Imputing Missing values in Relational Event History data: A Framework for Social Network Research

Vira Dvoriak (7334966)
Mahdi Shafiee Kamalabad
Gerko Vink
Master Applied Data Science, Utrecht University
July 7, 2023

Abstract

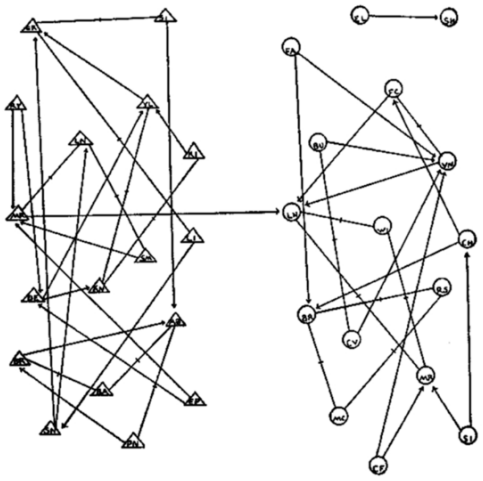
Background – Missing values are practically inevitable when it comes to data analysis and can cause loss of valuable information and introduce bias in the estimates. In social network data even a small proportion of missing values can have a substantial impact on the validity of results. The most common approach to deal with missing values is complete case analysis which does not always prevent these issues. Still, the research investigating this problem is scarce. This paper aims to address this gap by providing an overview of multiple imputation as a method of handling missing values in social network data.
Methods – Relational event model analysis is performed on the fully observed relational event history dataset to produce the true estimates. Next, a simulation study is performed to introduce missingness to a fully observed dataset. Then the results of two approaches - multiple imputation and complete case analysis are compared to the results of the analysis on the fully observed dataset.
Results –Multiple imputation with relational event model produced estimates with close to zero bias, high coverage rate, and low average width. However, multiple imputation produced false significant p-values. In addition, the distributions of the effect estimates were slightly skewed for all effects. Complete case analysis produced overestimated effects and standard errors but did not produce false significant p-values.
Conclusion – This study made first steps in evaluating whether multiple imputation with relational event model is a valid method to address missing values in relational event history data and compared it to the more common solution – complete case analysis. The findings reveal potential benefits of multiple imputation and propose direction for future research.

1. Introduction

1.1 Networks

Newman (2018) defines a network as a collection of points (nodes) connected to each other with lines (edges). Networks are used to study various systems in social, physical, and biological sciences. A network can be used to represent a system of interest, for example, an online or offline social network, biological neural network, etc. (Newman, 2018). An example of a real network can be seen in figure 1.

Figure 1. Example of a real social network. The network shows friendship patterns between school children in a class of 1930. The nodes are coded according to gender and lines represent friendship connections between children.



Note. From “Networks”, by M. E. J. Newman, 2018. Oxford University Press, p. 48

A social network is a network in which nodes represent individuals and edges represent various types of connections or relationships between them (Newmann, 2018). Social structures represented through social networks can allow for accurate reflection of relationships and other various social phenomena (Butts & Marcum, 2017). The network shown in figure 1 is an example of such utility of networks. This social network revealed an interesting relationship pattern among school children - there are many friendship connections within genders, but only one between genders (Newman, 2018). The significance of social networks in social research can be attributed to their simplicity and efficiency. One of the important qualities of social networks is that the edges can represent a variety of

relationships beginning with acquaintance and including professional relationships, close friendships, and romantic connections. Therefore, they allow the investigation of many types of social phenomena. In addition, social networks make it easy to convey conclusions of studies through a visual representation of the social phenomena (Newmann, 2018; Serrat, 2017).

Social network analysis can be employed to investigate questions such as: “what are communication patterns in local communities?” (Bernard, 1989); “How do behaviours and ideas spread through a social network?” (Fattore et al., 2009); “How do scientists collaborate?” (Grossman, 2002); “What are social structures of dating among young people?” (Bearman, Moody, Stovel, 2004); “What is the dynamic nature of interaction between teachers and students in the classes during the semester?” (Grunspan, Wiggins, & Goodreau, 2014). As well as more focused questions, such as “which aspects of past interactions determine future interactions?” (Meijerink-Bosman, Back, & Geukes, 2023). Social network analysis (SNA) is important for several reasons. Understanding relationships: SNA helps uncover and analyse the relationships and connections between individuals, groups, or entities (Hawe, & Ghali, 2008). It provides insights into how relationships develop, progress, change and how they can be utilised to achieve different outcomes (Hawe, & Ghali, 2008; Boyer et al., 2010; Newmann 2018). Identifying key players: SNA enables the identification of influential individuals or entities within a network (Serrat, 2017; Landherr et al., 2010). SNA can identify the most important nodes through measuring centrality of the network which can point to the nodes with most connections, or the nodes that lay on the path to most connections or the nodes that are connected to the highest number of other well-connected nodes (Newmann, 2018). Revealing structural patterns: SNA reveals the underlying structural patterns of a network, such as clusters, subgroups, or communities. SNA can identify nodes that are tightly connected between each other and categorise it as a community (Newmann, 2018). All these and many more capabilities of SNA are widely utilized due to the increasing availability of social network data.

1.2 Relational Event History Data

Social network data, at a minimum, consists of rows and columns which represent individuals or other social entities and a presence or absence of a relationship between them (Scott, &

Carrington, 2011). There are many types of social network data: interviews, archival records, social media data, phone records, etc., (Newmann, 2018). One of such types of social network data is relational event history (REH) data. In this type of data, relational events are analogous to edges in social network data. Butts and Marcum (2017) define REH as a sequence of discrete instances of interaction between a set of individuals or entities. This data consists of at least a time or order of the communication event, called relational event, and the actors that are involved in the event (Meijerink-Bosman, Back, & Geukes, 2023; Butts, 2008). An example of the REH data can be seen in table 1. The "Time" column shows the time that the communication event took place, which is continuous. The "Sender" column shows who initiated the communication event and the receiver column identifies at whom the communication event is directed. Names of the actors were coded as numbers.

Table 1. First four relational events in the Apollo 13 data.

Time	Sender	Receiver
11849.2	18	2
11854.2	2	18
11885.2	18	2
11890.2	2	18
…	…	…

REH data has become widely available with the development of technology. REH can be found in the form of online communication (email, Twitter, Facebook etc.), phone records, online collaboration, etc. This data makes it possible to study questions like "what behaviour drives communication?", "how can the next interaction event be predicted?", "what changes the dynamic of interaction over time?", "which actors' attributes influence communication dynamic?", "which patterns of interaction are common?" (Meijerink-Bosman, Back, & Geukes, 2023; Butts, 2008; Pilny et al., 2016). REH data is a widely regarded and a highly prevalent and influential type of network. One key characteristic of this type of data is the inherent dependency between the nodes and edges (Meijerink-Bosman, Back, & Geukes, 2023). Hence, it seems that traditional statistical methods do not handle this type of data very

well. Therefore, this data requires a unique tool to analyse it. The model that can take into account such a structure is relational event model (REM) (Butts, 2008). REM is especially suited to analyse continuous, fine grained, social interaction data, like REH data (Meijerink-Bosman, Back, & Geukes, 2023).

Missing values in REH data is a common occurrence. Just like in other types of social network data, it can cause a number of problems for the statistical analysis if not handled properly. Social network data is especially vulnerable to the problems of missing data (Burt, 1987). Missingness even in just one part of the node causes "missing parts of a single realization of a dependent process" (Gile, & Handcock, 2006, p. 2). Models that are used to analyse network data assume data completeness. Therefore, missing data can result in misleading or incorrect conclusions (Wang et al., 2016). In addition, Kossinets (2006) found that missing data in networks can affect network-level statistics, such as clustering and assortativity coefficients.

Based on whether the missingness is data dependent or not, different approaches to handle it must be applied (van Buuren, 2018). Regardless of the missingness mechanism, the default method that is used to handle missing values in R packages that are used to perform REM analysis like "survival" (Therneau, 2023), "remify" (Arena, 2023), and "remstats" (Meijerink-Bosman et al, 2023) is complete case analysis (removing observations with missing values). The disadvantages of complete case analysis are the loss of information and potential of biased results.

Several other approaches were proposed to handle missing data in social networks. Imputation methods proposed by Burt (1987) whereby missing values are replaced with values that represent weak relations and a reconstruction method proposed by Stork and Richards (1992). Likelihood-based estimation methods were also discussed by Robins et al. (2004), Gile and Handcock (2006), Handcock and Gile (2007) and Koskinen (2007) (Huisman & Steglich, 2008). One of the latest studies by Huisman, (2020) investigated the effectiveness of simple imputation techniques on conservation of structural properties of the network. According to Huisman's results, simple imputation techniques did not achieve successful correction of missing values apart from a few specific situations.

To the author's best knowledge, no previous study has evaluated multiple imputation (MI) performance with REM. MI is a method that can conserve the data and potentially produce

unbiased results (van Buuren, 2018). The MI process consists of the following steps: creating multiple completed datasets, conducting a statistical analysis on each of the completed datasets, and then pooling the results to produce the final unbiased estimate and standard errors. The pooling is performed by "Rubin's rules". The pooled variance is estimated by combining the within-imputation variance with the between-imputation variance caused by the missing values (van Buuren, 2018; Enders, 2022). In the following sections the problem of missing values in data and the method of addressing them - MI will be introduced. Then, the method of simulating missingness and the process of multiple imputation will be described. In order to make the design of the study as straightforward as possible, only missingness in one column (sender) will be considered, as a first step in addressing this issue. Hence, I propose the following research question: can multiple imputation with REM be applied on REH data with missing values in the sender variable, to produce valid inference?

2. Data

The dataset that was analysed in the current study is a part of Apollo 13 communication records between the astronauts and control team. Apollo 13 was a mission conducted by NASA within the Apollo space program. Apollo 13 is an infamous mission that was cut short after just circling the moon due to an oxygen tank exploding. Due to the extreme circumstances the communication that took place was out of the ordinary as well. The famous words "Houston we've had a problem" originated during the Apollo 13 mission (NASA, 2022; Kamalabad, Leenders & Mulder, 2023). The dataset is publicly accessible from Apollo 13 Real Time.

Because of how unconventional this situation is, the communication records from this mission present an especially interesting case for social network research. This data represents a relational event sequence where each row is a single, directed communication event with a time point, sender, and receiver (see Table 1). Sender column represents the actors that had initiated the communication event at a corresponding time point. Receivers are the targets of the communication event. The subset that was used for the current study consists of 3882 relational events (edges), each corresponding to a separate timestamp and 16 actors (nodes). The dataset is fully observed and did not require any further processing before analysing.

This dataset is anonymised (actors' names are substituted with numbers) and publicly available, therefore the dataset does not bear any ethical or legal considerations.

3. Methods

3.1 Relational Event Model

REM allows researchers to study how the history of interaction influences the future probability of interaction through modelling the probability of occurrence of relational events at a certain time point. Who is going to interact and at which time point the interaction will occur is determined by the event rate λ. The event rate is a log linear function of statistics (Meijerink-Bosman, Back, & Geukes, 2023):

$$log\lambda(i,j,t) = \beta_1 x_1(i,j,t) + \beta_2 x_2(i,j,t) + \beta_3 x_3(i,j,t) + \ldots$$

Where β represents the magnitude of the effect of the statistics (x_1, x_2, x_3), *t* is the time point, *i.j* represent the pair of actors.

Statistics are defined as predictors that can be used to model the event rate. Statistics can encode both endogenous and exogenous predictors. Exogenous predictors refer to any variables that are external to the relational event history data, such as attributes and characteristics of the actors or the environment. On the other hand, endogenous predictors refer to variables that summarise the volume of occurrence of past interactions, such as inertia, reciprocity, etc., (Meijerink-Bosman, Back, & Geukes, 2023, Butts, 2008). As the data used in the current study is a subset of the data used by Kamalabad, Leenders, & Mulder, (2023). We decided to use the same endogenous statistics. The analysed statistics were reciprocity, indegree sender, and outdegree receiver.

Reciprocity

Reciprocity refers to the tendency of actors to reciprocate contact. Positive reciprocity coefficient would point to a higher probability of an actor B initiating a communication event directed at actor A if previously actor A has initiated a communication event directed at actor

B. Reciprocity is modelled as a function of frequencies of past interactions (Butts & Marcum, 2017; Kamalabad, Leenders & Mulder, 2023).

Indegree sender

Indegree refers to the number of edges that point towards a node representing a number of connections. In-degree sender effect points to a higher probability of a node being an initiator of contact in the future if the node has been often a target of communication events in the past (Kamalabad, Leenders & Mulder, 2023). For example, if node A has a high indegree (number of times the node has been a target of communication events), then node A will have a higher probability of being a sender (a node that initiates a communication event) in the future.

Outdegree receiver

Outdegree refers to the connections pointing towards other nodes. Nodes with high outdegree have many edges pointing outwards. Positive outdegree receiver indicates that nodes that have had high outdegree in the past, meaning had initiated many contacts, will have a higher probability of being a target of a communication event in the future. (Kamalabad, Leenders & Mulder, 2023). For example, if a node B has initiated a lot of communication events in the past and therefore has a high outdegree, then node B will have a higher probability of being a receiver.

To calculate the statistics, first, for every time point (*t*) a 'risk set' needs to be constructed. A risk set is a list of all potential events for a specific time point N(N-1). The statistics are calculated for every possible pair of nodes in the risk set (Meijerink-Bosman, Back, & Geukes, 2023)

3.2 Multiple Imputations

Missing values can occur due to several reasons: participants' refusal to answer certain questions or refusal to participate all together, drop-out of participants before the end of the study, malfunctioning of the software or hardware that is used to collect the information, etc. Not all missing values are the same but can be distinguished through their missingness mechanism. There are three mechanisms of missingness, which are: missing completely at random (MCAR), missing at random (MAR) and missing not at random (MNAR). MCAR

refers to occurrences where all values have the same probability of being missing, which means that missingness is not related to the data. The data is considered MCAR when:

$$Pr(R = 0|\, Yobs, Ymis, \psi) = Pr(R = 0|\psi)$$

Where R denotes the data of the population, Y refers to the data of the sample, Y_{obs} is observed data, and Y_{mis} is the missing data, and ψ represent the parameters of the missing data model. The notation used in this thesis is the same as van Buuren's (2018).

If the probability of being missing is the same conditional on information that is present in the data, the missingness mechanism is said to be MAR. Therefore, conditional on information that is observed, MCAR becomes MAR. The data is considered MAR when:

$$Pr(R = 0|\, Yobs, Ymis, \psi) = Pr(R = 0|\, Yobs, \psi)$$

MNAR refers to a missingness mechanism that is dependent on the data, however, this information is not in the data. Therefore, the probability of data being missing varies, but for unknown reasons. The data is considered MNAR when:

$$Pr(R = 0|\, Yobs, Ymis, \psi)$$

One of the methods of dealing with missing values in the data is multiple imputation. MI is defined as 'state-of-the-art technique for drawing valid conclusions from incomplete data' (Oberman & Vink, 2023, p. 1). In broad terms, multiple imputation is a method of filling in missing values with estimates of what could have been if the value would have been observed (Oberman & Vink, 2023). The process of multiple imputation is as follows: first, multiple imputation creates multiple complete datasets, then the chosen statistical analysis is applied to each of the complete datasets, and finally the results of the analysis are pooled into a final estimate and standard error. The process of creating complete datasets from incomplete datasets involves creation of multiple versions of complete datasets by means of replacing the missing values with the values that were drawn from a distribution that is modelled specifically for each missing value. The statistical analysis that is performed on the complete datasets in this study is REM. Pooling is an important process in multiple imputation and is not done by simple averaging of the estimates but is calculated so the standard errors reflect the uncertainty that comes with the presence of missing values in the data. The pooled variance is a result of combining the conventional within-imputation variance and the between-imputation variance which is the result of missing values. Pooling of the estimates

of multiple imputed datasets requires the assumption of normal distribution of the estimates (Rubin 1987; van Buuren, 2018).

The reason why MI is a unique method of dealing with missing values is because it reflects the uncertainty that is inherent to missing values. The uncertainty of the imputed values is expressed as a variation of the values across the imputed datasets. The larger the variation of the imputed values across the imputed datasets, the larger the uncertainty. Due to the presence of missing values in the sample it is impossible to attain the estimand, therefore the goal of multiple imputation is to attain an estimate that is unbiased and confidence valid (van Buuren, 2018; Rubin, 1996).

Van Buuren (2018) proposes several measures that can help evaluate the statistical validity of MI. The measures that will be used in this study are raw bias (RB), percent bias (PB) coverage rate (CR), and average width (AW). RB of the estimate refers to the difference between the estimate and the truth. The closer to zero the RB is, the more evidence there is for the validity of the process. PB is calculated by dividing the RB by the true value and multiplying that by hundred. PB should not extend 5%. CR is defined as the proportion of times that the true value falls within the range of the multiply imputed values. Preferably, this would occur 95% of the time. In case the CR falls below the nominal rate, it would indicate that the process is too optimistic leading to potential false positives. In case the CR is substantially above the nominal rate it means that the imputation method is inefficient and produces inferences that are too conservative. The latter case is preferred over the former one. AW is calculated by taking the average of the confidence intervals across the multiple imputations. AW serves as an indicator of statistical efficiency and should be as small as possible, but not so small as to cause CR to fall below the desired level.

3.3 Data Analysis

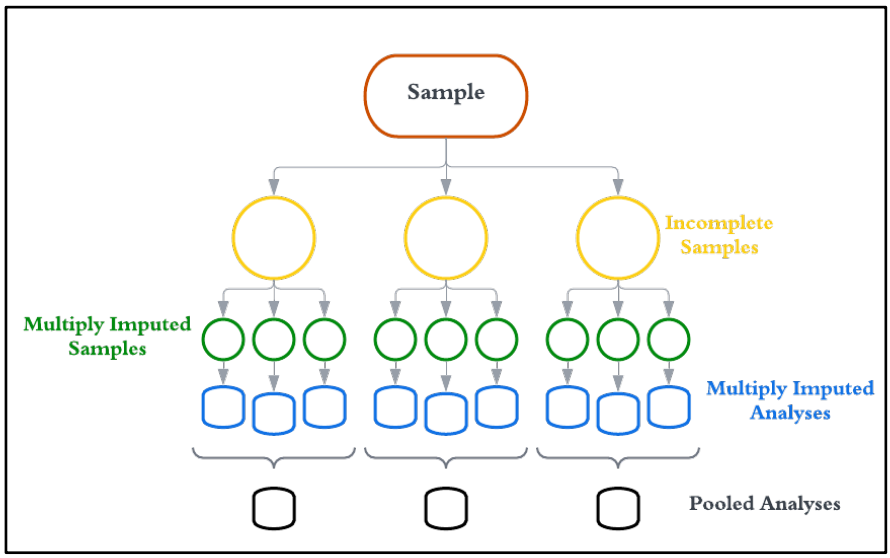
In order to evaluate whether multiple imputation is a viable and preferred approach for dealing with missing values in REH data, REM analysis of fully observed data and REM on complete cases was compared to REM analysis on multiply imputed data.

To perform REM analysis on the fully observed data, reciprocity, indegree sender, and outdegree receiver statistics had to be calculated first. To calculate the statistics remify

(Arena, 2023) and remstats (Meijerink-Bosman et al, 2023) packages were used. After calculating the statistics, cox proportional hazard model from survival package (Therneau, 2023) was used to perform REM analysis.

In order to perform a multiple imputation, missingness had to be introduced to the fully observed data first. Missingness simulation, multiple imputation, pooling, and averaging were performed in accordance with "Strategies for simulating missingness" (Vink, 2022). Because a single finite observed dataset was used for the simulation study, sampling variance was not considered and was excluded from the evaluation of the performance of imputations (Vink, 2022). See figure 2 for the schematic representation of the process. The red circle represents the fully observed Apollo 13 dataset. The yellow circles correspond to the multiple amputed dataset that were a result of the missingness simulation. Green circles represent the completed datasets that were created with multiple imputation. And finally black circles represent the pooled estimates of the multiple imputations, which means that every simulation produced one pooled final point estimate. Missingness was simulated a hundred times, to ensure the validity of the results. Each, out of a hundred dataset, had a slightly different missingness pattern.

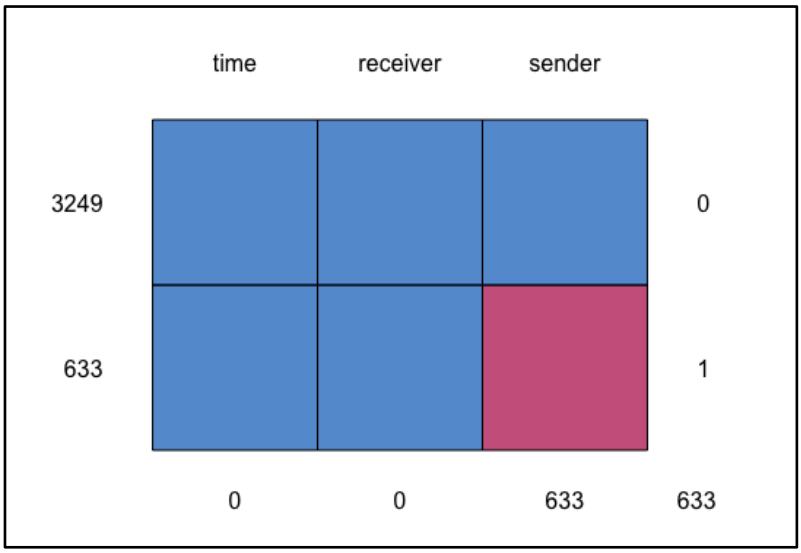
Figure 2. Schematic representation of missing data simulation using finite population according to Vink (2022).



Note. From "Strategies for simulating missingness", by G., Vink, 2022. Retrieved from: https://www.gerkovink.com/simulate/

The amputation was performed using the MICE package, which stands for Multiple Imputation by Chained Equations (van Buuren & Groothuis-Oudshoorn, 2011). MICE allows selection of proportion of missing values as well as the missingness mechanism, and pattern. Figure 3 shows that after amputation, missing values constituted 20% of the dataset and only existed in the 'sender' column which is the focus of this study. The proportion of missingness was selected to constitute 20% in order to introduce a substantial amount of missingness without risking the reliability of final estimates. For the missingness mechanism for the amputation process, MCAR was selected. Although MCAR is not always reasonable to assume, it is a stepping stone towards assessing the validity of the method. If the results suggest that MI cannot perform effectively under MCAR, likely it will not perform under MAR either. It is important to note that the missingness was not fully random. During simulation of amputations some datasets would end up having fewer actors in the sender column. Some actors appear rarely in the dataset therefore, when missingness was introduced, in some cases all instances of one of the actors was completely removed from the dataset. In this case the amputed dataset had only 15 actors appear instead of 16. The MI method that was applied in this study can only impute values that appear in the amputed dataset. Therefore, some of the complete datasets also included only 15 actors. Less actors result in fewer dyads and a smaller risk set. A risk set for 16 actors consists of 240 possible communication events while the risk set for 15 actors consists of 210 communication events. This discrepancy between some of the amputed datasets made REM analysis impossible on some of the completed datasets. To remedy this, a part of the observed dataset (1500 observations) was conserved and not amputed. This ensured that all actors were present in the amputed sample.

Figure 3. Missingness pattern in one of simulations after amputation of the Apollo 13 data. The rows represent the missingness patterns occurring in the dataset. The most common pattern - 3249 rows are fully observed. The other pattern - 633 rows have a missing value in the sender column (red square represents missing values)

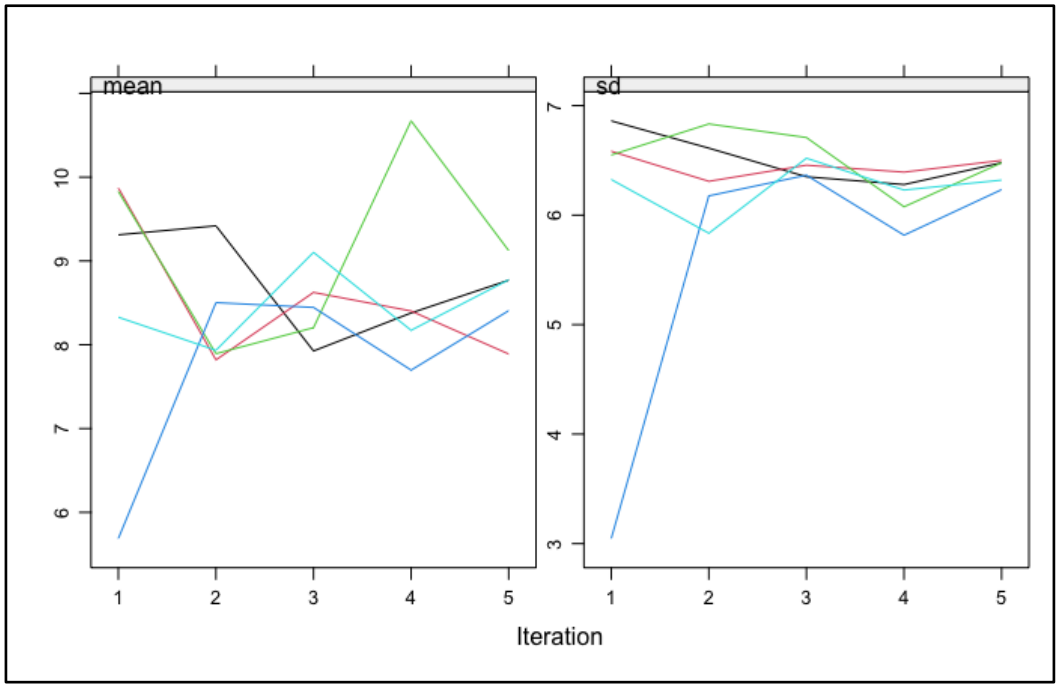


Because complete case analysis is often a chosen method for handling missing values when conducting REM, the amputed datasets were used to perform REM on complete cases before performing multiple imputations. A hundred Apollo 13 datasets with missingness in the sender column were analysed with REM whereby all relational events that contained missing values were excluded from the analysis. The results were later averaged across the analyses to represent the final estimate for complete case analysis.

The datasets with simulated missingness were also used to perform multiple imputations using MICE. There are several parameters that must be considered when setting up a multiple imputation in MICE: number of imputations, number of maximum iterations, method for multiple imputation, and which variables are used as predictors for the missing values. The number of imputations as well as maximum iterations was set to five. Number of imputations determine the number of complete datasets that will be created with multiple imputations. Although it is often suggested that the number of imputations should be set to a higher number (van Buuren, 2018), the computational efficiency had to be considered. Therefore, a total number of five multiple imputations was selected. The number of iterations determine the number of cycles that multiple imputation runs to refine the estimates of missing values. In each iteration, the variable that was specified gets imputed using the predictor variables (van Buuren, 2018). The number of iterations can be increased to ensure proper convergence, however, typically a low number of iterations, between 5 and 20 is considered to be able to achieve convergence (van Buuren, 2018). A convergence plot for one of the simulations can be seen in figure 4. The plot shows mean (left panel) and standard deviation (sd) (right panel) for imputed values in the sender column across five imputations and five iterations.

Convergence is evaluated using two criteria: mixing and stationarity. Mixing refers to the intermingling of the imputation values and stationarity refers to the absence of a clear trend across iterations (Gelman et al, 2013; Oberman, van Buuren, & Vink, 2021). In figure 4 (left panel) a slight deviation from a typical convergence can be seen. One of the imputations is not mixing very well with the rest. However, the right panel shows that the convergence was achieved after the second iteration. It is possible that the MI would benefit from a higher number of iterations, however the computational efficiency had to be prioritized.

Figure 4. Trace line plots of means (left) and standard deviations (right) of imputed values for the sender column in one of the simulations.



For the method of multiple imputation, predictive mean matching was used. Predictive mean matching "calculates the predicted value of a target value according to the specified imputation model" (van Buuren, 2018, Chapter 3.4). From all complete cases in which predicted values are closest to the predicted values of the missing observation, for each missing value a small set of candidate donors is formed. Then one donor is randomly drawn from the candidate set and the observed value of the donor replaces the missing observation. Predictive mean matching assumes that the distribution of the missing value is the same as the distribution of the observed candidate values. However due to the structure and nature of the data a custom method had to be created in order to properly impute the Apollo 13 data. This method had to avoid creation of loops, whereby one actor is initiating a contact with

themselves. Therefore, a conditional predictive mean matching was used, where candidate donors cannot include the same value as the corresponding receiver. Finally, for the predictor variables, only the receiver variable was used because when the time variable was included as a predictor it was creating loops in the imputed datasets. After creating the completed datasets, REM analysis was performed on each set. As potential predictors of interactions reciprocity, outdegree receiver and indegree sender were included.
Finally, all the REM results were pooled, and the pooled results were averaged across a hundred simulations.

The code to perform simulations and analyse the data is available in the appendix.

4. Results

REM results of the fully observed subset of Apollo 13 data can be seen in table 2. The effects of all three statistics are close to zero and only indegree sender has a statistically significant effect on the hazard rate of the future interactions. Which means that there is no evidence to suggest propensity of reciprocating past interactions. Neither there is evidence to point to the higher probability of actors to be a target of a communication event if they have initiated contact in the past. The effect for indegree sender is small, but statistically significant and suggests higher probability of initiating contact for actors who have been contacted more in the past.

Table 2. REM results on the fully observed Apollo 13 data.

Statistic	Estimate	Standard error	p-value
Reciprocity	0.0235	0.0185	0.204
Indegree Sender	0.0004	74.0×10^{-6}	<.001
Outdegree Receiver	-91.15×10^{-6}	74.4×10^{-6}	0.220

REM results of the amputed, but not yet multiply imputed datasets can be seen in table 3. All effects are slightly overestimated compared to the results of the model on the fully observed data. Standard errors are also larger than in the true model. The significance of the p-values

remained the same – only effect for indegree sender is significant in both models. Last column shows the RB of the complete case analysis estimates compared to the true value. RB is small across all effects.

Table 3. Complete case REM results averaged across 100 simulations of missingness.

Statistic	Estimate	Standard error	p-value	bias
Reciprocity	0.0261	0.0259	0.319	0.0026
Indegree Sender	0.0008	0.0002	<.001	0.0004
Outdegree Receiver	-0.0002	0.0002	0.390	-85.77×10^{-6}

Averaged REM results after MI across hundred simulations can be seen in table 4. 2.5% and 97.5% represent the lower and upper boundary of the confidence intervals. Cov stands for coverage rate and shows the proportion of times that confidence intervals covered the 'true' value. The raw bias is very small, close to zero across all effects. Notably, it is smaller than in the case of complete case analysis, across all effects. The PB for reciprocity, indegree sender, and outdegree receiver is also below 5%: 2.33%, 0.17%, and 1.06% respectively (refer to section 3.2 for the description of how to calculate RB). Moreover, for all three effects, the true value is found within the confidence intervals 100% of the time. Coverage rate higher than 95% is suboptimal, but acceptable. AW of confidence intervals is also very close to zero for all effects: reciprocity (0.0035); indegree sender (5.44×10^{-6}); outdegree receiver (19.0×10^{-6}) (refer to section 3.2 for the description of AW). Notably, the MI method produced false significant p-values. Unlike in REM on fully observed data and REM on complete cases p-values for reciprocity and outdegree receiver are significant here. This could have been caused by the conservation of part of the dataset while introducing missingness in the dataset.

Table 4. REM results averaged across 100 simulations of missingness and 5 multiple imputation sets.

Statistic	Estimate	Standard error	p-value	2.5%	97.5%	Cov	Bias

Reciprocity	0.0241	0.0006	<.001	0.0223	25.9×10^{-3}	0.99	0.0005
Indegree Sender	0.0004	980×10^{-9}	<.001	0.0004	433×10^{-6}	1.00	-714×10^{-9}
Outdegree Receiver	-92.1×10^{-6}	3.43×10^{-6}	<.001	-0.0001	-82.6×10^{-6}	1.00	-968×10^{-9}

In figures 3-5, the distribution of estimates after MI for the investigated effects can be seen. The distribution for reciprocity is slightly skewed to the right and the distribution for indegree sender is slightly skewed to the left. The distribution for the estimates of outdegree receiver appears to be normal. However, after visual inspection of quantile-quantile plots (Figures 1-3 in Appendix) it appears that neither of the distributions is normal. In addition, the results of Kolmogorov-Smirnov test for reciprocity ($D = 0.51$, $p < 0.001$), indegree sender ($D = 0.50$, $p<0.001$), and outdegree receiver ($D = 0.50$, $p<0.001$) showed that the hypothesis that the distributions of the estimates are normal must be rejected. The skewed distributions violate the normality assumption of Rubin's rules. Not meeting the normality assumption could result in overestimated standard errors and biased estimates.

Figure 3. Density plot of estimates for reciprocity of 100 simulations with 5 multiple imputations each.

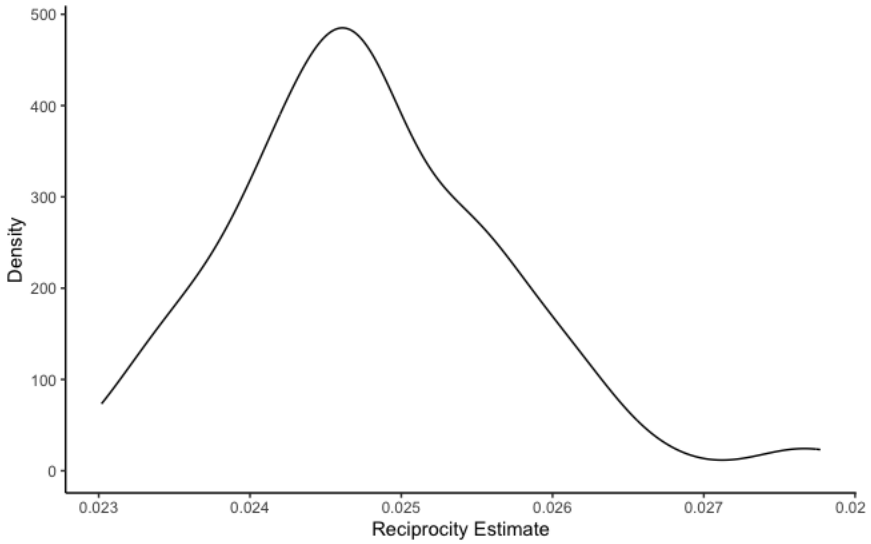


Figure 5. Density plot of estimates for indegree sender of 100 simulations with 5 multiple imputations each.

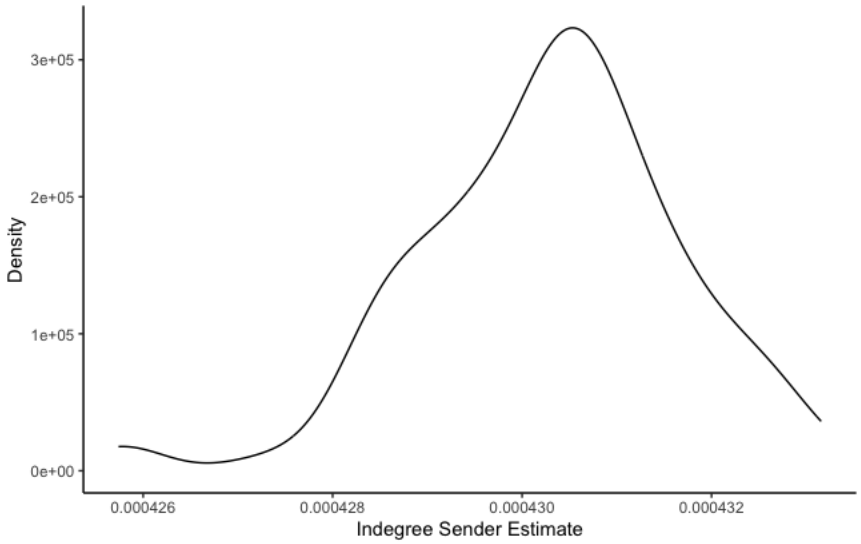
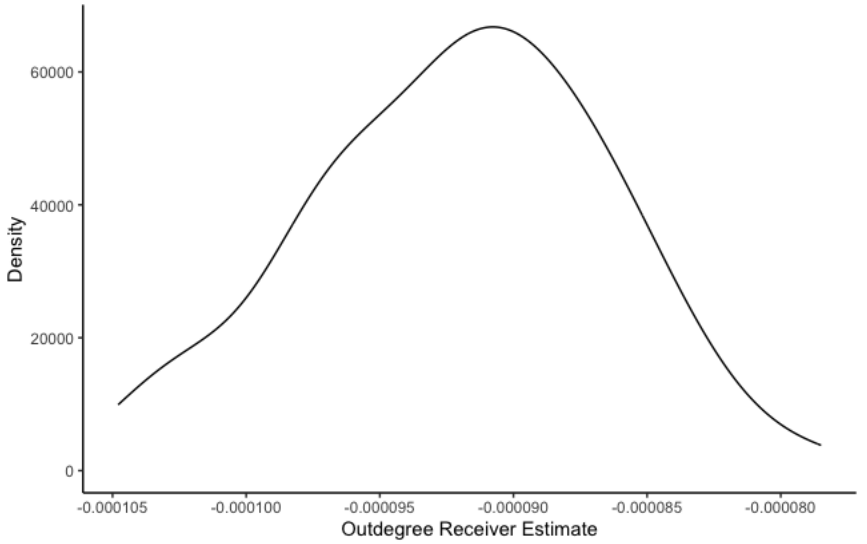


Figure 4. Density plot of estimates for outdegree receiver of 100 simulations with 5 multiple imputations each.



5. Discussion and Conclusion

Missing data had an identifiable effect on the results of the analysis. Similarly to other studies (Huisman, & Steglich, 2008; Gile, & Handcock, 2006), complete case analysis produced biased inferences. In the current study, complete case analysis overestimated the effects of the predictors and produced larger standard errors. As an alternative to complete case analysis, this study has performed MI to handle the missing values. Based on the criteria for

evaluation of MI RB, PB, CR and AW, the MI with REM appears to be a valid approach for addressing the missing data in REH. The raw and percent biases are close to zero across all three effects. The CR is too high, suggesting that the confidence intervals are too wide. However, conservative inferences that always cover the true value are pointing to statistical insufficiency and not invalid inference (van Buuren, 2018). Therefore, the CR is acceptable. AW for all three effects appears to be very small. It is important to note that the effects of the model are all close to zero, therefore it is difficult to reasonably assess the widths of the intervals. Perhaps analysing the full Apollo 13 dataset and not just the subset would have produced larger estimates.

To summarise, the MI approach appears to have performed better than complete case analysis. The estimates of MI have lower RB, and the standard errors are smaller. However, although according to van Buuren's evaluating criteria the MI method appears to be valid, unlike complete case analysis MI have caused false statistical significance of the p-values. According to the results of REM analysis on fully observed data, only the effect for indegree sender is statistically significant. Yet, in the MI with REM all the effects are statistically significant. This could be due to the pooling rules that were used for variance calculation which resulted in the underestimation of variance and standard errors creating false significant p-values. Because the sampling variance was excluded due to the use of finite populations, the variance was calculated as follows: $T = \frac{B+B}{m}$ (where T denotes total variance, B the variance between imputations, and *m* stands for the number of multiple imputations). This has caused the underestimation of variance due to the introduction of a constant (conserved part of the dataset) during the amputation process. Notably, if conventional Rubin's rules for pooling were applied (including the sampling variance), it would have likely overestimated the variance. Identifying correct variance pooling remains a task for future research. Furthermore, the density plots and Kolmogorov-Smirnov test results showed slight deviation from the assumption of normal distribution. However, the deviation from normal distribution is not substantial, therefore, most likely does not constitute a problem for the validity of the pooled estimates.

This study has several limitations. First, MCAR often cannot be reasonably assumed for the data at hand. Therefore, it is essential to test MI methods on the data that have missing values created under the MAR mechanism. Second, in this study only three statistics were

investigated. Future studies could benefit from considering more statistics, for example inertia, or enriching the data with attribute variables and using them to calculate exogenous statistics. Third, the conservation of the part of the dataset had a negative effect on the variance estimates which must be addressed in future research. In addition, the performance of the multiple imputation could have suffered from the small number of imputations and iterations. Future studies should consider increasing both to achieve optimal performance of MI. Additionally, in this study the time variable was removed because it appeared to create loops in the completed datasets. In the future studies this should be explored. It is also important to mention that this study only considered missingness in the sender column. Future research should also investigate the effect of missing values and MI in the time and receiver columns as well as in multiple columns at the same time. Lastly, the small effect sizes of the estimated parameters from the relational event model make it difficult to assess certain performance criteria of the imputation (AW, RB). Future research should consider conducting a simulation study on the full Apollo 13 dataset.

The main goal of the current study was to determine whether multiple imputation with REM can produce unbiased results. Based on the results of this study, multiple imputation, even if only under MCAR missingness assumption, can produce satisfactory results. However, more research is needed to address the limitations of this study.

References

Apollo 13 Real-time, (n.d.). http://apollo13realtime.org/.
Arena G (2023). _remify: Processing and transforming REH to formats suitable for the remverse packages and more_. R package version 3.0.0, <https://github.com/TilburgNetworkGroup/remify>.

Bearman, P. S., Moody, J., and Stovel, K. (2004). Chains of affection: The structure of adolescent romantic and sexual networks, Am. J. Sociol. 110, 44–91 .

Bernard, H. R., Johnsen, E. C., Killworth, P. D., and Robinson, S., Estimating the size of an average personal network and of an event population, in M. Kochen, ed., The Small World, pp. 159–175, Ablex Publishing, Norwood, NJ (1989).

Boyer, L., Belzeaux, R., Maurel, O., Baumstarck-Barrau, K., & Samuelian, J. C. (2010). A social network analysis of healthcare professional relationships in a French hospital. *International Journal of Health Care Quality Assurance*, *23*(5), 460-469.

Burt, R. S. (1987). A note on missing network data in the general social survey. Social Networks, 9(1), 63–73. https://doi.org/10.1016/0378-8733(87)90018-9

Butts, C.T. (2008). A relational event framework for social action. Sociological Methodology, 38 (1), 155–200. https://doi-org.proxy.library.uu.nl/10.1111/j.1467-9531.2008.00203.x

Butts, C.T., Marcum, C.S. (2017). A Relational Event Approach to Modeling Behavioral Dynamics. In: Pilny, A., Poole, M. (eds) Group Processes. Computational Social Sciences. Springer, Cham. https://doi.org/10.1007/978-3-319-48941-4_4

Enders, C. K. (2022). *Applied missing data analysis*. Guilford Publications.

Faisal, S., & Tutz, G. (2021). Multiple imputation using nearest neighbor methods. Information Sciences, 570, 500–516. https://doi.org/10.1016/j.ins.2021.04.009

Fattore, G., Frosini, F., Salvatore, D., & Tozzi, V. (2009). Social network analysis in primary care: the impact of interactions on prescribing behaviour. Health Policy, 92(2-3), 141–148. https://doi.org/10.1016/j.healthpol.2009.03.005

Gelman A, Carlin JB, Stern HS, Dunson DB, Vehtari A and Rubin DB (2013) *Bayesian Data Analysis*. Philadelphia, PA, United States: CRC Press LLC.

Grossman, J. W. (2002). The evolution of the mathemati- cal research collaboration graph, Congr. Numer. 158, 202–212.

Grunspan, D. Z., Wiggins, B. L., & Goodreau, S. M. (2014). Understanding classrooms through social network analysis: A primer for social network analysis in education research. *CBE—Life Sciences Education*, *13*(2), 167-178.

Hawe, P., & Ghali, L. (2008). Use of social network analysis to map the social relationships of staff and teachers at school. *Health Education Research*, *23*(1), 62–9.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). An introduction to statistical learning: with applications in r (Second, Ser. Springer texts in statistics). Springer. https://doi.org/10.1007/978-1-0716-1418-1

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.

Landherr, A., Friedl, B. & Heidemann, J. Eine kritische Analyse von Vernetzungsmaßen in sozialen Netzwerken.(2010). *WIRTSCHAFTSINFORMATIK* **52**, 367–382. https://doi-org.proxy.library.uu.nl/10.1007/s11576-010-0244-0

Leifeld, P., Cranmer, S. J., & Desmarais, B. A. (2018). Temporal exponential random graph models with btergm: Estimation and bootstrap confidence intervals. Journal of Statistical Software, 83(6), 1–36.

Lusher, D., Koskinen, J. and Robins, G. (2012), Exponential random graph models for social networks: Theory, methods, and applications, Cambridge University Press.

Meijerink-Bosman M, Arena G, Karimova D, Lakdawala R, Shafiee Kamalabad M, Generoso Vieira F (2023). _remstats: Computes Statistics For Relational Event History Data_. R package version 3.1.0, <https://github.com/TilburgNetworkGroup/remstats>.

Meijerink-Bosman, M., Back, M., Geukes, K. et al. (2023). Discovering trends of social interaction behavior over time: An introduction to relational event modeling. Behav Res 55, 997–1023. https://doi-org.proxy.library.uu.nl/10.3758/s13428-022-01821-8

NASA Administrator, (2022). Apollo 13. *NASA*. https://www.nasa.gov/mission_pages/apollo/missions/apollo13.html#

Newman, M. E. J. (2018). Networks. Oxford University Press.

Oberman, H. I., & Vink, G. (2023). Toward a standardized evaluation of imputation methodology. Biometrical Journal. Biometrische Zeitschrift, E2200107, 2200107. https://doi.org/10.1002/bimj.202200107

Oberman, H. I., van Buuren, S., & Vink, G. (2021). Missing the point: Non-convergence in iterative imputation algorithms. *arXiv preprint arXiv:2110.11951*.

Pilny, A., Schecter, A., Poole, M. S., & Contractor, N. (2016). An illustration of the relational event model to analyze group interaction processes. *Group Dynamics: Theory, Research, and Practice*, *20*(3), 181–195.

Rubin, D. B. (1987). Multiple imputation for nonresponse in surveys (Ser. Wiley series in probability and mathematical statistics: applied probability and statistics). Wiley. https://doi.org/10.1002/9780470316696

Rubin, D. B. (1996). Multiple imputation after 18 years. Journal of the American Statistical Association, 91(434), 473–489.

Scott, J., & Carrington, P. J. (Eds.). (2011). *The sage handbook of social network analysis*. SAGE. Retrieved July 3, 2023, from https://ebookcentral.proquest.com/lib/uunl/detail.action?docID=786857.

Serrat, O. (2017). Social Network Analysis. In: Knowledge Solutions. Springer, Singapore. https://doi-org.proxy.library.uu.nl/10.1007/978-981-10-0983-9_9

Snijders, T.A.B., van de Bunt, G.G. & Steglich, C.E.G. (2010) Introduction to stochastic actor-based models for network dynamics. Social Networks, 32, 44–60

Therneau T (2023). _A Package for Survival Analysis in R_. R package version 3.5-5, <https://CRAN.R-project.org/package=survival>.

van Buuren, S. (2018). Flexible imputation of missing data. CRC press.

van Buuren S, Groothuis-Oudshoorn K (2011). "mice: Multivariate Imputation by Chained Equations in R." Journal of Statistical Software, 45(3), 1-67. doi:10.18637/jss.v045.i03.

Vink G., (2022). Strategies for simulating missingness (v1.0). Zenodo. https://doi.org/10.5281/zenodo.7467995

Wang, C., Butts, C. T., Hipp, J. R., Jose, R., & Lakon, C. M. (2016). Multiple imputation for missing edge data: a predictive evaluation method with application to add health. Social Networks, 45, 89–98. https://doi.org/10.1016/j.socnet.2015.12.003

Appendix

Figure 1. Quantile-quantile plot for distribution of estimates for reciprocity effect

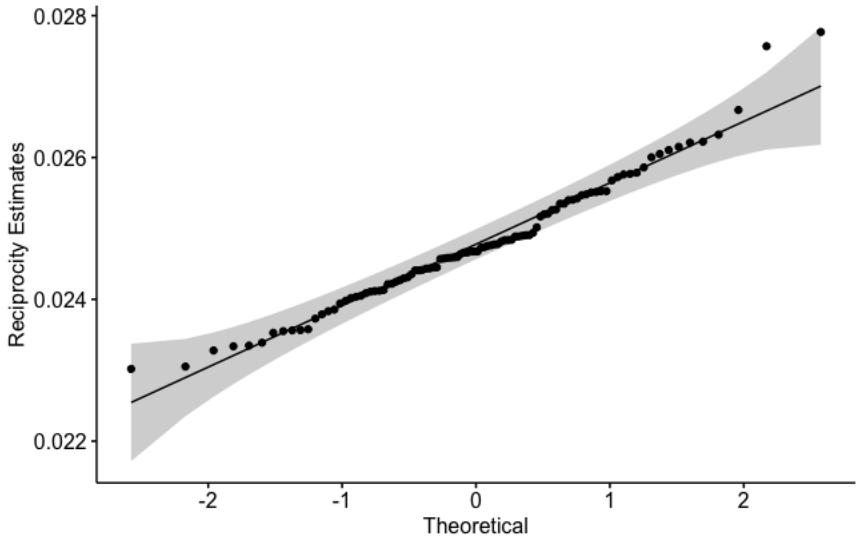


Figure 2. Quantile-quantile plot for distribution of estimates for indegree sender effect

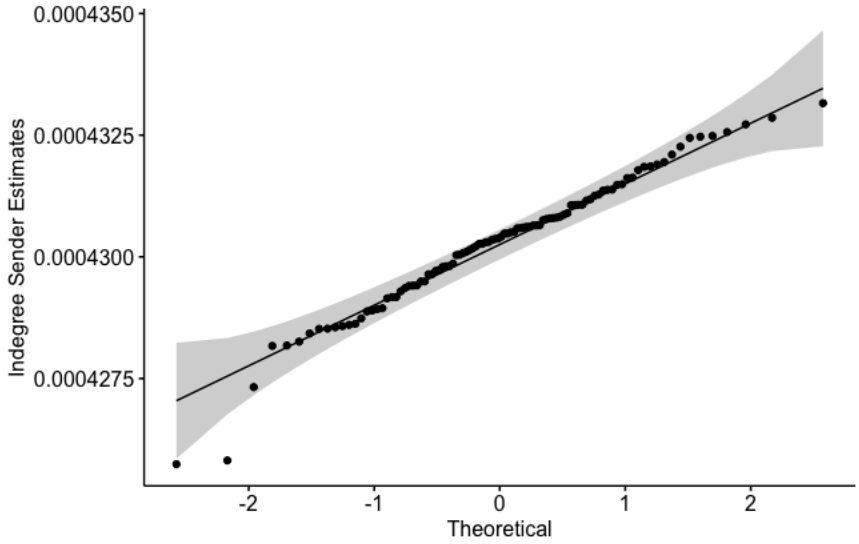
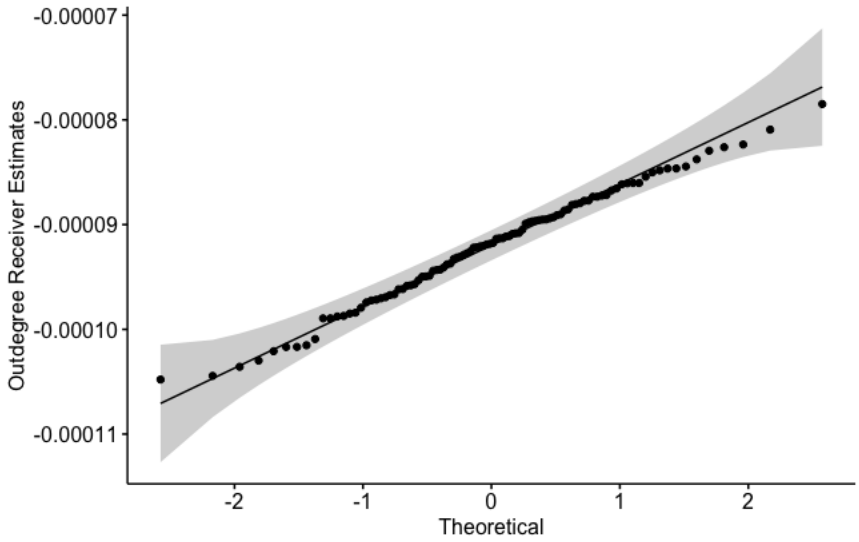


Figure 3. Quantile-quantile plot for distribution of estimates for outdegree receiver effect


-0.00007
-0.00008
-0.00009
-0.00010
-0.00011
Outdegree Receiver Estimates
-2
-1
0
1
2
Theoretical

Code for the R analysis

```
library(mice, warn.conflicts = FALSE)
require(lattice)
library(tidyverse)
library(magrittr)
library(dplyr)
library(purrr)
library(furrr)
library(relevent)
library(broom.mixed)
library(rem)
library(ggplot2)
library(survival)
library(remify)
library(remstats)
library(devtools)
library(tibble)

## 1. Read the data
load("UUsummerschool.Rdata")
rm(Twitter_data_rem3, WTCPoliceCalls, ClassIntercept, ClassIsFemale,
   ClassIsTeacher, WTCPoliceIsICR, Class) # remove data that is not used


## 2. Ampute and then impute the data
# renaming dataset for later convenience
apollo.renamed <- PartOfApollo_13
apollo.renamed <- apollo.renamed %>%
  rename(
    actor1 = sender,
    actor2 = receiver
  )

# making the dataset a tibble
apollo.renamed <- as_tibble(apollo.renamed)

# determine which column to condition on
whichcol <- c("", "actor2", "actor1")
names(whichcol) <- colnames(apollo.renamed)
```

```
# create predictor matrix for imoutations
pred <- make.predictorMatrix(apollo.renamed)
pred[ ,"time"] <- 0
pred

# use the custom pmm method
method <- make.method(apollo.renamed)
method[c(2,3)] <- "pmm.conditional"

#### set with sufficient actors & dyads
set.seed(123) # fix seed to realize a sufficient set

indic <- sample(1:nrow(apollo.renamed), 1500)
remify(apollo.renamed[indic, ], model = "tie") %>% dim()



#### Combine the sufficient set and the incomplete set
make_missing <- function(x, indic){
  sufficient <- x[indic, ]
  miss <- x[-c(indic), ] |>
    ampute(prop = .8,
           mech = "MCAR",
           patterns = c(1,0,1)) %>%
    .$amp
  combined <- rbind(sufficient,
                    miss)
  return(combined[order(combined$time), ]) # sort it all like apollo
}
# simulate 100 datasets with missingness in the sender column with MCAR mechanism then
ampute the data with the conditional pmm and 5 multiple imputations and 5 itterations,
exclude time column as a predictor

MCAR_finite <- furrr::future_map(1:100, ~ { # map over 100 sims
```

```
  apollo.renamed %>%
    make_missing(., indic) %>%
    mice(m = 5,
         maxit = 5,
         method = method,
         pred=pred,
         whichcolumn = whichcol,
         print = F)
}, .options = furrr_options(seed = 123))

#check if it is correct
# MCAR_finite |> map(~.x %>%
#   complete("long") |>
#   summarize(all(actor1 != actor2)))



# checking convergence
convergence <- lapply(MCAR_finite, plot) # plot means and sd for every simulation

plot(MCAR_finite[[58]],
     print=F,
     y = "actor1",
     layout = c(2,1)) # plot one of the plots

# Effects and function definition
# creating functions
# Define effects
effects <- ~ -1 + reciprocity(scaling = ("std")) +
  indegreeSender() + outdegreeReceiver()

# function to remify each imputed dataset

# code edited by Gerko
rem_function <- function(data) {
```

```
  # Perform the analysis as before
  remify::remify(edgelist = data, model = "tie") %>%
    remstats(tie_effects = effects)
}


# function to create a dataset for the cox model
cox_sets_function <- function(statsObject_imp, apollo_data) {
  # take the single riskset
  # remove the id column
  risk_sets <- attr(statsObject_imp, "riskset") %>% select(-'id')
  # creating one set with all risksets for each time point
  combined <- merge(risk_sets, apollo_data$time, by=NULL)  %>%
    rename(time = y) %>%
    .[, c("time", "sender", "receiver")] %>%
    mutate(sender = as.numeric(sender),
           receiver = as.numeric(receiver))

  # GV: Calculate divergence
  diff <- apollo_data[rep(seq_len(nrow(apollo_data)), 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
  reciprocity <- statsObject_imp[,,1]
  indegreeSender <- statsObject_imp[,,2]
  outdegreeReceiver <- statsObject_imp[,,3]

  recip.vector <- c(reciprocity)
  combined$reciprocity <- recip.vector
```

```
  indegSen.vector <- c(indegreeSender)
  combined$indegreeSender <- indegSen.vector

  outRec.vector <- c(outdegreeReceiver)
  combined$outdegreeReceiver <- outRec.vector

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

  return(combined)
}

########################## cox model on complete data ##########################
# Prepare event history
true.reh <- remify(edgelist = apollo.renamed,
                   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 <- cox_sets_function(stats, PartOfApollo_13)
# fit cox model
true.cox.fit <- coxph(Surv(time, status) ~ reciprocity + indegreeSender +
                        outdegreeReceiver,
                      data=true.cox.set)
true <- coefficients(true.cox.fit)

# prepare the data for the cox model
t1 <- Sys.time()
cox.models.sims <- MCAR_finite[1:10] %>%
  map(~.x %>% # for every simulated multiple imputation....
        complete("all") %>% # create a list of completed data sets
        map(~.x %>% # GV: changed into regular pipe - for every completed data set....
              rem_function() %>% # GV: removed .x - remify the imputed set
              cox_sets_function(apollo_data=PartOfApollo_13) %$%
```

```
        coxph(Surv(time, status) ~
                reciprocity +
                indegreeSender +
                outdegreeReceiver)) %>%
    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")) # create the sets for cox model
Sys.time() - t1

cox.models.sims2 <- MCAR_finite[11:20] %>%
  map(~.x %>% # for every simulated multiple imputation....
    complete("all") %>% # create a list of completed data sets
    map(~.x %>% # GV: changed into regular pipe - for every completed data set....
        rem_function() %>% # GV: removed .x - remify the imputed set
        cox_sets_function(apollo_data=PartOfApollo_13) %$%
        coxph(Surv(time, status) ~
                reciprocity +
                indegreeSender +
                outdegreeReceiver)) %>%
    pool(custom.t = ".data$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")) # create the sets for cox model

cox.models.sims3 <- MCAR_finite[21:30] %>%
  map(~.x %>% # for every simulated multiple imputation....
    complete("all") %>% # create a list of completed data sets
    map(~.x %>% # GV: changed into regular pipe - for every completed data set....
          rem_function() %>% # GV: removed .x - remify the imputed set
          cox_sets_function(apollo_data=PartOfApollo_13) %$%
          coxph(Surv(time, status) ~
                reciprocity +
                indegreeSender +
                outdegreeReceiver)) %>%
    pool(custom.t = ".data$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")) # create the sets for cox model

cox.models.sims4 <- MCAR_finite[31:40] %>%
  map(~.x %>% # for every simulated multiple imputation....
    complete("all") %>% # create a list of completed data sets
    map(~.x %>% # GV: changed into regular pipe - for every completed data set....
        rem_function() %>% # GV: removed .x - remify the imputed set
        cox_sets_function(apollo_data=PartOfApollo_13) %$%
        coxph(Surv(time, status) ~
            reciprocity +
            indegreeSender +
            outdegreeReceiver)) %>%
    pool(custom.t = ".data$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")) # create the sets for cox model

cox.models.sims5 <- MCAR_finite[41:50] %>%
  map(~.x %>% # for every simulated multiple imputation....
    complete("all") %>% # create a list of completed data sets
    map(~.x %>% # GV: changed into regular pipe - for every completed data set....
        rem_function() %>% # GV: removed .x - remify the imputed set
        cox_sets_function(apollo_data=PartOfApollo_13) %$%
        coxph(Surv(time, status) ~
            reciprocity +
            indegreeSender +
            outdegreeReceiver)) %>%
    pool(custom.t = ".data$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")) # create the sets for cox model

cox.models.sims6 <- MCAR_finite[51:60] %>%
```

```
  map(~.x %>% # for every simulated multiple imputation....
        complete("all") %>% # create a list of completed data sets
        map(~.x %>% # GV: changed into regular pipe - for every completed data set....
              rem_function() %>% # GV: removed .x - remify the imputed set
              cox_sets_function(apollo_data=PartOfApollo_13) %$%
              coxph(Surv(time, status) ~
                    reciprocity +
                    indegreeSender +
                    outdegreeReceiver)) %>%
        pool(custom.t = ".data$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")) # create the sets for cox model

cox.models.sims7 <- MCAR_finite[61:70] %>%
  map(~.x %>% # for every simulated multiple imputation....
        complete("all") %>% # create a list of completed data sets
        map(~.x %>% # GV: changed into regular pipe - for every completed data set....
              rem_function() %>% # GV: removed .x - remify the imputed set
              cox_sets_function(apollo_data=PartOfApollo_13) %$%
              coxph(Surv(time, status) ~
```

```
            reciprocity +
            indegreeSender +
            outdegreeReceiver)) %>%
    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")) # create the sets for cox model

cox.models.sims8 <- MCAR_finite[71:80] %>%
  map(~.x %>% # for every simulated multiple imputation....
    complete("all") %>% # create a list of completed data sets
    map(~.x %>% # GV: changed into regular pipe - for every completed data set....
        rem_function() %>% # GV: removed .x - remify the imputed set
        cox_sets_function(apollo_data=PartOfApollo_13) %$%
        coxph(Surv(time, status) ~
            reciprocity +
            indegreeSender +
            outdegreeReceiver)) %>%
    pool(custom.t = ".data$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")) # create the sets for cox model

cox.models.sims9 <- MCAR_finite[81:90] %>%
  map(~.x %>% # for every simulated multiple imputation....
    complete("all") %>% # create a list of completed data sets
    map(~.x %>% # GV: changed into regular pipe - for every completed data set....
        rem_function() %>% # GV: removed .x - remify the imputed set
        cox_sets_function(apollo_data=PartOfApollo_13) %$%
        coxph(Surv(time, status) ~
              reciprocity +
              indegreeSender +
              outdegreeReceiver)) %>%
    pool(custom.t = ".data$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")) # create the sets for cox model

cox.models.sims10 <- MCAR_finite[91:100] %>%
  map(~.x %>% # for every simulated multiple imputation....
    complete("all") %>% # create a list of completed data sets
    map(~.x %>% # GV: changed into regular pipe - for every completed data set....
        rem_function() %>% # GV: removed .x - remify the imputed set
        cox_sets_function(apollo_data=PartOfApollo_13) %$%
        coxph(Surv(time, status) ~
            reciprocity +
            indegreeSender +
            outdegreeReceiver)) %>%
    pool(custom.t = ".data$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")) # create the sets for cox model

#stitch the lists together
sims <- list(cox.models.sims,
             cox.models.sims2,
             cox.models.sims3,
             cox.models.sims4,
             cox.models.sims5,
             cox.models.sims6,
             cox.models.sims7,
             cox.models.sims8,
             cox.models.sims9,
             cox.models.sims10) %>%
  purrr::flatten() # make a single flat list instead of a list of lists

save(sims, file = "sims_moremissing.RData")

# Average sims

Reduce("+", sims) / length(MCAR_finite)



# Long data frame

reciprocity <- sims %>%
  map(~.x %>% .["reciprocity", ]) %>% #select row reciprocity
  do.call("rbind", .)
indegreeSender <- sims %>%
  map(~.x %>% .["indegreeSender", ]) %>%
  do.call("rbind", .)
outdegreeReceiver <- sims %>%
  map(~.x %>% .["outdegreeReceiver", ]) %>%
  do.call("rbind", .)
```

```
colMeans(reciprocity)


# Plot

library(ggplot2)

reciprocity %>%
  ggplot(aes(x = estimate)) +
  geom_density() + theme_classic() + labs(x ="Reciprocity Estimate", y = "Density")

outdegreeReceiver %>%
  ggplot(aes(x = estimate)) +
  geom_density() + theme_classic() + labs(x ="Outdegree Receiver Estimate", y = "Density")

indegreeSender %>%
  ggplot(aes(x = estimate)) +
  geom_density() + theme_classic() + labs(x ="Indegree Sender Estimate", y = "Density")

# calculate percent bias

av.resiprociy <- colMeans(reciprocity)
av.indegreeSender <- colMeans(indegreeSender)
av.outdefgreeReceiver <- colMeans(outdegreeReceiver)


PB.recip <- 100 * abs((av.resiprociy["estimate"] - true['reciprocity'])/ true['reciprocity'])
PB.indegSender <- 100 * abs((av.indegreeSender["estimate"] - true['indegreeSender'])/
true['indegreeSender'])
PB.outdegReciever <- 100 * abs((av.outdefgreeReceiver["estimate"] -
true['outdegreeReceiver'])/ true['outdegreeReceiver'])


# calculate average width
```

```
AW.recip <- av.resiprociy["97.5 %"] - av.resiprociy["2.5 %"]
AW.indegSend <- av.indegreeSender["97.5 %"] - av.indegreeSender["2.5 %"]
AW.outdegRecip <- av.outdefgreeReceiver["97.5 %"] - av.outdefgreeReceiver["2.5 %"]

# look at the distribution more closely
ggqqplot(reciprocity$estimate, ylab = "Reciprocity Estimates")
ks.test(reciprocity$estimate, "pnorm")

ggqqplot(indegreeSender$estimate, ylab = "Indegree Sender Estimates")
ks.test(indegreeSender$estimate, "pnorm")

ggqqplot(outdegreeReceiver$estimate, ylab = "Outdegree Receiver Estimates")
ks.test(outdegreeReceiver$estimate, "pnorm")



################ COMPLETE CASE ANALYSIS ################

MCAR_missing <- furrr::future_map(1:100, ~ { # map over 100 sims
  apollo.renamed %>%
    make_missing(., indic)}, .options = furrr_options(seed = 123))



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.renamed[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))

  return(combined)
}


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

```
            indegreeSender +
            outdegreeReceiver))

# disable scientific notation
options(scipen=999)

# 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[1:3]
  )
  rownames(df) <- c("reciprocity", "indegreeSender", "outdegreeReceiver")
  # Append the dataframe to the list
  results[[i]] <- df
}


# average the results across all simulations
average <- results %>%
  map(~.x %>%
        mutate(bias = coef - true) %>% # bias
        select(true, coef, se, p,
               bias)) %>%
  Reduce("+", .) / length(MCAR_missing)
```