t = ".data$b + .data$b / .data$m") %>% # pool
coefficients
        .$pooled %>% # extract table of pooled coefficients
          mutate(true = true, # add true
                 df = m-1,  # correct df
                 riv = Inf, # correct riv
                 std.error = sqrt(t), # standard error
                 statistic = estimate / std.error, # test statistic
                 p.value = 2 * (pt(abs(statistic),
                                   pmax(df, 0.001),
                                   lower.tail = FALSE)), # correct p.value
                 `2.5 %` = estimate - qt(.975, df) * std.error, # lower bound
CI
                 `97.5 %` = estimate + qt(.975, df) * std.error, # upper
bound CI
                 cov = `2.5 %` < true & true < `97.5 %`, # coverage
                 bias = estimate - true) %>% # bias
          select(term, m, true, estimate, std.error, statistic, p.value,
                 riv, `2.5 %`, `97.5 %`, cov, bias) %>%
          column_to_rownames("term") # `term` as rownames
  )

Results4 <-
  mbased_finite_apollo[31:40] %>%
  map(~.x %>% # for every simulation
        complete("all") %>%
        map(~.x %>% # for every imputation
              stats_function() %>% # do stats function
              prepare_coxph_data(apollo = .x) %$% # prepare cox ph
              coxph(Surv(time, status) ~
                      reciprocity +
                      indegreeSender +
                      outdegreeReceiver+
                      same_location)) %>%
        pool(custom.t = ".data$b + .data$b / .data$m") %>% # pool
coefficients
        .$pooled %>% # extract table of pooled coefficients
          mutate(true = true, # add true
                 df = m-1,  # correct df
                 riv = Inf, # correct riv
                 std.error = sqrt(t), # standard error
                 statistic = estimate / std.error, # test statistic
                 p.value = 2 * (pt(abs(statistic),
                                   pmax(df, 0.001),
                                   lower.tail = FALSE)), # correct p.value
                 `2.5 %` = estimate - qt(.975, df) * std.error, # lower bound
CI
                 `97.5 %` = estimate + qt(.975, df) * std.error, # upper
bound CI
                 cov = `2.5 %` < true & true < `97.5 %`, # coverage
```

```
                bias = estimate - true) %>% # bias
          select(term, m, true, estimate, std.error, statistic, p.value,
                 riv, `2.5 %`, `97.5 %`, cov, bias) %>%
          column_to_rownames("term") # `term` as rownames
  )

Results5 <-
  mbased_finite_apollo[41:50] %>%
  map(~.x %>% # for every simulation
        complete("all") %>%
        map(~.x %>% # for every imputation
              stats_function() %>% # do stats function
              prepare_coxph_data(apollo = .x) %$% # prepare cox ph
              coxph(Surv(time, status) ~
                      reciprocity +
                      indegreeSender +
                      outdegreeReceiver+
                      same_location)) %>%
        pool(custom.t = ".data$b + .data$b / .data$m") %>% # pool
coefficients
        .$pooled %>% # extract table of pooled coefficients
          mutate(true = true, # add true
                 df = m-1,  # correct df
                 riv = Inf, # correct riv
                 std.error = sqrt(t), # standard error
                 statistic = estimate / std.error, # test statistic
                 p.value = 2 * (pt(abs(statistic),
                                   pmax(df, 0.001),
                                   lower.tail = FALSE)), # correct p.value
                 `2.5 %` = estimate - qt(.975, df) * std.error, # lower bound
CI
                 `97.5 %` = estimate + qt(.975, df) * std.error, # upper
bound CI
                 cov = `2.5 %` < true & true < `97.5 %`, # coverage
                 bias = estimate - true) %>% # bias
          select(term, m, true, estimate, std.error, statistic, p.value,
                 riv, `2.5 %`, `97.5 %`, cov, bias) %>%
          column_to_rownames("term") # `term` as rownames
  )

Results6 <-
  mbased_finite_apollo[51:60] %>%
  map(~.x %>% # for every simulation
        complete("all") %>%
        map(~.x %>% # for every imputation
              stats_function() %>% # do stats function
              prepare_coxph_data(apollo = .x) %$% # prepare cox ph
              coxph(Surv(time, status) ~
                      reciprocity +
                      indegreeSender +
                      outdegreeReceiver+
                      same_location)) %>%
        pool(custom.t = ".data$b + .data$b / .data$m") %>% # pool
coefficients
        .$pooled %>% # extract table of pooled coefficients
          mutate(true = true, # add true
                 df = m-1,  # correct df
```

```
                riv = Inf, # correct riv
                std.error = sqrt(t), # standard error
                statistic = estimate / std.error, # test statistic
                p.value = 2 * (pt(abs(statistic),
                                  pmax(df, 0.001),
                                  lower.tail = FALSE)), # correct p.value
                `2.5 %` = estimate - qt(.975, df) * std.error, # lower bound
CI
                `97.5 %` = estimate + qt(.975, df) * std.error, # upper
bound CI
                cov = `2.5 %` < true & true < `97.5 %`, # coverage
                bias = estimate - true) %>% # bias
          select(term, m, true, estimate, std.error, statistic, p.value,
                 riv, `2.5 %`, `97.5 %`, cov, bias) %>%
          column_to_rownames("term") # `term` as rownames
  )

Results7 <-
  mbased_finite_apollo[61:70] %>%
  map(~.x %>% # for every simulation
        complete("all") %>%
        map(~.x %>% # for every imputation
              stats_function() %>% # do stats function
              prepare_coxph_data(apollo = .x) %$% # prepare cox ph
              coxph(Surv(time, status) ~
                      reciprocity +
                      indegreeSender +
                      outdegreeReceiver+
                      same_location)) %>%
        pool(custom.t = ".data$b + .data$b / .data$m") %>% # pool
coefficients
        .$pooled %>% # extract table of pooled coefficients
          mutate(true = true, # add true
                 df = m-1,  # correct df
                 riv = Inf, # correct riv
                 std.error = sqrt(t), # standard error
                 statistic = estimate / std.error, # test statistic
                 p.value = 2 * (pt(abs(statistic),
                                   pmax(df, 0.001),
                                   lower.tail = FALSE)), # correct p.value
                 `2.5 %` = estimate - qt(.975, df) * std.error, # lower bound
CI
                 `97.5 %` = estimate + qt(.975, df) * std.error, # upper
bound CI
                 cov = `2.5 %` < true & true < `97.5 %`, # coverage
                 bias = estimate - true) %>% # bias
          select(term, m, true, estimate, std.error, statistic, p.value,
                 riv, `2.5 %`, `97.5 %`, cov, bias) %>%
          column_to_rownames("term") # `term` as rownames
  )

Results8 <-
  mbased_finite_apollo[71:80] %>%
  map(~.x %>% # for every simulation
        complete("all") %>%
        map(~.x %>% # for every imputation
              stats_function() %>% # do stats function
```

```
              prepare_coxph_data(apollo = .x) %$% # prepare cox ph
              coxph(Surv(time, status) ~
                      reciprocity +
                      indegreeSender +
                      outdegreeReceiver+
                      same_location)) %>%
        pool(custom.t = ".data$b + .data$b / .data$m") %>% # pool
coefficients
        .$pooled %>% # extract table of pooled coefficients
          mutate(true = true, # add true
                 df = m-1,  # correct df
                 riv = Inf, # correct riv
                 std.error = sqrt(t), # standard error
                 statistic = estimate / std.error, # test statistic
                 p.value = 2 * (pt(abs(statistic),
                                   pmax(df, 0.001),
                                   lower.tail = FALSE)), # correct p.value
                 `2.5 %` = estimate - qt(.975, df) * std.error, # lower bound
CI
                 `97.5 %` = estimate + qt(.975, df) * std.error, # upper
bound CI
                 cov = `2.5 %` < true & true < `97.5 %`, # coverage
                 bias = estimate - true) %>% # bias
          select(term, m, true, estimate, std.error, statistic, p.value,
                 riv, `2.5 %`, `97.5 %`, cov, bias) %>%
          column_to_rownames("term") # `term` as rownames
  )

Results9 <-
  mbased_finite_apollo[81:90] %>%
  map(~.x %>% # for every simulation
        complete("all") %>%
        map(~.x %>% # for every imputation
              stats_function() %>% # do stats function
              prepare_coxph_data(apollo = .x) %$% # prepare cox ph
              coxph(Surv(time, status) ~
                      reciprocity +
                      indegreeSender +
                      outdegreeReceiver+
                      same_location)) %>%
        pool(custom.t = ".data$b + .data$b / .data$m") %>% # pool
coefficients
        .$pooled %>% # extract table of pooled coefficients
          mutate(true = true, # add true
                 df = m-1,  # correct df
                 riv = Inf, # correct riv
                 std.error = sqrt(t), # standard error
                 statistic = estimate / std.error, # test statistic
                 p.value = 2 * (pt(abs(statistic),
                                   pmax(df, 0.001),
                                   lower.tail = FALSE)), # correct p.value
                 `2.5 %` = estimate - qt(.975, df) * std.error, # lower bound
CI
                 `97.5 %` = estimate + qt(.975, df) * std.error, # upper
bound CI
                 cov = `2.5 %` < true & true < `97.5 %`, # coverage
                 bias = estimate - true) %>% # bias
```

```
          select(term, m, true, estimate, std.error, statistic, p.value,
                 riv, `2.5 %`, `97.5 %`, cov, bias) %>%
          column_to_rownames("term") # `term` as rownames
  )

Results10 <-
  mbased_finite_apollo[91:100] %>%
  map(~.x %>% # for every simulation
        complete("all") %>%
        map(~.x %>% # for every imputation
              stats_function() %>% # do stats function
              prepare_coxph_data(apollo = .x) %$% # prepare cox ph
              coxph(Surv(time, status) ~
                      reciprocity +
                      indegreeSender +
                      outdegreeReceiver+
                      same_location)) %>%
        pool(custom.t = ".data$b + .data$b / .data$m") %>% # pool
coefficients
        .$pooled %>% # extract table of pooled coefficients
          mutate(true = true, # add true
                 df = m-1,  # correct df
                 riv = Inf, # correct riv
                 std.error = sqrt(t), # standard error
                 statistic = estimate / std.error, # test statistic
                 p.value = 2 * (pt(abs(statistic),
                                   pmax(df, 0.001),
                                   lower.tail = FALSE)), # correct p.value
                 `2.5 %` = estimate - qt(.975, df) * std.error, # lower bound
CI
                 `97.5 %` = estimate + qt(.975, df) * std.error, # upper
bound CI
                 cov = `2.5 %` < true & true < `97.5 %`, # coverage
                 bias = estimate - true) %>% # bias
          select(term, m, true, estimate, std.error, statistic, p.value,
                 riv, `2.5 %`, `97.5 %`, cov, bias) %>%
          column_to_rownames("term") # `term` as rownames
  )

##Combining Results
##### Combining Results
Results <- list(Results1,
                Results2,
                Results3,
                Results4,
                Results5,
                Results6,
                Results7,
                Results8,
                Results9,
                Results10) %>%
  purrr::flatten()

##Save results
save(Results, file = "TimeINT_A12_MI.RData")
##Percentage bias and average width
load("TimeINT_A12_MI.RData")
```

```
##### Adding percentage bias and average width to the Results
#function for average width and percentage bias.
AW <- function(df) df[["97.5 %"]] - df[["2.5 %"]]
PB <- function(df) 100 * abs((df[["estimate"]] - df[["true"]]) /
df[["true"]])

Results_with_extra <- lapply(Results, function(df) {
  df$PB <- PB(df)
  df$AW <- AW(df)
  df
})
###### Average sims
Reduce("+", Results_with_extra) / length(mbased_finite_apollo)

##Complete Case Analysis
cox_sets_for_incomplete <- function(data) {
 statsObject <- remify::remify(edgelist = data, model = "tie") %>%
 remstats(tie_effects = effects) # create statistics for every amputed
dataset

 # make sure that complete apollo data to compare with is the same size as
 # amputed dataset with only complete cases
 complete.cases <- data[complete.cases(data), ]
 index <- as.numeric(rownames(complete.cases))
 apollo.missing <- apollo[index, ]

 # take the single riskset

 # remove the id column
 risk_sets <- attr(statsObject, "riskset") %>% select(-'id')
 # creating one set with all risksets for each time point
 combined <- merge(risk_sets, apollo.missing$time, by=NULL) %>%
 rename(time = y) %>%
 .[, c("time", "sender", "receiver")] %>%
 mutate(sender = as.numeric(sender),
 receiver = as.numeric(receiver))

 # GV: Calculate divergence
 diff <- apollo.missing[rep(seq_len(nrow(apollo.missing)), each = 240), ] %>%
 data.matrix() %>%
 .[, 1:3] - combined
 # GV: identify non-divergence
 combined$status <-
 rowSums(diff == 0) == ncol(diff)

 #combining the dataset with riskset to the statistic
 combined$reciprocity <- c(statsObject[,,1])
 combined$indegreeSender <- c(statsObject[,,2])
 combined$outdegreeReceiver <- c(statsObject[,,3])

 combined$status <- as.integer(as.logical(combined$status))

  ## add same location
 combined$s_ast <- ifelse(combined$sender > 16, 1, 0)
 combined$r_ast <- ifelse(combined$receiver > 16, 1, 0)
 combined$same_location <- ifelse(combined$s_ast == combined$r_ast, 1, 0)
 combined$s_ast <- NULL
```

```
 combined$r_ast <- NULL

 return(combined)
}

set.seed(123)
# cox model on complete cases
complete.case.fit <- mbased_finite_apollo_miss_cc %>%
 map(~.x %>% # for every completed data set....
 cox_sets_for_incomplete() %$%
 coxph(Surv(time, status) ~
 reciprocity +
 indegreeSender +
 outdegreeReceiver +
 same_location))

# create a dataframe out of the cox model objects
results <- list()
# Generate 100 dataframes
for (i in 1:100) {
 # Create a dataframe with columns of results from the cox models
 df <- data.frame(
 coef = complete.case.fit[[i]]$coefficients,
 se = coef(summary(complete.case.fit[[i]]))[, "se(coef)"],
 p = coef(summary(complete.case.fit[[i]]))[, "Pr(>|z|)"],
 true = true[1:4]
 )
 rownames(df) <- c("reciprocity", "indegreeSender", "outdegreeReceiver",
"same_location")
 # Append the dataframe to the list
 results[[i]] <- df
}

bic_complete_case <- numeric(length(complete.case.fit))
# Iterate over the list and extract BIC values
for (i in seq_along(complete.case.fit)) {
  bic_complete_case[i] <- BIC(complete.case.fit[[i]])
}
# Compute the average BIC
average_bic_complete_case <- mean(bic_complete_case)
average_bic_complete_case

# average the results across all simulations
average <- results %>%
 map(~.x %>%
 mutate(bias = coef - true) %>% # bias
 select(true, coef, se, p,
 bias)) %>%
 Reduce("+", .) / length(mbased_finite_apollo_miss_cc)
average
# Calculate risk set size and number of relational events in CC-analysis
nevent_values <- map_dbl(complete.case.fit, "nevent")
min(nevent_values)
max(nevent_values)
n_values <- map_dbl(complete.case.fit, "n")
min(n_values)
max(n_values)
```