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

Kasper Hermanns (5211905)
Mahdi Shafiee Kamalabad
Gerko Vink
Applied Data Science, Master Thesis, Utrecht University

July 7th, 2024

## Contents

## Abstract

**Background.** Relational Event History (REH) data is a valuable type of social network data due to its increasing availability and precise time resolution. However REH data is especially vulnerable to missing data, a small proportion of missing values lead to biased estimates which makes valid inference impossible. Traditionally missing values in REH data were handled using complete case analysis (CCA), which does not solve these issues. A potential solution can be using multiple imputation (MI), but research incorporating this method is scarce. This paper aims to continue on the work of an earlier simulation study by testing the effectiveness of MI in handling more complex missing values patterns in REH data.

**Methods.** True estimates are produced by performing relational event model (REM) analysis on a fully observed REH dataset. REH datasets with missing values are simulated from the full dataset by introducing missingness according to a variety of MAR simulation settings. These datasets are then analyzed using MI and CCA. The estimates produced separately by the two methods are compared to the true estimates, assessing whether they produced valid inferences.

**Results.** MI produced unbiased estimates in most simulations, but in higher proportions of missingness some of the estimates were biased. Variance of the estimates was underestimated, resulting in low coverage rates and false significant p-values. CCA produced biased estimates in all simulations.

**Conclusion.** The study provided further evidence that MI can be an effective solution to the missing data problem in REH data. Valid inference was achieved in most MAR simulations, but did show reduced effectiveness for certain analyzed statistics and in higher proportions of missingness. Future simulation research should focus on investigating their effects on MI results further.

# 1. Introduction

## 1.1 Social Networks

One of the most prevalent types of data in the modern world is network data. A network is a collection of nodes that are connected to each other by edges. Nodes represent entities and edges linking them together represent interactions or connections[18]. Networks can be directed or undirected. A directed network uses arrows as edges to show the direction of the relationship between the nodes. If the relationships between the nodes are non-directional by nature, an undirected network with lines as edges between nodes is used [18].

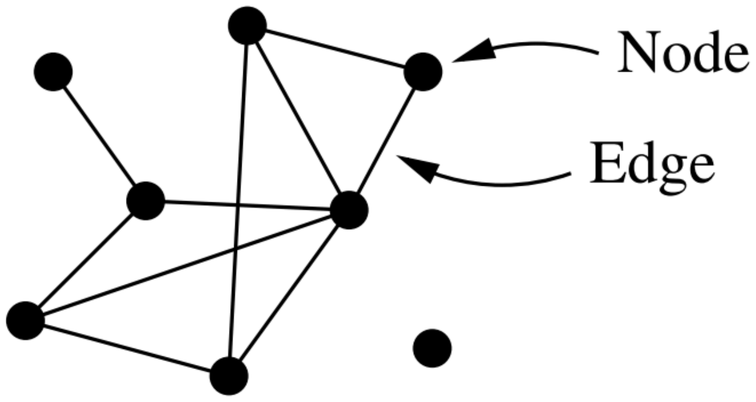


*Figure 1: Example of a small undirected network [18, p1].*

Studying these networks is becoming increasingly relevant since they crop up in almost every branch of science, especially in the physical, biological and social sciences [18]. This research focuses on social networks due to its increasing availability in the age of internet and social media. Social networks increasingly structure our society and grow stronger as forms of organization of human activity [24].

A social network is any network in which *actors* (nodes) represent people and *ties* (edges) represent their connection. This connection can be everything from familial bonds and friendships, to small one-time interactions such as sending a message. Furthermore, the actors can represent not just individuals but also animals, organizations and countries. This versatility allows the application of social network analysis to a large variety of real-world social structures. This helps us to better understand the inner workings of these networks and understand ourselves, the network's participants [24]. As shown by the examples in Figure 2, these networks can range from very simple, e.g. a family tree, to incredibly complex such as the one used by Huang [11]. This immense social network of 680 million email addresses and 6 billion relationships was used to model the patterns of ‘real’ online users. This was then used by Microsoft to detect spam accounts.

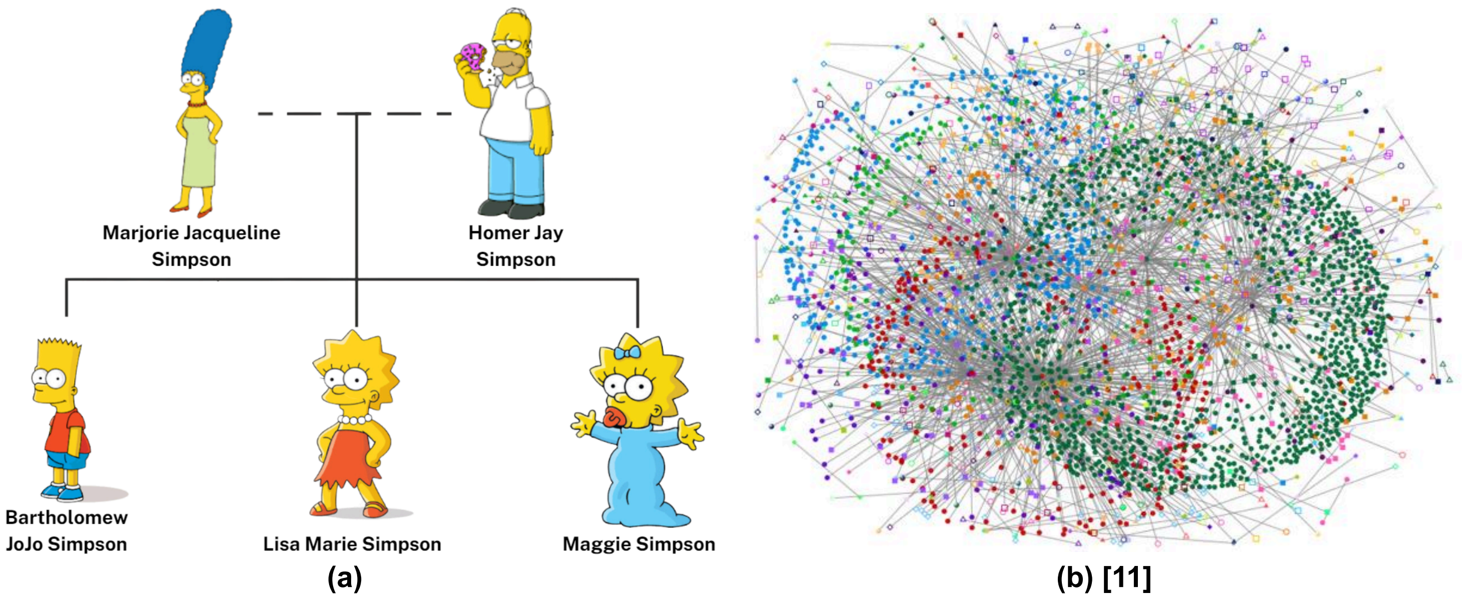


*Figure 2. Two examples of social networks: a) Family tree of a beloved cartoon family*
*b) Huge social network of email-address interactions*

Detecting spam accounts is just one example of the utility offered by social network analysis. Being able to investigate social patterns of these networks and accurately model them has helped us figure out how humans collaborate and interact with each other in 'local communities' [1], 'between scientists' [8] and 'between teachers and students' [9]. These and many other studies into social networks continue to unravel the mechanisms behind our own behavior, helping us understand ourselves.

## 1.2 Relational Event History Data

As mentioned above, a social network consists of actors (nodes) and ties (edges) connecting them. They are constructed from social network data where each row contains at least the actors involved and whether they are connected. However the social network data can contain much more information, e.g. the direction of the tie (for a directed network), the type of tie, timing and more [10]. More information in the social network data allows the creation of more sophisticated social networks, which in turn can be used to describe more complex social processes.

Social network data has become increasingly available due to new technologies in the form of Big Data. The internet has allowed the mining of huge amounts of social network data from social media, archives, email, phone records, etc [18]. Most of that data comes in the form of Relational Event History (REH) data. A relational event or action is defined as "a discrete event generated by a social actor (the 'sender') and directed toward one or more targets (the 'receivers') at a certain point in time" [3]. An example of such REH data can be seen in Table 1, each row containing a single relational event.

| Sender | Receiver | Action type | Time |
|---|---|---|---|
| John Doe | Jane Smith | Follow | 17-6-2024 14:00 |
| John Doe | Jane Smith | Like | 17-6-2024 14:05 |
| John Doe | Jane Smith | Comment | 17-6-2024 14:07 |
| Jane Smith | John Doe | Block | 17-6-2024 14:12 |

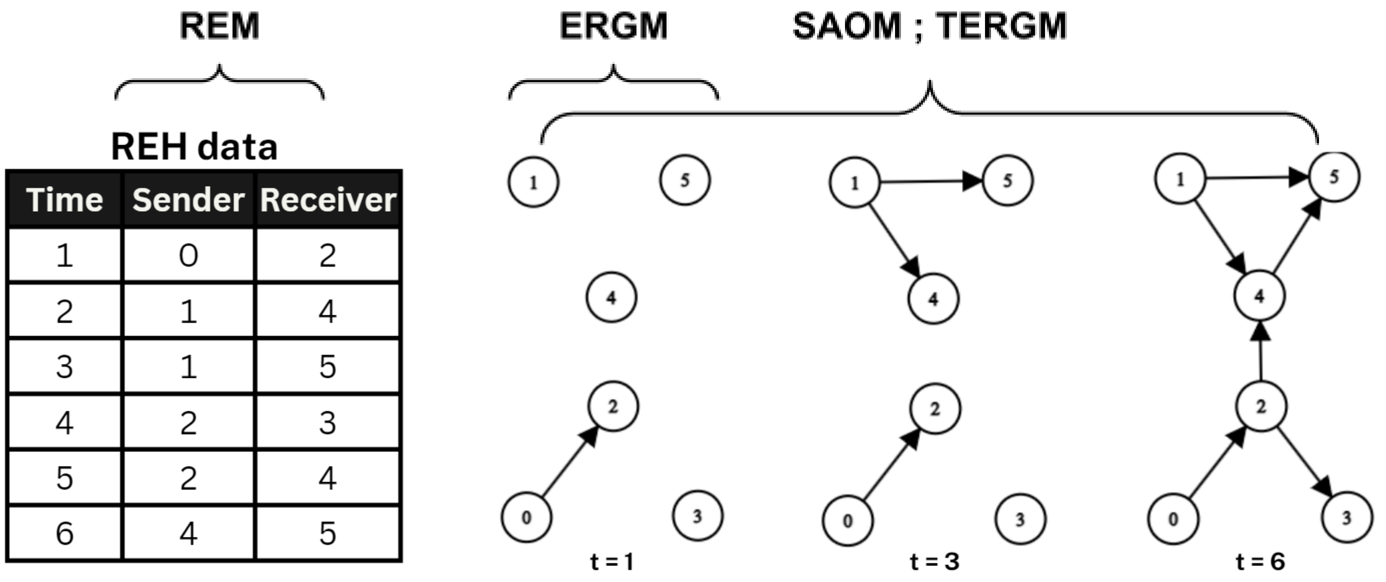
*Table 1: Example of REH data from a social media platform.*

In REH data, relational events are represented by tuples of the form:
*action* $a = (i, j, k, t)$, representing a *sender* $i$, a *receiver* $j$, an *action type* $k$ and a *time* $t$.
This was introduced by Butts et al. in 2008 [3], which introduced the Relational Event Model.

REH data contains detailed information about *what* happened, *who* were involved and *when* it happened. The timing information is especially important since it allows the creation of *dynamic social networks*. These are networks with an added function for time. This means that the network changes over time, as can be seen in Figure 3.
This importance of timing information in REH data is because of the fact that how time is recorded in social network data determines which of the four network inference models can be used [6]:

1. Exponential Random Graph Models (ERGM)
2. Stochastic Actor Oriented Models (SAOM)
3. Temporal Exponential Random Graph Models (TERGM)
4. Relational Event Models (REM)

ERGMs should be used when only 1 snapshot of the network is available, if there is no time variable in the data. E.g. in Figure 3, only t = 1 is available. Use a SAOM or TERGM if multiple snapshots are available, e.g. the networks at times 1, 3 and 6 are available without the REH data. This means that the network changes in multiple ways between each timepoint, but that the data does not contain the exact time/order each tie was created [6].



| Time | Sender | Receiver |
|---|---|---|
| 1 | 0 | 2 |
| 2 | 1 | 4 |
| 3 | 1 | 5 |
| 4 | 2 | 3 |
| 5 | 2 | 4 |
| 6 | 4 | 5 |

*Figure 3: Snapshots of a dynamic social network changing over time, based on the REH data on the left.*

If the time/order that each tie is created in a network is available, then a REM can be run. This is the case for REH data, since it is a sequence of events [5] ordered by the timestamps of the interactions. The data shows exactly what the social network looks like at each timepoint. This high-resolution time precision can be used by the REM to study how the network and the underlying social behavior changes over time. It makes it possible to model complex interaction behavior and use it to determine how interaction dynamics change over time, predict what or when the next event is, figure out how the past affects future interactions, etc. [18; 4] Many interesting and diverse phenomena are studied using models based on REH data such as food-sharing among birds [26]; cooperation in organizational networks [14] and conversational norms in online political discussions [15]. These broad applications and increasing availability make REH data, and the REMs trained on them, increasingly valuable in the social sciences.

## 1.3 Relational Event Model

Traditional statistical methods are unable to accurately model REH data due to the independence assumption being violated. The relational events are mutually dependent since the actors involved are embedded in the underlying social interaction processes. The REM developed by Butts in 2008 has been shown to be uniquely suited to take advantage of the embeddedness of the actors and use it to model how social interactions unfold over time[3].

The REM has a few advantages over other analytical methods. It has the ability to model non-stationary behavior, allowing it to capture complex dependencies over time, which is not possible in other models [3]. It allows modeling complex dependencies among events (endogenous

variables), parameterizing them to include effects of past history on behavior; for example a reciprocity effect that denotes if there is a higher chance of an event from
B to A if there was a previous event from A to B. Lastly it can model whether exogenous variables such as age, gender, etc. influence the social processes [3].
All of these advantages allow the REM to gain insights into which variables and past behavior patterns influence future interactions and how that shapes our social processes.

The REM uses the frequency and time to activation among all relational events to calculate the relative propensity for each event to occur at each timepoint [5]. It can also model how endogenous or exogenous variables influence these hazard rates. The significant variables may represent valuable insights into the inner workings of various social processes. To calculate these values for each relational event, the REM not only has to take into account exactly when it happened but also each time point it didn't happen. Due to this the REM is a powerful tool for social network analysis, but it also makes it computationally expensive and sensitive to the missing data problem.

## 1.4 Missing Data Problem

Network data will commonly have missing values, just like any other type of data, but network analysis seems to be especially vulnerable to the missing data problem [2].
Network analytics assume a complete and observed network, but missing values in network data means missing edges and an incomplete, inaccurate network. Estimating the properties of the network with just a small, completely observed part of the network is also difficult because of the local differences present in each network. The properties are non-extensive by nature and thus can not be applied to the entire network [32].

Traditionally the missing data problem was 'solved' by performing a complete-case-analysis (CCA), done by simply ignoring all rows with missing values. This is also the default method of R packages used to make REMs and analyze REH network data, such as "remstats" [17], "survival" [25] and "relevent" [4]. Unfortunately, CCA will often produce incorrect estimates due to the model being trained on an incomplete part of the real network. Missingness is usually not completely random and will therefore be more prevalent in some parts of the network. This can potentially cause shifts in the mean effects of the model, creating bias.
Imputing the missing values instead of deleting them could prevent these issues. A recent study used simulated network data with missing values to compare the effectiveness of several imputation methods [13]. Results indicated that using multiple imputation (MI) should be preferred to the usual

imputation methods since it leads to considerably less bias for both the descriptive statistics and the model parameters [13].
There is not a lot of research yet into using MI to solve the missing data problem for REH data, but a study last year using simulated missing REH data found MI to outperform CCA in specific missing data scenario's [7]. These results, although promising, do not confirm whether using MI will always produce unbiased results. Whether MI would still perform as well if multiple values are missing or the values are missing according to a Missing At Random (MAR) [28] mechanism still needs to be investigated.

This study hopes to build on that previous work and test whether MI still performs well in those more complex missing data scenarios. For that purpose the following research question must be answered: “Can a REM still produce valid inference after multiple imputation was used to impute missing values, following a MAR missingness, in all 3 columns of the REH data?”

On top of that, this study will also test the effectiveness of MI in situations with higher proportions of missingness and different missingness mechanisms. Thus an additional research question will be answered: “If the missingness proportion increases or the missingness mechanism becomes more complex in REH data, will valid inference still be possible by applying multiple imputation?”

## 2. Data

The REH data that is analyzed in this study originates from the communication records from NASA during the Apollo 13 mission. It was supposed to be a routine mission with solid procedures and safety protocols set by experience. Unfortunately it did not go as planned. After almost 57 hours an explosion occurred in an oxygen tank, “Houston, we’ve had a problem”. The spaceship was now uncontrollably drifting through space and the Apollo team was in crisis mode. From this point on the normal communication patterns changed to the safety protocol the crew was trained on [12].

This dataset is a unique case where there is a known change in communication patterns. The combination with an extensive complete record of relational events makes this dataset perfect for social network analysis and specifically REH data analysis. This is why [12] used this dataset to detect whether changepoints in communication could be detected by REM.
Part of that subset was used in [7] to assess the efficacy of MI and will also be used in this study. The sequence runs from an hour before the explosion until the end.

| Time | Sender | Receiver |
|---|---|---|
| 11849.2 | 18 | 2 |
| 11854.2 | 2 | 18 |
| 11885.2 | 18 | 2 |
| 11890.2 | 2 | 18 |
| ... | ... | ... |

*Table 2: First rows of Apollo 13 data [7]*

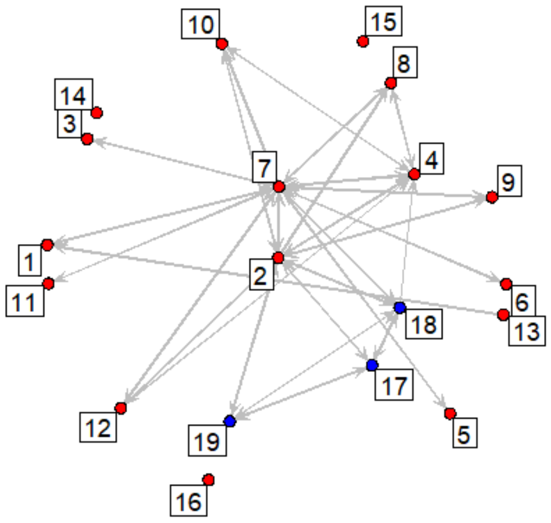


*Figure 4: Network graph of Apollo data*

The dataset is complete sequential REH data, each row is a single directed communication event representing a message sent from one crew member to another. As can be seen in table 2, each event only consists of the timepoint, sender and receiver, since there were no other action types recorded and the content of the message is not relevant. The subset contained 3882 events between 16 actors from a total of 19 crew members. As you can see in Figure 4, crew members 14 - 16 were not involved in any of the events in this subset. Actors 17 - 19 represented the 3 astronauts, the rest were at mission control on Earth. The data presents no ethical or privacy concerns since it was anonymized by swapping the actors names with numbers and is publicly available online.

## 3. Methods

The goal of using MI is not to get perfect imputations of the missing values, but to get unbiased estimates of the estimand compared to the situation with complete data. Incorrect imputations do not matter at all if the model accurately estimates the actual coefficients for the statistics that are being researched; the focus is valid inference [22].

The problem is that normally, in a model fit on a dataset with missing values, the true coefficients are not known. Luckily for us the Apollo 13 data is fully observed with 0 missing values, thus making it easy to estimate the true coefficients for our statistics with the REM.
These true coefficients can then be compared to the coefficients that the REM calculates after using MI, to assess whether MI creates datasets that produce accurate estimands.
This obviously requires the dataset to actually have missing values to impute, thus simulated datasets with missing values are generated from the original complete data. Essentially, a missing data problem is created to determine the performance of MI in simulation.

The research broadly followed these steps:

1. Run the REM model on the complete data to get the true coefficients
2. Use *amputation* to simulate missing values in the dataset 100 times
3. Use MI to impute the missing values for each of the 100 simulated datasets
4. Run the REM model on each multiple imputed dataset and pool the results
5. Evaluate MI performance by comparing the pooled coefficients to the true values

The following sections describe all these steps in more detail.
Figure 5 shows a rough schematic of steps 2-4. This is the overall setup for the simulation study and will be used in each of the tests. As you can see, the ampute function has multiple variations of arguments denoted by '{ }'. This correlates to the various tests run to evaluate how the effectiveness of MI changes with different proportions or missingness types. The missingness proportions will be 0.1, 0.2, 0.3, 0.4 and 0.5. The missingness types for "MAR" will be right-tailed ('Right') and centered ('Mid'). Meaning that larger probabilities to be missing will be assigned to candidates with high and average covariates respectively.
The standard amputation variables will be a right-tailed MAR with 0.2 proportion. Additionally a test with MCAR missingness will also be run, to show the difference between mechanisms.

CCA analysis will also be performed, its performance will act as a benchmark since it currently is the standard method on which MI should improve. The CCA results are obtained in a very similar way, but no need for step 3 and pooling the results in step 4.

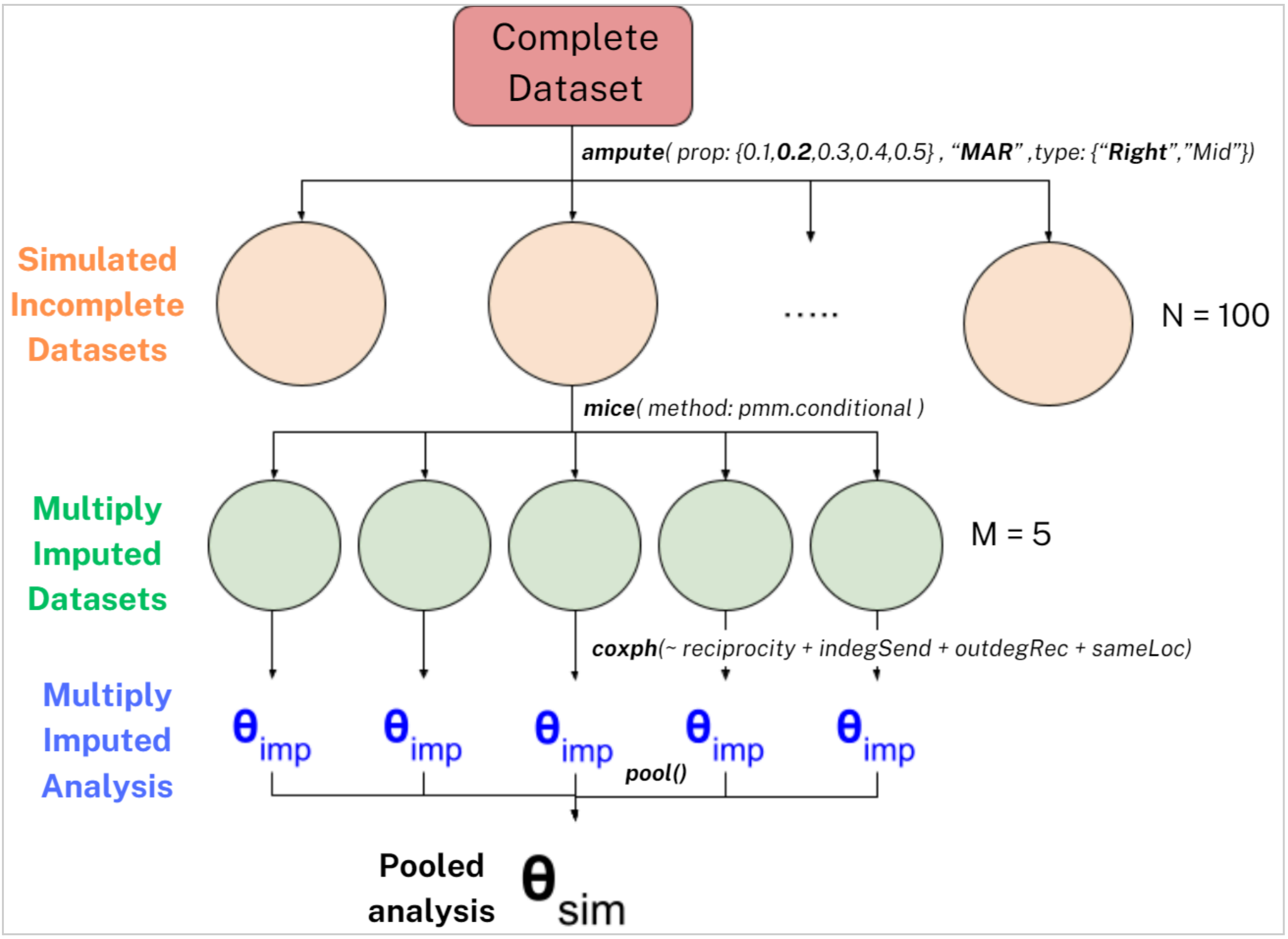

Complete Dataset
ampute( prop: {0.1,0.2,0.3,0.4,0.5} , "MAR" ,type: {"Right","Mid"})
Simulated Incomplete Datasets
.....
N = 100
mice( method: pmm.conditional )
Multiply Imputed Datasets
M = 5
coxph(~ reciprocity + indegSend + outdegRec + sameLoc)
Multiply Imputed Analysis
θimp
θimp
θimp
θimp
θimp
pool()
Pooled analysis
θsim

*Figure 5: Schema of the simulation study depicting the amputation → imputation → analysis process [30]. The complete REH data is turned into 100 simulated incomplete datasets with R function* ***ampute()*** *[27]. Multiple imputation with R function* ***mice()*** *[27] is used to create 5 MI datasets for each simulated dataset. The REM model is applied to each MI dataset with* ***coxph()*** *[25], then* ***pool()*** *[27] to get* $\boldsymbol{\theta}_{sim}$ *for each dataset. The functions show the used arguments.* ***ampute()*** *shows multiple variations with the standard values in bold.*

## 3.1 REM

As mentioned before, the REM determines the propensity for each event to occur in the future, based on the entire event history up to that point. The REM treats the dyad of sender-receiver (s,r) as the relational event. The set of all possible events, (s,r) pairings at a time point t is referred to as the risk set R(t) [17]. In the Apollo data each of the 16 actors could be a sender or receiver at each timepoint, which means that all
N x (N-1) = 16 x 15 = 240 possible events constitute the risk set R(t) for all values of t. Table 3 further down shows some of the 240 events in the risk set for the first time point. Each (s,r) pair has a rate of occurrence at time t, which is referred to as the event rate $\boldsymbol{\lambda}$(s,r,t). The REM models $\boldsymbol{\lambda}$ as a log-linear function of the chosen endogenous and exogenous variables:

(Eq 1) $$\log \lambda(s, r, t) = \sum_{p=1}^{P} \theta_p X_p(s, r, t) \ [17]$$

$X_p(s, r, t)$ is the value for the p = 1, …, P statistics for pair (s,r) at time t, the endogenous and exogenous variables chosen by the researcher to model. $\theta_p$ is the model coefficient for statistic $X_p$, determining what effect on the event rate the corresponding statistic has [17].

The researcher chooses what statistics $\boldsymbol{X}$ to include in the REM, allowing them to study whether those variables are a significant predictor for the relational events. These statistics act as either endogenous or exogenous predictors for the event rate. Endogenous predictors incorporate the history of interactions, e.g. reciprocity, in contrast to exogenous predictors which take external factors or the attributes of the actors/dyad into account, e.g. age, gender or whether both actors are at the same location.

This study will include 3 endogenous statistics:

- *Indegree sender:* A positive coefficient indicates that actors are more likely to be *sender* if they have often been receiver in the past (*indegree*) [12].
- *Outdegree receiver:* A positive coefficient indicates that actors are more likely to be *receiver* if they have often been sender in the past (*outdegree*) [12].
- *Reciprocity:* A positive coefficient indicates that actors are more likely to *reciprocate* contact. Relational events (B,A) are more likely if history included events (A,B) [12].

These statistics were chosen as they were also used in [7], since this research is a continuation of that earlier work. On top of that we also included 1 exogenous statistic:

- *Same location:* A dyad attribute denoting whether both actors are at the same location (mission control or space). A negative coefficient indicates that communication between astronauts and mission control has a higher propensity to occur, compared to astronaut-astronaut or crew-crew events [12].

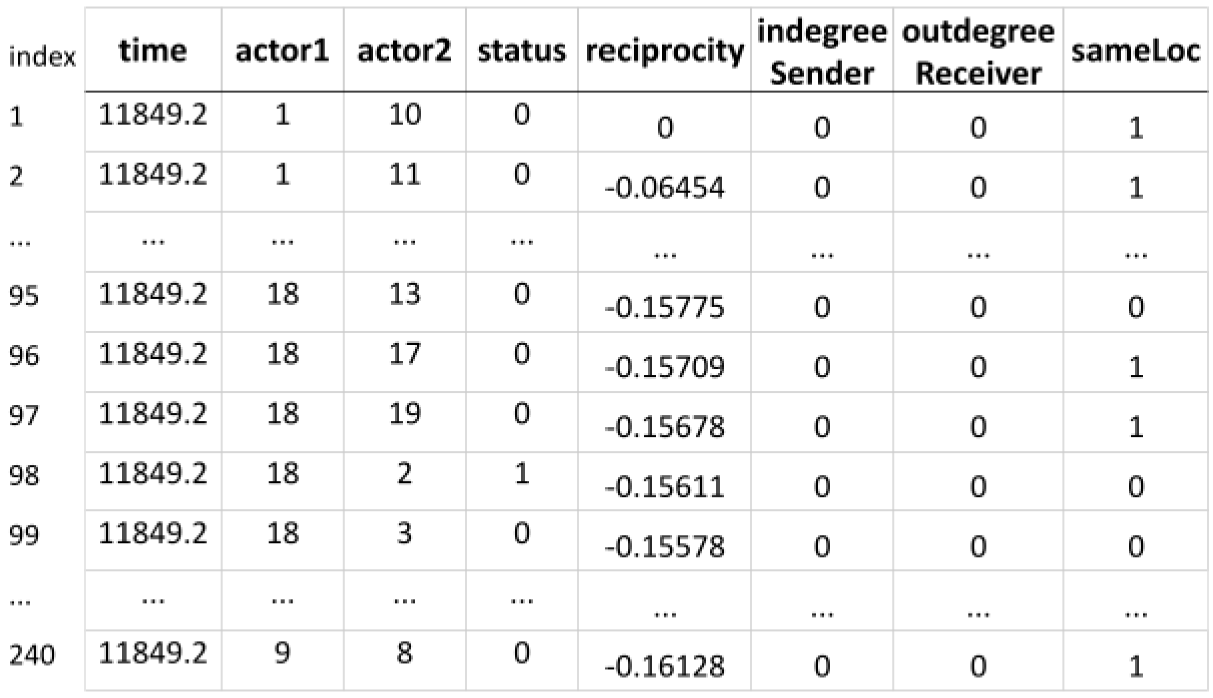
This statistic was included to test whether MI can also produce valid inference for exogenous statistics. Earlier work had only incorporated endogenous statistics [7].
With these statistics, the event rate equation(Eq 1) in the REM becomes:

(Eq 2) $$\begin{aligned}\log \lambda(s, r, t) = {} & \theta_{reciprocity} X_{reciprocity}(s, r, t) \ + \\ & \theta_{indegreeSender} X_{indegreeSender}(s, r, t) \ + \\ & \theta_{outdegreeReceiver} X_{outdegreeReceiver}(s, r, t) \ + \\ & \theta_{sameLocation} X_{sameLocation}(s, r)\end{aligned}$$

Therefore to determine the coefficient θ of each statistic, the REM requires; all statistics values for each event at each timepoint **X**(s,r,t). These values were calculated using the R package 'remstats' [17]. Table 3 shows the statistics for some of the (s,r) dyads at the first time point of the data set *t = 11849.2*.

The statistics are then used to fit a Cox proportional hazards model from the R package 'survival' [25]. The output of the fitted Cox model are **θ**, they represent whether the statistics have a significant positive or negative effect on the event rates.

This entire process was first performed on the fully observed Apollo 13 data subset to get the true coefficients of the four statistics of interest: $\boldsymbol{\theta}_{true}$.



| index | time | actor1 | actor2 | status | reciprocity | indegree Sender | outdegree Receiver | sameLoc |
|---|---|---|---|---|---|---|---|---|
| 1 | 11849.2 | 1 | 10 | 0 | 0 | 0 | 0 | 1 |
| 2 | 11849.2 | 1 | 11 | 0 | -0.06454 | 0 | 0 | 1 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 95 | 11849.2 | 18 | 13 | 0 | -0.15775 | 0 | 0 | 0 |
| 96 | 11849.2 | 18 | 17 | 0 | -0.15709 | 0 | 0 | 1 |
| 97 | 11849.2 | 18 | 19 | 0 | -0.15678 | 0 | 0 | 1 |
| 98 | 11849.2 | 18 | 2 | 1 | -0.15611 | 0 | 0 | 0 |
| 99 | 11849.2 | 18 | 3 | 0 | -0.15578 | 0 | 0 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 240 | 11849.2 | 9 | 8 | 0 | -0.16128 | 0 | 0 | 1 |

*Table 3: Example of the values of the statistics; **X**(s,r, t = 11849.2). Each row corresponds to one of the 240 possible (s,r) events.*
*The 'status' variable marks which event is at that time in the original dataset; (18,2).*
*The 'sameLoc' = 1 if both actors are at the same location; both actor id's > 16 or <= 16.*

## 3.2 Simulating Missing Values

The next step is to simulate missingness in the Apollo 13 data in such a way that the multiple imputation technique is tested well. The purpose of this is to gain insights into how MI can be used to solve the missing data problem in other research with REH data. Therefore the simulated missingness should mimic missing value patterns in 'normally' gathered data.

In real-world data gathering, missing values can occur due to a variety of different reasons; software malfunctioning, input errors, drop-out, etc. In the dataset these look the same (*NA*), but they differ in their *missingness mechanism* due to the data missing for different reasons.

There are three mechanisms that can govern the probability of being missing for data [28]:

- Missing Completely At Random (MCAR)

All data has an equal probability to be missing, because the cause of the missingness is unrelated to the data values, for example a software malfunction randomly deleting values.
The distributions and averages of the variable should not be affected much since the missingness is completely random. Thus MCAR is generally the easiest to solve, even CCA can obtain relatively valid inferences with low bias for this missingness mechanism [22].

- Missing At Random (MAR)

The probability that data is missing is dependent on the observed data. For example if participants from a poorer neighborhood have a higher chance to not answer questions about their income. Since income and ZIP code are correlated, using CCA would bias the results due to more missing values occurring in the participants from poorer neighborhoods.
In such a case MI is required, since it can use the correlation in combination with the known value (*neighborhood*) to get a good imputation for the missing value (*income*) [22].

- Missing Not At Random (MNAR)

The probability that the data is missing is dependent on unknown data. This can be due to the missing value itself, or a variable that was not included in the dataset, influencing the probability. For example if the weight variable has a higher probability to be missing in participants who are overweight, or participants who had a bad day. MNAR is the most complex mechanism to solve since the cause of the missingness is not measured. This makes it hard to obtain imputations that you are certain will not bias the estimates.

This study will simulate MAR missingness in the Apollo data. Previous research [7] has shown that MI produces valid inferences with low bias for MCAR. However, assuming MCAR missingness, while convenient, is often unrealistic. Modern missing data methods usually start from the more realistic assumption of MAR [28], thus whether MI is still effective in MAR situations merits investigation. On top of that the missingness will be induced in every combination of the *time, sender* and *receiver* variables. Both of these adjustments cause the missing data problem to be much more complex, which will be a good test for MI's effectiveness for REH data.

The amputation of data is the opposite of imputation, it is the generation of missing values in complete data [23]. Multivariate amputation can be easily done with the R function *ampute* from R package *mice* [27], which also allows for precise specification of the type of missingness that will be generated. See Figure 6 for a graphical depiction of this process. An in-depth explanation of the function and its arguments can be found at [23].

It is important to note that, just like the previous research [7], it was necessary to exclude a set of 1500 samples from the ampute function. This was needed to ensure that each actor and dyad of the original dataset would also be present in the simulated datasets since MI can not impute values not present in the data.

Arguments used for the ampute function called on the remaining 2382 samples were:

- *patterns* = all combinations of the 3 variables. These are shown on the right side of Figure 6 as the resulting missing data patterns after amputing. The original dataset is split up into candidate data blocks, 1 for each pattern. See step 1 in Figure 6 [23].

- *freq* = $\{\frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}\}$ ; same frequency for each pattern. This defines the size of the candidate block for the patterns. See step 1 in Figure 6 [23].

- *mech* = 'MAR' ; MAR mechanism for all missingness patterns [23].

- weights = 0 for the amputated variable(s), 1 for the other variables. The default for MAR. These weights are used to calculate the *sum score* based on the covariates. Here each covariate has the same effect on *sum score.* See step 2 in Figure 6 [23].

- *type* = **RIGHT** or MID ; This defines what value of the sum scores will be more likely to be missing. Right-tailed-MAR / Mid-MAR means higher probability if the weighted sum scores are high / average. Right-tailed is standard [23].

- *prop* = 0.1, **0.2,** 0.3, 0.4 or 0.5 ; Specifies the proportion of incomplete **rows**. A prop of 0.2 means that ~20% of the rows of each candidate block are made incomplete according to its corresponding missingness pattern. After merging ~20% of the rows in the dataset will be missing. See step 3 and 4 in Figure 6.

This entire process results in a dataset with missing values. This will be done 100 times for each configuration of arguments, resulting in 100 simulated datasets with missingness ready for multiple imputation.

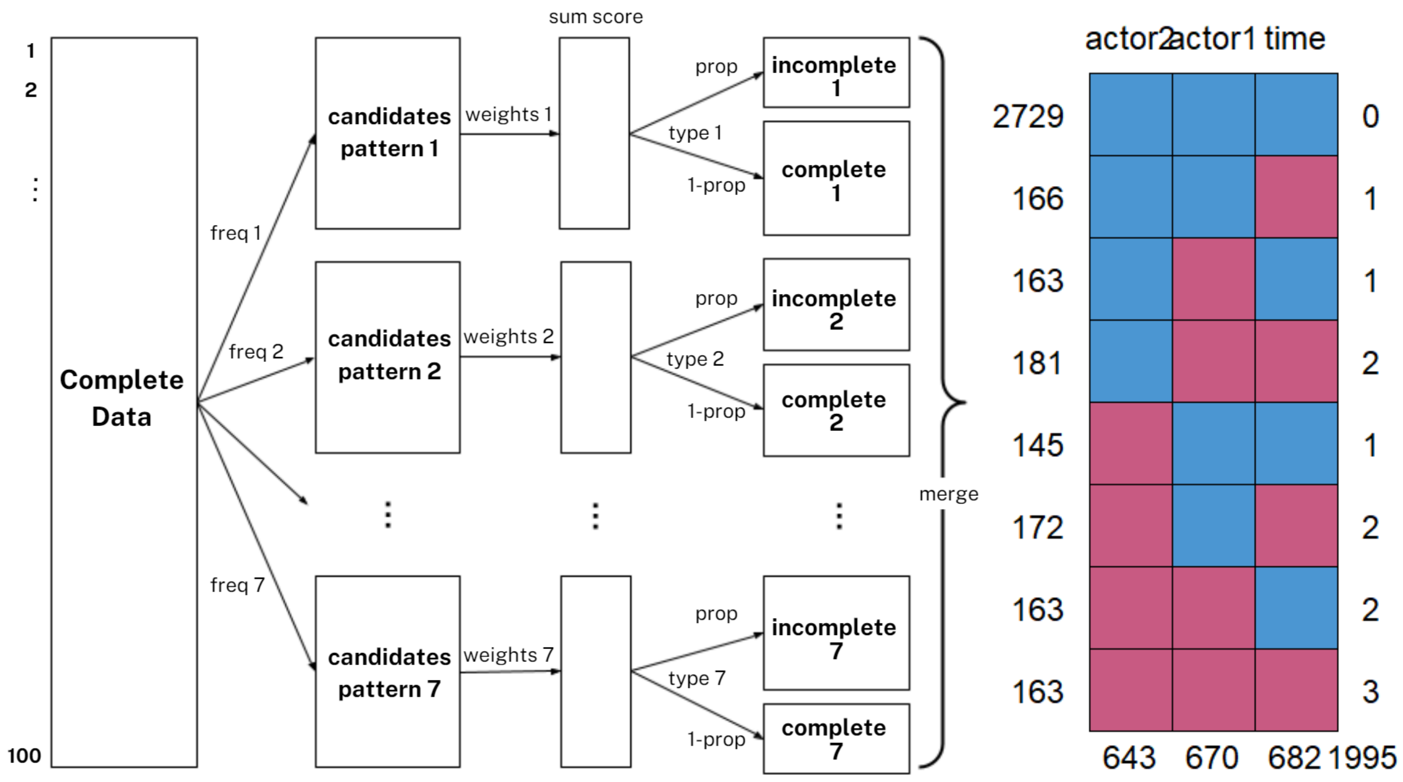


*Figure 6: Schematic showing how **ampute()** generates missing patterns (right) in a complete dataset (left) [23].*
*In our case, there are 7 missingness patterns, that all have the same frequency: $\frac{1}{7}$.*
*Each pattern has the default weights: 0 for the amputated variable(s), 1 for the other variables.*
*The proposition varies between simulations. 'prop' : {0.1, 0.2, 0.3, 0.4, 0.5}.*
*Each pattern has the same missingness 'type', usually the default: 'RIGHT', 1 simulation has 'MID'.*

## 3.3 Multiple Imputation

Multiple imputation creates $m$ complete datasets from an observed incomplete dataset [21]. All missing values are replaced by plausible values, drawn from a specific distribution for each entry that MI calculates based on the observed values. The result is that the $m$ completed datasets will only differ from each other in their imputed values. How much these imputations differ is based on the variance in the observed data, reflecting the uncertainty of the 'guess'. This prevents standard errors that are too small, which is a problem for other imputation methods [28]. Analysis, in our case the REM, is then run on the $m$ datasets. The coefficients differ due to the variance in imputations, but are pooled into one estimate using Rubin's pooling rules [21]. The variance is estimated by combining the within- and between-imputation variance and requires the estimates to be normally distributed [28]. See Figure 7 for an overview.

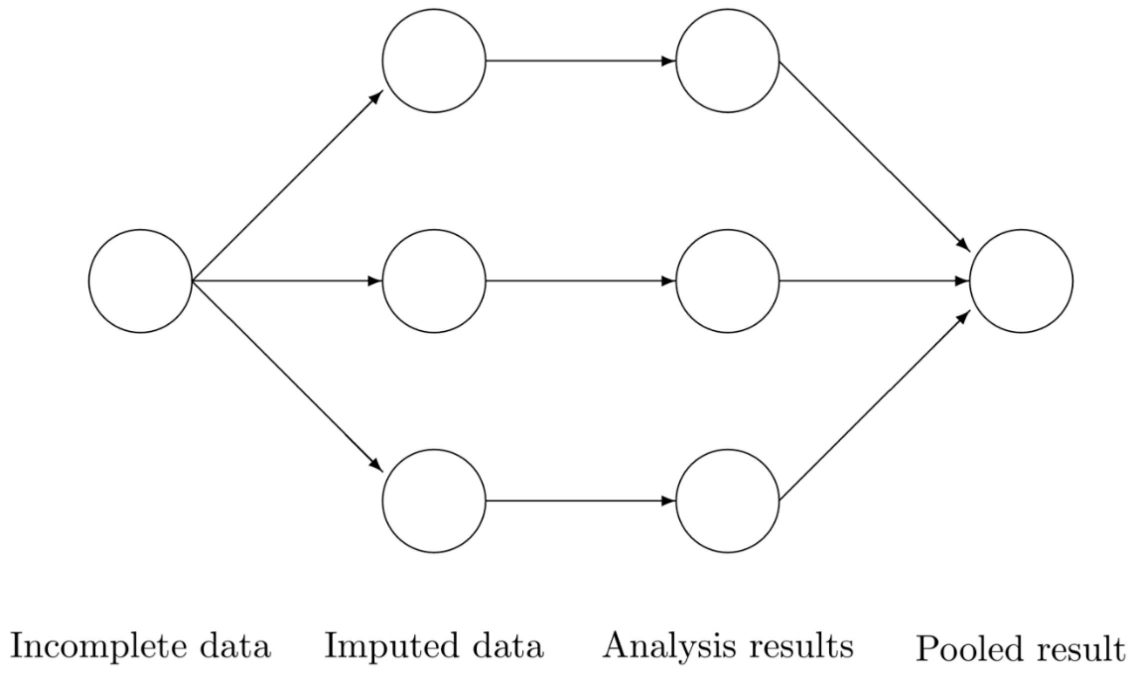


*Figure 7: Scheme of the main steps in multiple imputation, from [27]*

Before imputing the values using MI, there is an issue specific to REH data that needs to be addressed. A unique aspect of the REM is that it takes the history into account, which means that the model can not be fitted if the *time* variable of the REH data is not ordered correctly. This prerequisite can not be guaranteed if MI is allowed to impute the missing values in *time*.
Instead the simple ad-hoc method of mean interpolation was used to replace a missing time value with the average time of the preceding and succeeding events.

After imputing the *time* variable in this way, there are only missing values remaining in the *actor1* and *actor2* variables. These will be imputed using MI by running the function *mice()* from the R package *mice* [27]. MICE automatically chooses the method 'predictive mean matching' (PMM) to determine which values to impute for the actor variables. It uses the observed variables to draw potential donor values as imputation for the missing value; according to a specified imputation model. PMM creates a small set of possible donors from complete cases with covariates close to the covariates of the missing value. From this donor set it then chooses one of the values for the variable that is being imputed to replace the missing value [28]. For example, in a row where *actor1* is missing such as {12560 , NA, 18}, it would create a donor set D from the complete rows with similar values in *time* and *actor2*:

D = ( {12580, 5, 18} ; {12570, 2, 18} ; {12550, 9, 18} ; {12559.9, 18, 17} )

Then it will choose one of the donors from set D that will 'donate' the value for *actor1*. E.g. if it chooses {12580, 5, 18}, then the imputed value will be 5, resulting in {12560 , 5, 18}.

The classic PMM usually works well, but unfortunately it has the potential to create loops where actors will be sending messages to themselves. E.g. if the donor {12580, 18, 17} had been chosen, then it would have resulted in {12560 , 18, 18} which is not possible in this dataset. To prevent this, a conditional PMM will be used. This prevents donors being added to D if their value for *actor1* and *actor2* matches the missing rows' observed value for *actor2* and *actor1* respectively. In our earlier example, this would have prevented donor {12580, 18, 17} from being added to D because its proposed donor value for *actor1* matches the value for *actor2* in the row being imputed. This prevents *actor1* == *actor2*. This 'pmm.conditional' method [31] was also used in the earlier research [7].

MICE was used to generate 5 complete datasets for each simulated incomplete dataset. The number of iterations used to create the imputed datasets was 5, which is the default. More iterations can be used but that is computationally expensive and only needed if convergence is an issue. Manual checking of the trace line plots was performed to check convergence, Figure 8 shows an example. As you can see, the variable means and standard deviations of the imputations seem to be stationary and mixing well; which is a sign of convergence [19].

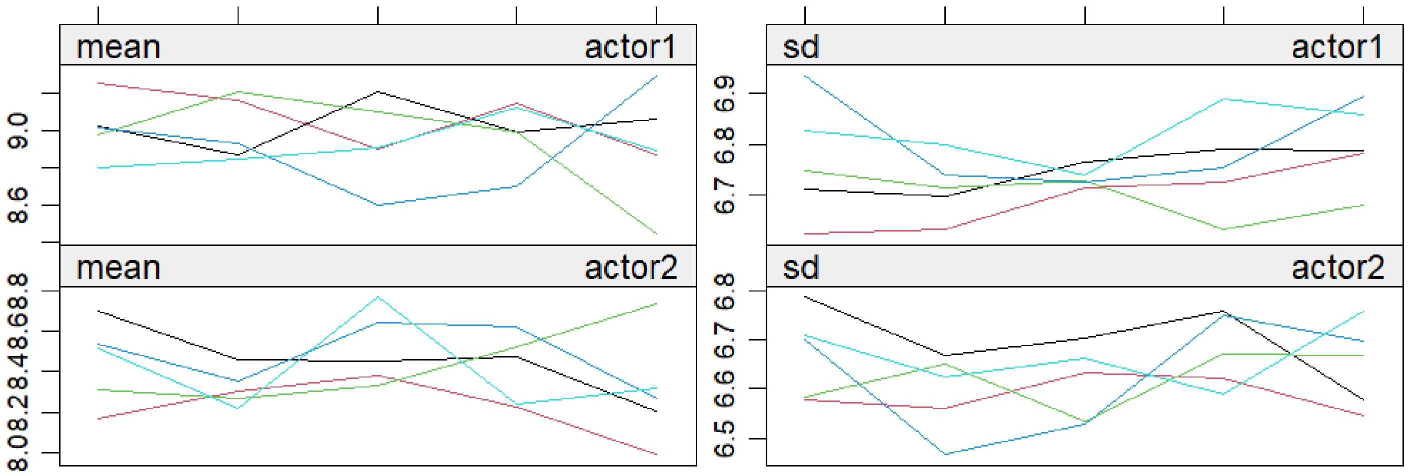


*Figure 8: Trace line plots for the multiple imputation of one of the simulated dataset*

As you can see from the overview in Figure 5, each simulated incomplete dataset now has 5 imputed datasets. Since these are now complete, their full datasets can be constructed and used in the REM to estimate $\boldsymbol{\theta}_{imp}$ for each imputation. This was done by simply repeating the actions used in step 1 to obtain $\boldsymbol{\theta}_{true}$, but on the imputed datasets instead.

The MI process is then finished by pooling the 5 different $\boldsymbol{\theta}_{imp}$ analysis results to obtain a separate $\boldsymbol{\theta}_{sim}$ for all 100 simulations. These are averaged to calculate the final result $\boldsymbol{\theta}$:

$$\boldsymbol{\theta} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{\theta}_{sim,n}$$

## 3.4 Evaluate Results

To assess how effective MI was, the coefficients **θ** obtained from the imputed data are compared to $\boldsymbol{\theta}_{true}$. These measures, backed by the literature [28], were used to evaluate the statistical validity of MI's results:

- Bias = $\boldsymbol{\theta} - \boldsymbol{\theta}_{true}$ → Should be as close to 0 as possible.
- Percent bias (PB) = $100 * |(\boldsymbol{\theta} - \boldsymbol{\theta}_{true}) / \boldsymbol{\theta}_{true}|$ → Should not extend 5%.
- Coverage rate (CR) = amount of times $\boldsymbol{\theta}_{true}$ falls within the confidence interval of **θ**
  → Should be around the nominal rate, in this case 0.95.
- Average width (AW) = $\boldsymbol{\theta}_{97.5\%} - \boldsymbol{\theta}_{2.5\%}$ ; Average width of the confidence interval of **θ**
  → Indicates efficiency and should be as small as possible.

These measures are calculated for each simulation design to show the limits of using MI for this type of data.

CR and AW are based on the variance calculation in Rubin's rules for pooling [21]:

$$T = \bar{U} + B + \frac{B}{m}$$

T denotes the total variance, B is the between imputation, m is the number of imputations and $\bar{U}$ is the sampling variance. However in this study, the variance calculation will be done as follows: $T = B + \frac{B}{m}$. The sampling variance will not be included. This is necessary because 1500 samples had to be excluded from the amputation process as described above. This artificially lessened the sampling variation as the amputation process created a constant block of samples present in every simulation. Thus excluding the sampling variance from the variance calculation is necessary to not overestimate the variance. Earlier research [30] has shown that this can be valid if the sample happens to be the population, as is the case in this study.

# 4. Results

## 4.1 True Results

Table 4 shows the results obtained by running the REM on the fully observed Apollo 13 data. These consist of the true coefficients; $\boldsymbol{\theta}_{\mathbf{true}}$, the standard errors and the p-value for each of the 4 statistics included in the REM.

| Statistic | $\theta_{true}$ | SE | P.value |
|---|---|---|---|
| Reciprocity | 0.0233 | 0.0186 | 0.209 |
| Indegree Sender | $4.3*10^{-4}$ | $7.4*10^{-5}$ | <0.001 |
| Outdegree Receiver | $-9.0*10^{-5}$ | $7.4*10^{-5}$ | 0.225 |
| Same Location | -0.8629 | 0.0322 | <0.001 |

*Table 4: REM results for the original fully observed dataset.*

Eq 2 can be filled in with these coefficients:

(Eq 2)
$$\begin{aligned} log\,\lambda(s,r,t) \;=\; & 0.0233\,X_{reciprocity}(s,r,t) \;+ \\ & 4.3\,*\,10^{-4}\,X_{indegreeSender}(s,r,t) \;+ \\ & -\,9.0\,*\,10^{-5}\,X_{outdegreeReceiver}(s,r,t) \;+ \\ & -\,0.8629\,X_{sameLocation}(s,r) \end{aligned}$$

However, the effects of ‘Reciprocity’ and ‘Outdegree Receiver’ are not significant due to their relatively high standard errors. This means that, even though the coefficient would suggest a higher chance of reciprocating interactions, the evidence can not support this conclusion. Neither is there enough evidence that actors with a high outdegree have a lower propensity of being a receiver. Thus it can’t be concluded that actors who sent a lot of messages will receive less messages in the future. These statistics can be left out of the model and Eq 2.

The results do show a statistically significant positive and negative effect for the ‘Indegree Sender’ and ‘Same Location’ statistics respectively. In practical terms, this means that an actor who received a lot of messages in the past has a higher chance of initiating contact and a lower rate of communication between pairs who are both in space or both on earth.
‘Same Location’ seems to have a much larger effect on hazard rates than ‘Indegree Sender’. With possible values of {0,1} for $X_{\text{sameLocation}}$ it will either have an effect of $-\,0.8629$ or $0$. Compared to a max value of 1162 for $X_{\text{indegreeSender}}$: $4.3\,*\,10^{-4}\,*\,1162\,=\,0.5000$ ; a mean of 121.3, resulting in ‘IndegreeSender’ average effect: $4.3\,*\,10^{-4}\,*\,121.3\,=\,0.0522$

## 4.2 Standard Simulation Results

Table 5 and 6 show the results of the REM when using MI datasets and CCA respectively. The coefficients, SE's and p-values were obtained by fitting the REM on the 100 simulated datasets and then averaged to get to the final results shown in tables 5 and 6: $\boldsymbol{\theta}_{\mathbf{MI}}$ and $\boldsymbol{\theta}_{\mathbf{CCA}}$.

Tables 5 and 6 also show how the estimates compare to $\boldsymbol{\theta}_{\mathbf{true}}$ in the form of the 4 evaluation measures: Bias, PB, CR and AW.

| Statistic | $\theta_{MI}$ | SE | P.value | Bias | PB | CR | AW |
|---|---|---|---|---|---|---|---|
| Reciprocity | 0.0241 | $7.3*10^{-4}$ | <0.001 | $7.7*10^{-4}$ | 3.32% | 0.9 | 0.0041 |
| Indegree Sender | $4.4*10^{-4}$ | $7.4*10^{-6}$ | <0.001 | $4.2*10^{-6}$ | 0.97% | 0.97 | $4.2*10^{-5}$ |
| Outdegree Receiver | $-8.9*10^{-5}$ | $3.7*10^{-6}$ | <0.001 | $1.3*10^{-6}$ | 1.41% | 0.95 | $2.1*10^{-5}$ |
| Same Location | -0.9030 | 0.0122 | <0.001 | 0.0401 | 4.64% | 0.31 | 0.0678 |

*Table 5: Averaged REM results for the MI datasets; 0.2 proportion MAR missingness.*

MI produced unbiased results for all 4 statistics, as is shown by the bias (close to 0) and PB (<5%). The CR is a perfect 0.95 for 'Outdegree Receiver', which means that the uncertainty of the estimate is captured perfectly. A CR of 0.97 for 'Indegree Sender' is suboptimal but acceptable since the true value still falls in the confidence interval of $\boldsymbol{\theta}_{\mathbf{MI}}$ 97% of the time. This means that the uncertainty is slightly underestimated compared to the ideal situation. A CR of 0.9 for 'Reciprocity' is slightly lower than acceptable, but still much better than the abysmal CR for 'Same Location'. Its confidence interval contains the true value only 31% of the time. The AW for each statistic is relatively small for each statistic, which is usually good but not if it means an underestimation of the variance leading to a CR that is too low. Another notable consequence of this is that the p-value is now much lower for each effect, resulting in all of them being statistically significant which is not the case in the true model. This is due to the much lower standard errors in the MI coefficients.

| Statistic | $\theta_{CCA}$ | SE | P.value | Bias | PB | CR | AW |
|---|---|---|---|---|---|---|---|
| Reciprocity | 0.0268 | 0.0205 | 0.197 | 0.0035 | 15.01% | 1 | 0.0805 |
| Indegree Sender | $6.1*10^{-4}$ | $1.0*10^{-4}$ | <0.001 | $1.7*10^{-4}$ | 40.41% | 0.82 | $3.9*10^{-4}$ |
| Outdegree Receiver | $-9.9*10^{-5}$ | $1.0*10^{-4}$ | 0.332 | $8.9*10^{-6}$ | 9.81% | 1 | $3.9*10^{-4}$ |
| Same Location | -0.9959 | 0.0363 | <0.001 | 0.1330 | 15.41% | 0 | 0.1422 |

*Table 6: Averaged REM results with CCA; 0.2 proportion MAR missingness.*

CCA results show large biases in the estimation of each coefficient, with PB values much higher than 5%. The estimate for ‘Indegree Sender’ had a bias as large as 40% of the true estimate. The CR of ‘Reciprocity’ and ‘Outdegree Receiver’ were acceptable but too high as their confidence intervals always contained the true value. This was mainly due to the much higher AW of the confidence intervals, suggesting a low efficiency. Even though the AW was also much higher than MI’s for ‘Indegree Sender’ and ‘Same Location’, CCA still had much lower CRs for these statistics due to its much higher biases. CCA’s p-values did mimic the true model, however these are not really useful if the estimates themselves are very biased.

These were the results for the simulated datasets which had a missingness proportion of 0.2 and the MAR missingness mechanism. These were chosen as the standard simulation settings to mimic the amputation in previous research [7]. Results for other simulation settings for both MI and CCA were also obtained. These included:

- Proportions: {0.1, 0.2, 0.3, 0.4, 0.5}
- Mechanisms: {MCAR, MAR.R, MAR.M} → Corresponding to Right-tailed and centered types of MAR missingness.

Tables for these results can be seen in Appendix A.1.

## 4.3 Other Simulation Results

### Varying Proportions

Figure 9 shows the percent bias (PB) of the estimates obtained by MI and CCA for each statistic for differing values of missingness proportions. It shows 4 graphs, one for each statistic, plotting the PB on the y-axis for the proportions on the x-axis. The dotted line at PB = 5% is an indicator for whether the estimates were valid and relatively unbiased. The y-axis is scaled logarithmically to be able to show the wildly different results between MI and CCA without losing the clear differences between proportions within the perspective methods.

The results show that MI vastly outperforms CCA for each proportion in each statistic. The differences are especially large for ‘indegreeSender’ and ‘outdegreeReceiver’, where MI produced unbiased estimates even with a proportion of 0.5. As seen above, each statistic is unbiased in the standard proportion 0.2, however that changes for ‘sameLoc’ and ‘reciprocity’ at prop = 0.3. It seems that these statistics do not handle more missing values well as their biases increase much more than the other 2 statistics. This trend can also be seen in Appendix A.2 for the other 3 measures: Bias, CR and AW.

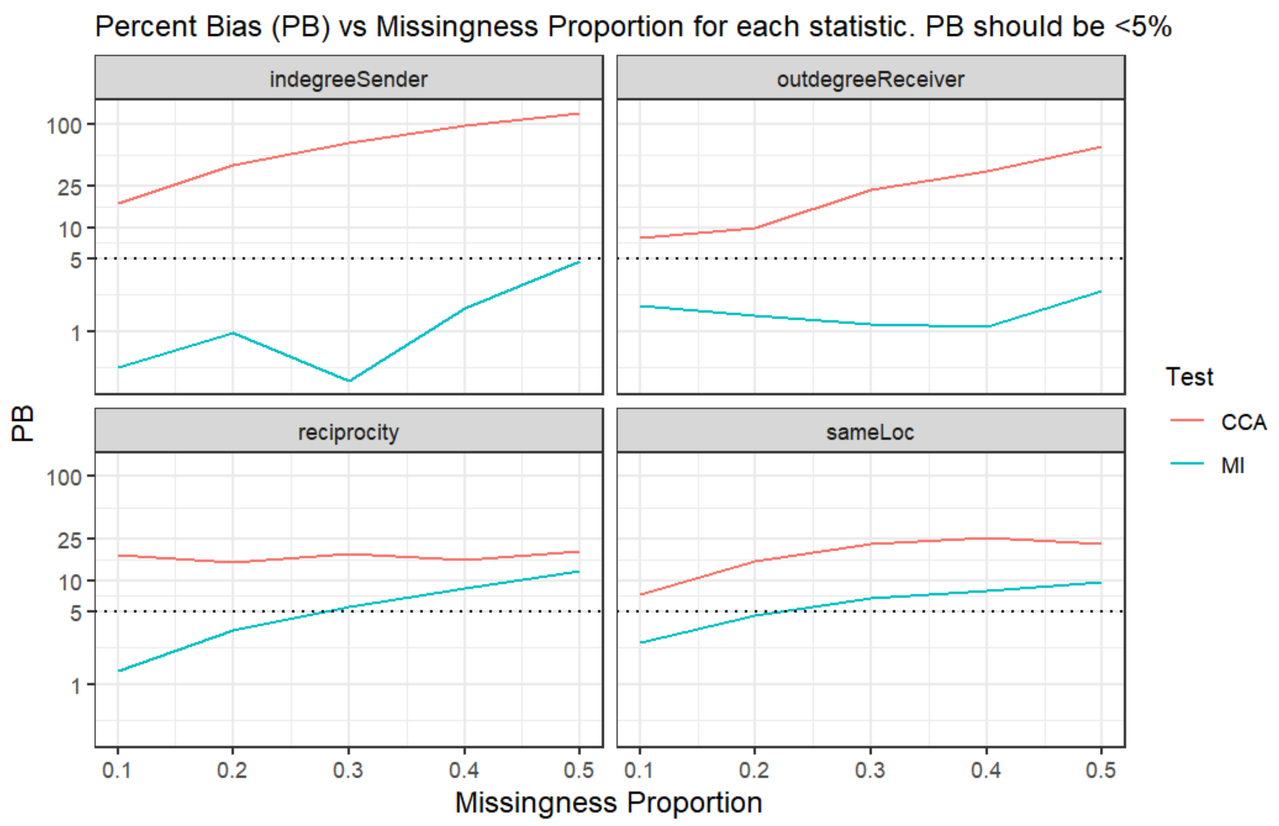


*Figure 9: A plot per statistic, showing the PB (y, log scale) vs Missing Proportion (x, 0.1-0.5)*

## Varying Mechanisms

Figure 10 shows the percent bias (PB) of the estimates obtained by MI and CCA for each statistic for differing missingness mechanisms. It shows 4 graphs, one for each statistic, plotting the PB on the y-axis for the mechanisms on the x-axis. The dotted line at PB = 5% is an indicator for whether the estimates were valid and relatively unbiased. The y-axis is again scaled logarithmically to be able to show the wildly different results between MI and CCA.

The results show that the estimates of MI for a MAR mechanism slightly outperform the other mechanisms. The PB is the lowest for MAR in 3 of the 4 statistics, and very similar to the other mechanisms in the 'sameLoc' statistic. On top of that the estimates are unbiased (<5%) for all statistics in every mechanism except for the 'reciprocity' statistic in MCAR and MAR.Mid.
MI also outperforms CCA in all but one statistic, 'sameLoc', where CCA has a much lower bias under MCAR. The results for the other evaluation measures are similar and can be seen in Appendix A.2.

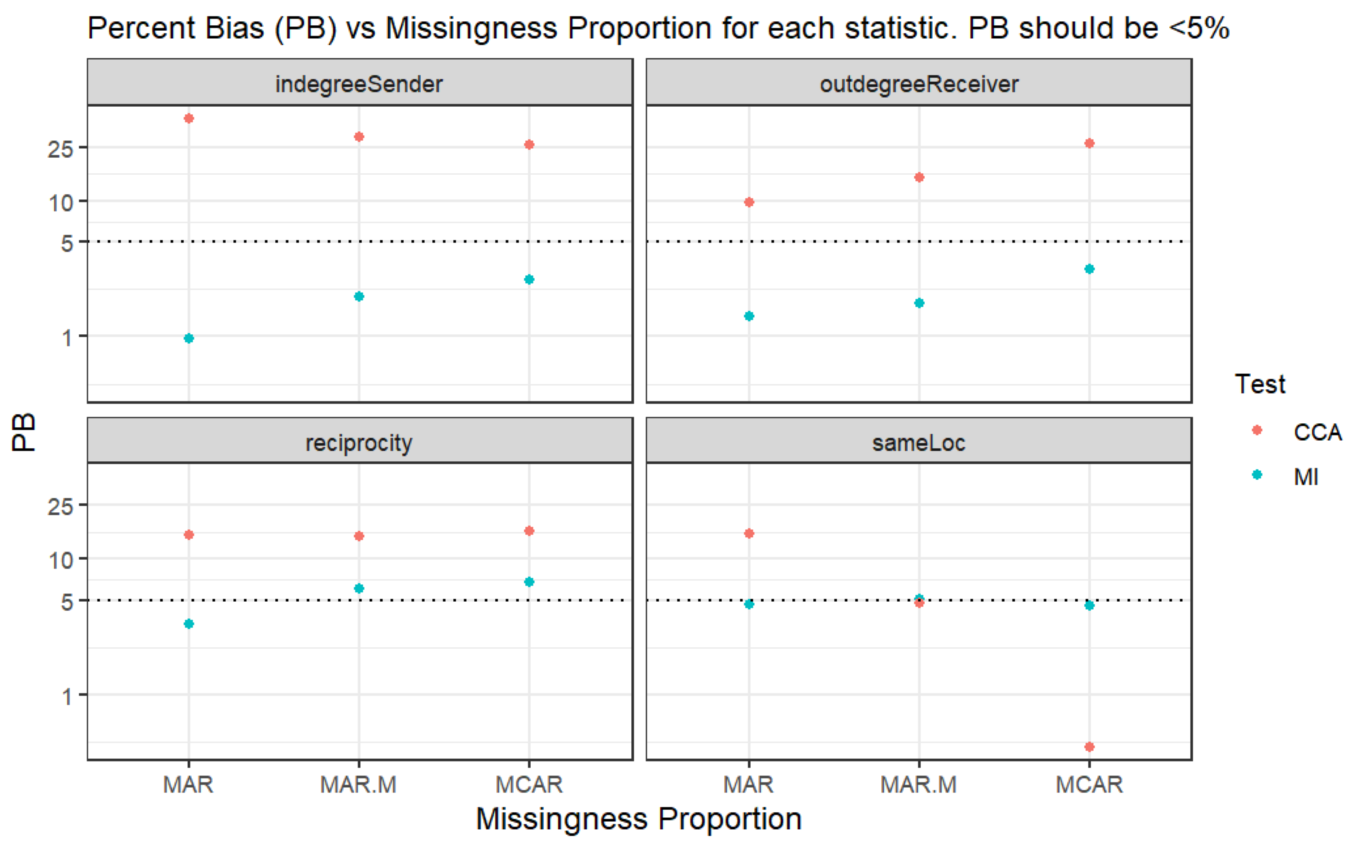


*Figure 10: A plot per statistic, showing the PB (y, log scale) vs Missingness Mechanism (“MAR”,”MAR”.M,”MCAR”)*

## 5. Discussion and Conclusion

Continuing the earlier work of [7], Multiple Imputation was tested and compared to CCA as a method for solving missing data problems in relational event history data. The missing data simulations were more complex to further test the capabilities of MI. The same measures were used to assess how well MI was able to produce a similar model to the truth [28].

MI seemed to produce valid inference for all the tested model statistics in the base missingness simulation settings, which induced missingness in all 3 columns *sender, receiver* and *time*. In those settings, missingness was induced in 20% of the rows according to a more realistic MAR mechanism. The raw and percent bias were very low, almost 0 and below 5% respectively, which shows that MI can reproduce the true model parameters relatively well.

In contrast, the results showed that using CCA produced highly biased estimates in all the statistics. This was the case even when there was only a relatively low proportion of missing rows. It seems that using CCA in a MAR situation is inadequate as a way of solving the missing data problem for REH data. This is not very surprising as earlier simulation studies [22] have already shown that CCA often produces biased estimates in a MAR situation. In those situations, with higher levels of missingness or more correlation between variables, MI can take advantage of the correlation between variables to get unbiased results where CCA can not. This was also shown in [22], at least for other data types. The results in this study now seem to confirm that this is also the case in REH data.

The results also showed that MI could not produce equally unbiased estimates in all statistics, especially for proportions of 0.3 and higher. The results of MI seemed to be a bit worse in all measures for the 'Reciprocity' and especially the 'Same Location' statistics. It could be the case that MI struggles more with statistics that incorporate both actors. Both those statistics are dependent on the pairing (s,r) as opposed to only dependent on either the sender or receiver. A missing sender or receiver that has to be imputed can alter the values for these statistics, while 'Indegree Sender' and 'Outdegree Receiver' are not impacted when missing values are induced in the receiver and sender respectively. For example, for a row {23456, NA, 4}, the value for 'Outdegree Receiver' will not vary between imputations. In the same vein, the 'Indegree Sender' value for row {23789, 5, NA} will not have any between-imputation variation, since sender is constant between imputations.
In both of these examples, the value of 'sameLocation' and 'Reciprocity' will vary between imputations. E.g. imputations of {23456, 18, 4} and {23456, 5, 4} will generate different 'sameLocation' values, 0 and 1 respectively. 'Reciprocity' is calculated based on the recency of the

opposite contact, in this case {4,18} or {4,5}. Therefore this value will also vary between the example imputations. This increase in variance potentially biases the results.
Future research should investigate whether these differences between model statistics alter the effectiveness of MI.

There were some significant limitations, which limit the conclusions which can be drawn from these results. As mentioned before, the missingness was not entirely random as 1500 samples had to be excluded from the amputation process. It was necessary, after amputation, to still have at least all actors and dyads present in the original dataset. Due to this the sampling variance in the amputation was artificially lessened. This was especially present in simulations with high levels of missingness, since a prop of 0.5 in a 3882 row dataset means 1941 rows with missing values. As 1500 rows were excluded, that left creating a 'random' 1941 missing rows from only 2382 remaining rows.
Due to this the MI pooling rules had to be changed to calculate the total variance *T* without the sampling variance *U;* $T = B + \frac{B}{m}$. Thus the variance now only took into account the between-imputation variance *B* and the number of imputations *m*. This caused the variance to be underestimated in the MI, resulting in narrow confidence intervals and lowering the AW. This in turn causes the true value to fall outside of the confidence interval more often, lowering the coverage rate. It may also be the reason for the sudden statistical significance of the previous insignificant effects in the model. This problem unfortunately could not be solved by including *U*, using the traditional Rubin's rules for pooling, as then the variance would be overestimated. Future research is needed to test whether these variance calculation issues persist when using MI on REH data, preferably with a dataset where inducing completely random missingness is possible.

The main goal of the study was to investigate if MI could produce valid estimates with low bias in more complex missing values scenarios of REH data. The results showed that MI is indeed possible to do this in a broad variety of missing value situations and thus did provide additional evidence for its effectiveness in solving the missing data problem for REH data. On top of that, even though the limitations of this study prevent the conclusion that MI will always produce valid inference, it did provide additional evidence for its superiority to CCA. In all the evaluation criteria, MI again vastly outperformed CCA in addressing missing data problems in REH. This study therefore recommends to future REH researchers: if missing data is afoot, using Multiple Imputation would be my input!

## References

1. Bernard, H. R., Johnsen, E. C., Killworth, P. D., and Robinson, S. (1989), 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.

2. 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

3. Butts, C.T. (2008). A relational event framework for social action. Sociological Methodology, 38 (1), 155–200. https://journals.sagepub.com/doi/10.1111/j.1467-9531.2008.00203.x

4. Butts, C. T., & Butts, M. C. T. (2015). Package ‘relevent’. https://cran.r-project.org/web/packages/relevent/index.html

5. Butts, Carter T and Christopher Steven Marcum (2017). ‘‘A Relational Event Approach to Modeling Behavioral Dynamics." In Andrew Pilny and Marshall Scott Poole, eds., Group Processes: Data-Driven Computational Approaches, pages 51-92. Springer International Publishing: Cham, Switzerland.

6. Cranmer, S., Desmarais, B., & Morgan, J. (2020). Inferential Network Analysis (Analytical Methods for Social Research). Cambridge: Cambridge University Press. https:// 10.1017/9781316662915.

7. Dvoriak, V. (2023). Imputing Missing values in Relational Event History data: A Framework for Social Network Research

8. 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

9. 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.

10. Hanneman, R. A., & Riddle, M. (2011). A Brief Introduction to Analyzing Social Network Data. The SAGE Handbook of Social Network Analysis, 331.

11. Huang, J., Xie, Y., Yu, F., Ke, Q., Abadi, M., Gillum, E., & Mao, Z. M. (2013, May). Socialwatch: detection of online service abuse via large-scale social graphs. In *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security* (pp. 143-148).

12. 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.

13. Krause, R. W., Huisman, M., Steglich, C., & Snijders, T. (2020). Missing data in cross-sectional networks–An extensive comparison of missing data treatment methods. Social Networks, 62, 99-112.

14. Leenders, R., Contractor, N. S., and DeChurch, L. A. (2015). Once upon a time: Understanding team dynamics as relational event networks. Organizational Psychology Review.

15. Liang, H. (2014). The organizational principles of online political discussion: A relational event stream model for analysis of web forum deliberation. Human Communication Research, 40(4):483–507.

16. Meijerink-Bosman, M., Back, M., Geukes, K. *et al* (2022). Discovering trends of social interaction behavior over time: An introduction to relational event modeling. *Behav Res* **55**, 997–1023. https://doi.org/10.3758/s13428-022-01821-8

17. 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

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

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

20. Rianne Margaretha Schouten, Peter Lugtig & Gerko Vink (2018) Generating missing values for simulation purposes: a multivariate amputation procedure, Journal of Statistical Computation and Simulation, 88:15, 2909-2930, https://doi.org/10.1080/00949655.2018.1491577

21. Rubin, D. (1987). Multiple Imputation for Nonresponse in Surveys. John Wiley and Sons, New York.

22. Schouten, R. M., & Vink, G. (2021). The dance of the mechanisms: How observed information influences the validity of missingness assumptions. Sociological Methods & Research, 50(3), 1243-1258.

23. Schouten, R. M., Lugtig, P., Brand, J. & Vink, G. (2022). Generate missing values with ampute. https://rianneschouten.github.io/mice_ampute/vignette/ampute.html

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

25. Therneau T (2023). _A Package for Survival Analysis in R_. R package version 3.5-5, https://cran.r-project.org/web/packages/survival/index.html

26. Tranmer, M., Marcum, C. S., Morton, F. B., Croft, D. P., and de Kort, S. R. (2015). Using the relational event model (rem) to investigate the temporal dynamics of animal social networks. Animal Behaviour, 101:99–105.

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

28. van Buuren, S. (2018). Flexible imputation of missing data. CRC press. https://stefvanbuuren.name/fimd/

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

30. Vink G., (2022). Strategies for simulating missingness (v1.0). https://www.gerkovink.com/simulate/

31. Vink G., (2023). mice::mice.impute.pmm.conditional() https://www.gerkovink.com/miceVignettes/

32. 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

## A.1 Results Tables

### MI - varying proportions

| Statistic | $\theta_{MI}$ | SE | P.value | Bias | PB | CR | AW |
|---|---|---|---|---|---|---|---|
| Reciprocity | 0.0236 | $5.1*10^{-4}$ | <0.001 | $3.1*10^{-4}$ | 1.35% | 0.95 | 0.0028 |
| Indegree Sender | $4.3*10^{-4}$ | $5.1*10^{-6}$ | <0.001 | $1.9*10^{-6}$ | 0.45% | 0.95 | $2.8*10^{-5}$ |
| Outdegree Receiver | $-8.9*10^{-5}$ | $2.5*10^{-6}$ | <0.001 | $1.6*10^{-6}$ | 1.76% | 0.91 | $1.4*10^{-5}$ |
| Same Location | -0.8843 | 0.0071 | <0.001 | 0.0584 | 2.48% | 0.41 | 0.0395 |

*Table A.1: Averaged REM results for the MI datasets; 0.1 proportion MAR missingness.*

| Statistic | $\theta_{MI}$ | SE | P.value | Bias | PB | CR | AW |
|---|---|---|---|---|---|---|---|
| Reciprocity | 0.0246 | $9.1*10^{-4}$ | <0.001 | 0.0013 | 5.63% | 0.84 | 0.0051 |
| Indegree Sender | $4.3*10^{-4}$ | $1.0*10^{-5}$ | <0.001 | $1.4*10^{-6}$ | 0.33% | 0.97 | $5.8*10^{-5}$ |
| Outdegree Receiver | $-8.9*10^{-5}$ | $4.1*10^{-6}$ | <0.001 | $1.1*10^{-6}$ | 1.18% | 0.91 | $2.3*10^{-5}$ |
| Same Location | -0.9213 | 0.0150 | <0.001 | 0.0214 | 6.77% | 0.41 | 0.0835 |

*Table A.2: Averaged REM results for the MI datasets; 0.3 proportion MAR missingness.*

| Statistic | $\theta_{MI}$ | SE | P.value | Bias | PB | CR | AW |
|---|---|---|---|---|---|---|---|
| Reciprocity | 0.0253 | 0.0012 | <0.001 | 0.0019 | 8.36% | 0.83 | 0.0069 |
| Indegree Sender | $4.2*10^{-4}$ | $1.4*10^{-5}$ | <0.001 | $7.3*10^{-6}$ | 1.70% | 0.97 | $7.6*10^{-5}$ |
| Outdegree Receiver | $-9.1*10^{-5}$ | $4.9*10^{-6}$ | <0.001 | $1.0*10^{-6}$ | 1.12% | 0.93 | $2.7*10^{-5}$ |
| Same Location | -0.9322 | 0.0187 | <0.001 | 0.0692 | 8.02% | 0.20 | 0.1036 |

*Table A.3: Averaged REM results for the MI datasets; 0.4 proportion MAR missingness.*

| Statistic | $\theta_{MI}$ | SE | P.value | Bias | PB | CR | AW |
|---|---|---|---|---|---|---|---|
| Reciprocity | 0.0262 | 0.0014 | <0.001 | $2.9*10^{-4}$ | 12.29% | 0.66 | 0.0077 |
| Indegree Sender | $4.1*10^{-4}$ | $1.5*10^{-5}$ | <0.001 | $2.1*10^{-5}$ | 4.75% | 0.88 | $2.8*10^{-5}$ |
| Outdegree Receiver | $-9.2*10^{-5}$ | $6.2*10^{-6}$ | <0.001 | $2.2*10^{-6}$ | 2.47% | 0.91 | $3.4*10^{-5}$ |
| Same Location | -0.9448 | 0.0220 | <0.001 | 0.0082 | 9.49% | 0.24 | 0.1223 |

*Table A.4: Averaged REM results for the MI datasets; 0.5 proportion MAR missingness.*

MI - varying mechanisms

| Statistic | $\theta_{MI}$ | SE | P.value | Bias | PB | CR | AW |
|---|---|---|---|---|---|---|---|
| Reciprocity | 0.0249 | $9.0*10^{-4}$ | <0.001 | $1.6*10^{-4}$ | 6.75% | 0.77 | 0.0050 |
| Indegree Sender | $4.2*10^{-4}$ | $8.9*10^{-6}$ | <0.001 | $1.1*10^{-5}$ | 2.65% | 0.83 | $5.0*10^{-5}$ |
| Outdegree Receiver | $-9.3*10^{-5}$ | $3.7*10^{-6}$ | <0.001 | $2.8*10^{-6}$ | 3.14% | 0.89 | $2.0*10^{-5}$ |
| Same Location | -0.9018 | 0.0100 | <0.001 | 0.0389 | 4.50% | 0.23 | 0.0554 |

*Table A.5: Averaged REM results for the MI datasets; 0.2 proportion MCAR missingness.*

| Statistic | $\theta_{MI}$ | SE | P.value | Bias | PB | CR | AW |
|---|---|---|---|---|---|---|---|
| Reciprocity | 0.0247 | $9.1*10^{-4}$ | <0.001 | 0.0014 | 6.00% | 0.90 | 0.0051 |
| Indegree Sender | $4.2*10^{-4}$ | $9.8*10^{-6}$ | <0.001 | $8.5*10^{-6}$ | 1.97% | 0.93 | $5.4*10^{-5}$ |
| Outdegree Receiver | $-9.2*10^{-5}$ | $3.8*10^{-6}$ | <0.001 | $1.6*10^{-6}$ | 1.75% | 0.93 | $2.1*10^{-5}$ |
| Same Location | -0.9065 | 0.0107 | <0.001 | 0.0436 | 5.05% | 0.17 | 0.0596 |

*Table A.7: Averaged REM results for the MI datasets; 0.2 proportion MAR.M missingness.*

CCA - varying proportions

| Statistic | $\theta_{CCA}$ | SE | P.value | Bias | PB | CR | AW |
|---|---|---|---|---|---|---|---|
| Reciprocity | 0.0274 | 0.0194 | 0.162 | 0.0041 | 17.56% | 1 | 0.0759 |
| Indegree Sender | $5.1*10^{-4}$ | $8.5*10^{-5}$ | <0.001 | $7.4*10^{-5}$ | 17.22% | 1 | $3.3*10^{-4}$ |
| Outdegree Receiver | $-9.7*10^{-5}$ | $8.6*10^{-5}$ | 0.260 | $7.1*10^{-6}$ | 7.87% | 1 | $3.4*10^{-4}$ |
| Same Location | -0.9267 | 0.0340 | <0.001 | 0.0638 | 7.39% | 0.66 | 0.1333 |

*Table A.8: Averaged REM results with CCA; 0.1 proportion MAR missingness.*

| Statistic | $\theta_{CCA}$ | SE | P.value | Bias | PB | CR | AW |
|---|---|---|---|---|---|---|---|
| Reciprocity | 0.0275 | 0.0219 | 0.215 | 0.0042 | 17.81% | 1 | 0.0859 |
| Indegree Sender | $7.1*10^{-4}$ | $1.2*10^{-4}$ | <0.001 | $2.8*10^{-4}$ | 64.59% | 0.04 | $4.7*10^{-4}$ |
| Outdegree Receiver | $-1.1*10^{-4}$ | $1.2*10^{-4}$ | 0.363 | $2.1*10^{-5}$ | 23.17% | 1 | $4.7*10^{-4}$ |
| Same Location | -1.0543 | 0.0389 | <0.001 | 0.1914 | 22.18% | 0 | 0.1526 |

*Table A.9: Averaged REM results with CCA; 0.3 proportion MAR missingness.*

| Statistic | $\theta_{CCA}$ | SE | P.value | Bias | PB | CR | AW |
|---|---|---|---|---|---|---|---|
| Reciprocity | 0.0270 | 0.0236 | 0.261 | 0.0036 | 15.62% | 1 | 0.0926 |
| Indegree Sender | $8.4*10^{-4}$ | $1.5*10^{-4}$ | <0.001 | $4.1*10^{-4}$ | 95.22% | 0 | $5.9*10^{-4}$ |
| Outdegree Receiver | $-1.2*10^{-4}$ | $1.5*10^{-4}$ | 0.428 | $3.1*10^{-5}$ | 34.53% | 1 | $5.9*10^{-4}$ |
| Same Location | -1.0834 | 0.0421 | <0.001 | 0.2205 | 25.55% | 0 | 0.1650 |

*Table A.10: Averaged REM results with CCA; 0.4 proportion MAR missingness.*

| Statistic | $\theta_{CCA}$ | SE | P.value | Bias | PB | CR | AW |
|---|---|---|---|---|---|---|---|
| Reciprocity | 0.0278 | 0.0259 | 0.292 | 0.0045 | 19.12% | 1 | 0.1014 |
| Indegree Sender | $9.8*10^{-4}$ | $2.0*10^{-4}$ | <0.001 | $5.5*10^{-4}$ | 126.80% | 0.01 | $7.7*10^{-4}$ |
| Outdegree Receiver | $-1.4*10^{-4}$ | $2.0*10^{-4}$ | 0.475 | $5.4*10^{-5}$ | 59.89% | 1 | $7.7*10^{-4}$ |
| Same Location | -1.0564 | 0.0459 | <0.001 | 0.1935 | 22.42% | 0 | 0.1799 |

*Table A.11: Averaged REM results with CCA; 0.5 proportion MAR missingness.*

CCA - varying mechanisms

| Statistic | $\theta_{CCA}$ | SE | P.value | Bias | PB | CR | AW |
|---|---|---|---|---|---|---|---|
| Reciprocity | 0.0271 | 0.0206 | 0.194 | 0.0038 | 16.14% | 1 | 0.0807 |
| Indegree Sender | $5.4*10^{-4}$ | $1.0*10^{-4}$ | <0.001 | $1.1*10^{-4}$ | 26.21% | 1 | $4.0*10^{-4}$ |
| Outdegree Receiver | $-1.1*10^{-4}$ | $1.0*10^{-4}$ | 0.281 | $2.4*10^{-5}$ | 26.38% | 1 | $4.1*10^{-4}$ |
| Same Location | -0.8665 | 0.0360 | <0.001 | 0.0035 | 0.41% | 1 | 0.1410 |

*Table A.12: Averaged REM results with CCA; 0.2 proportion MCAR missingness.*

| Statistic | $\theta_{CCA}$ | SE | P.value | Bias | PB | CR | AW |
|---|---|---|---|---|---|---|---|
| Reciprocity | 0.0268 | 0.0206 | 0.200 | 0.0035 | 14.81% | 1 | 0.0806 |
| Indegree Sender | $5.6*10^{-4}$ | $1.0*10^{-4}$ | <0.001 | $1.3*10^{-4}$ | 30.01% | 0.98 | $4.0*10^{-4}$ |
| Outdegree Receiver | $-1.0*10^{-4}$ | $1.0*10^{-4}$ | 0.323 | $1.3*10^{-5}$ | 14.87% | 1 | $4.0*10^{-4}$ |
| Same Location | -0.9034 | 0.0360 | <0.001 | 0.0041 | 4.69% | 0.99 | 0.1410 |

*Table A.14: Averaged REM results with CCA; 0.2 proportion MAR.M missingness.*

## A.2 Results Graphs

### Proportions

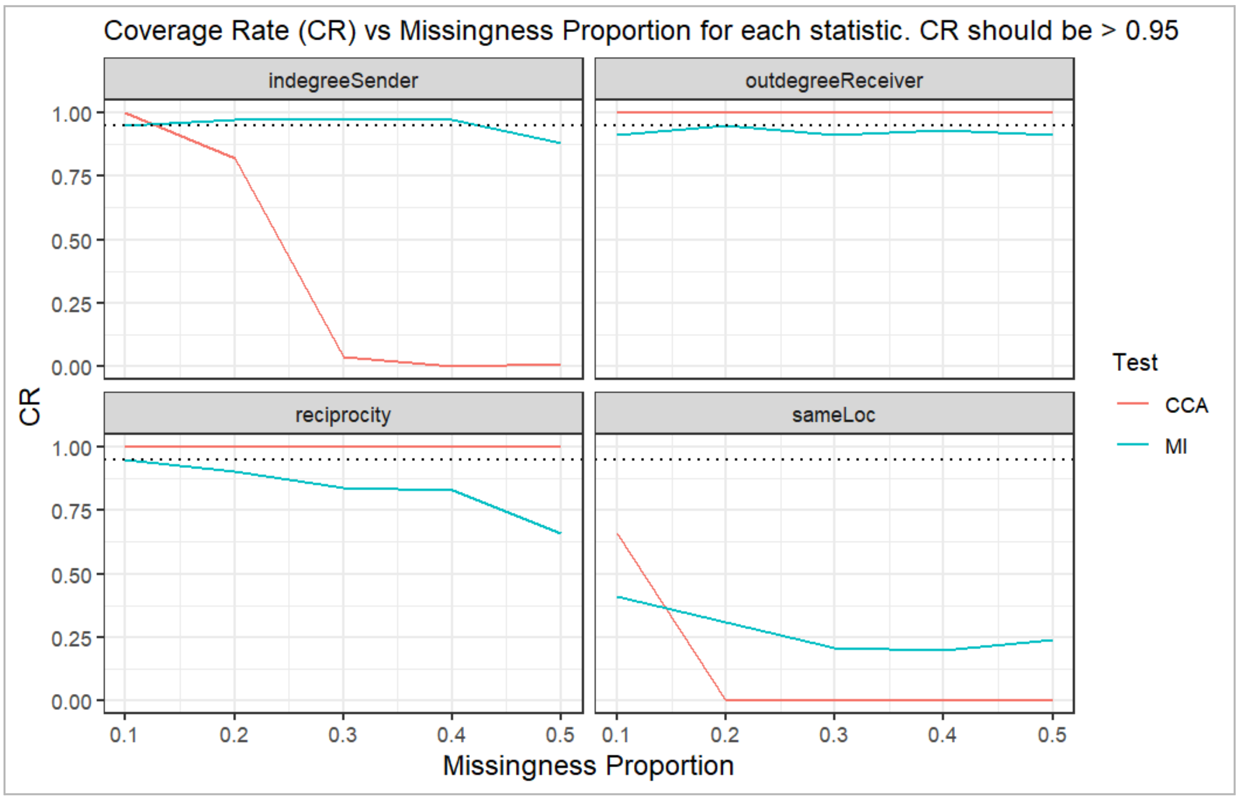


*Figure A.1: A plot per statistic, showing the CR (y, 0-1) vs Missing Proportion (x, 0.1-0.5)*

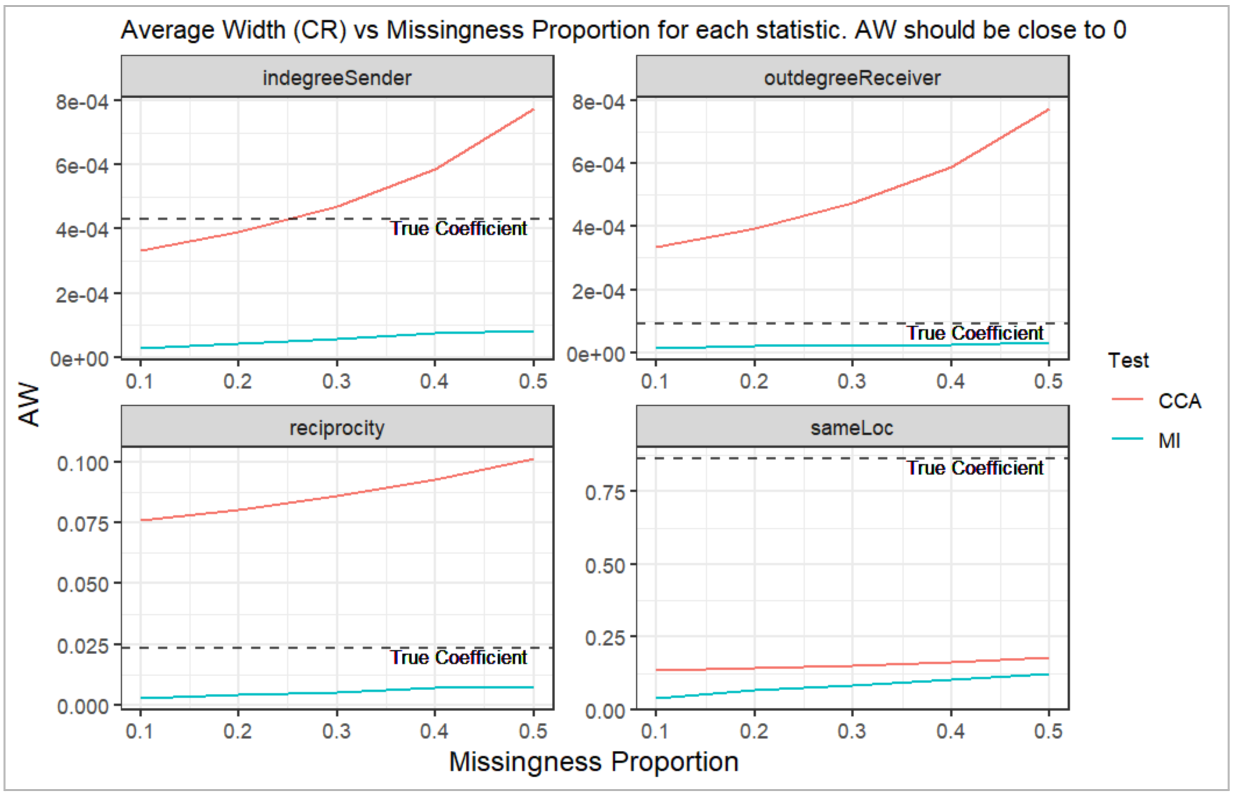


*Figure A.2: A plot per statistic, showing the AW vs Missing Proportion; dashed line:* $\boldsymbol{\theta}_{\text{true}}$

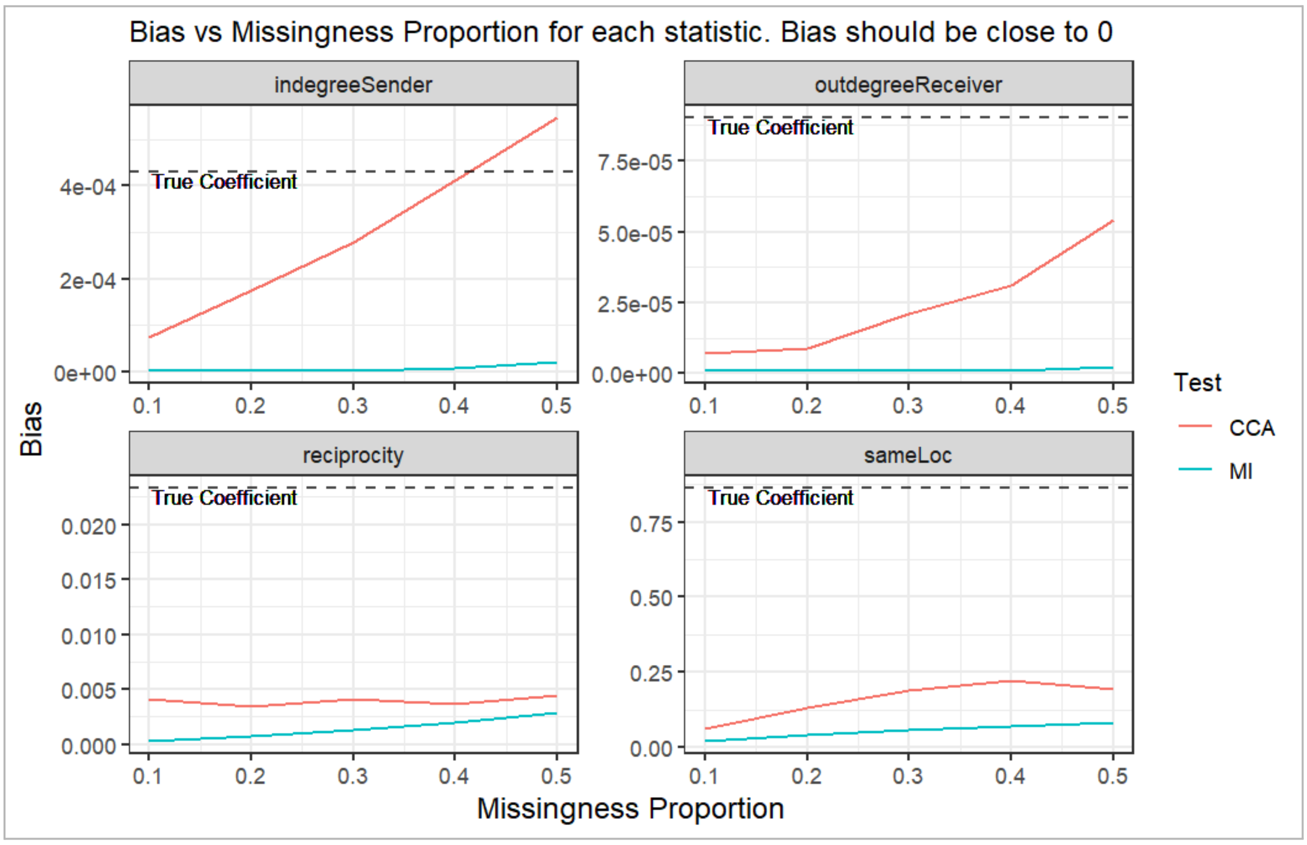


*Figure A.3: A plot per statistic, showing the bias vs Missing Proportion; dashed line:* $\boldsymbol{\theta}_{\text{true}}$

Mechanisms

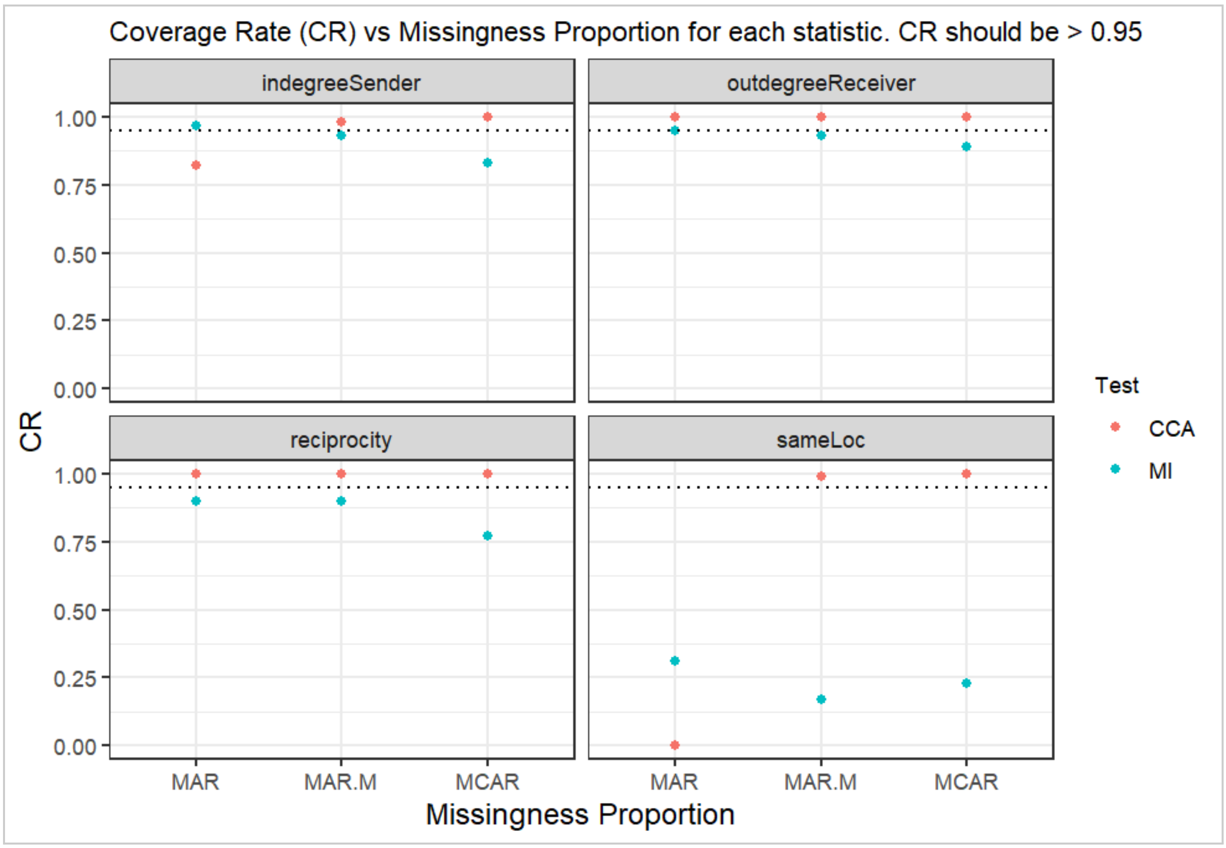


*Figure A.4: A plot per statistic, showing the CR(y, 0-1) vs Missingness Mechanism ("MAR","MAR".M,"MCAR")*

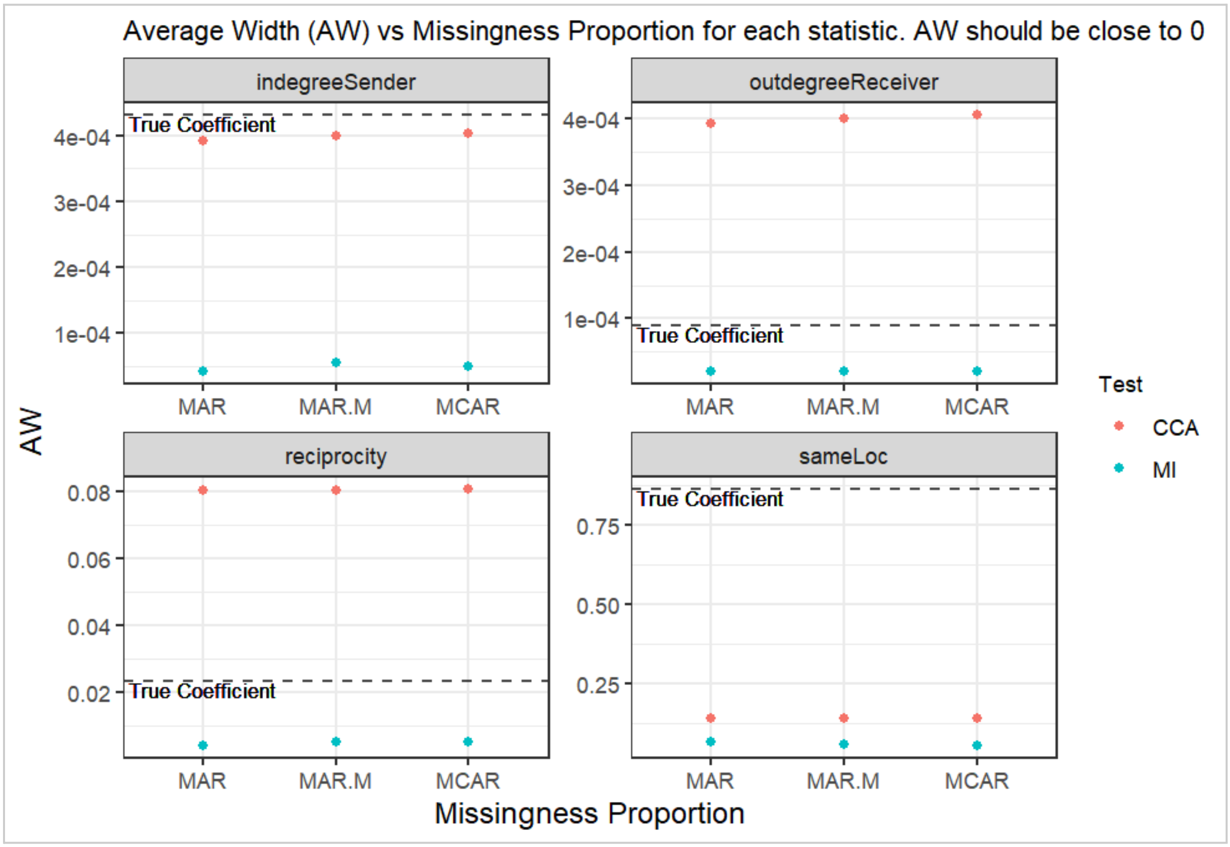


*Figure A.5: A plot per statistic, showing the AW vs Missingness Mechanism; dashed line:* $\boldsymbol{\theta}_{true}$

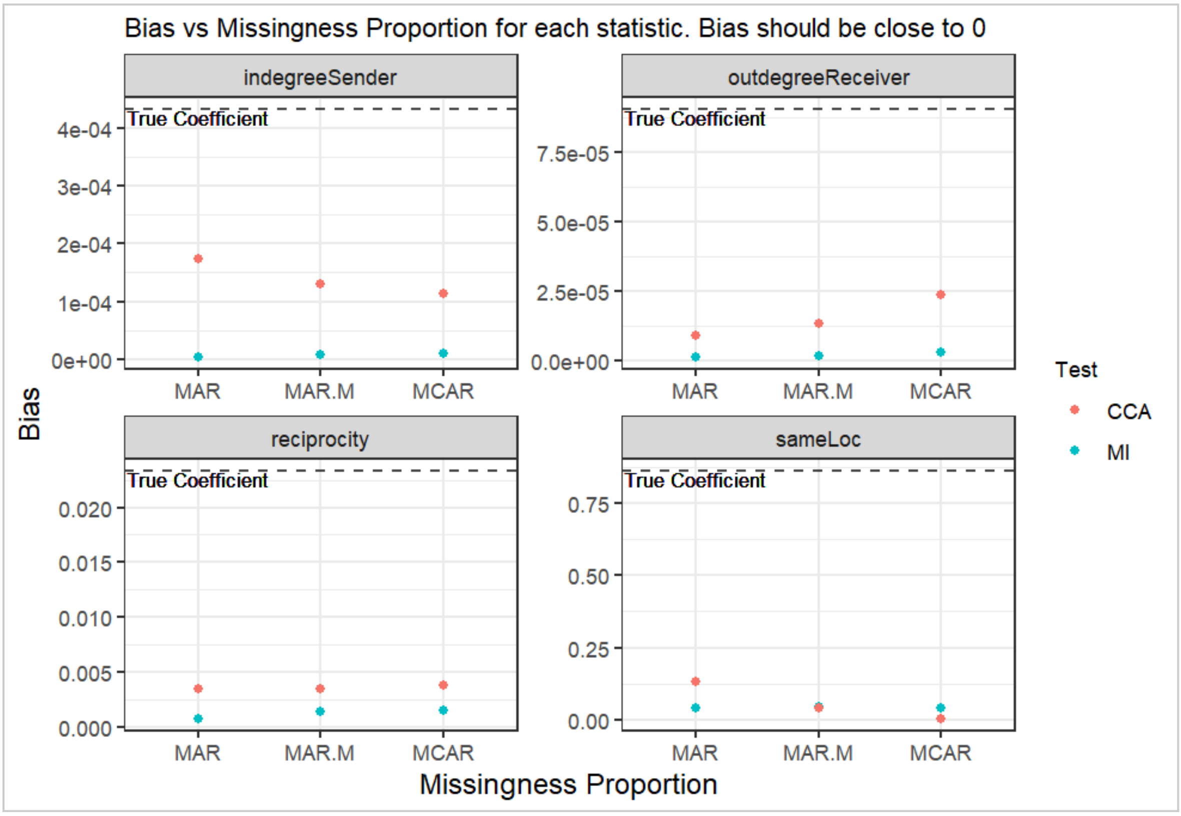


*Figure A.6: A plot per statistic, showing the bias Missingness Mechanism; dashed line:* $\boldsymbol{\theta}_{\text{true}}$

### A.3 Code

This code was made in R markdown, the sections below show the code blocks and their corresponding outputs. Parts of this code are based on the code from [7].

## Loading packages

Here we load all the packages.

```
# The needed packages, constantly updated
packages <-
c("mice","purrr","furrr","dplyr","tibble","remify","remstats","relevent","sna","
magrittr"
              ,"survival","imputeTS","survival","tibble","ggplot2")

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

lapply(packages, library,  warn.conflicts = FALSE, character.only=TRUE) # load
all the packages
```

## Load in the data

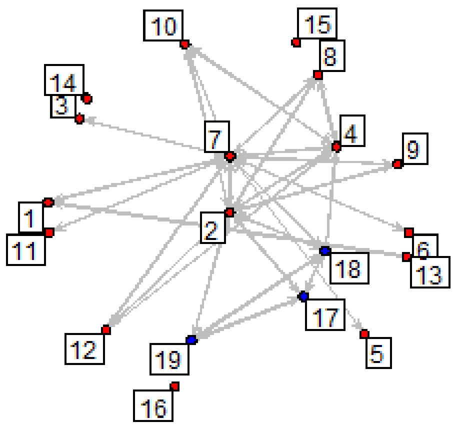
Remove the irrelevant data and rename the 'PartOfApollo_13' data

```
rm(list=ls())

# set working directory to be the current directory
setwd("~/Applied Data Science (Master)/Thesis 2024/Code")


# Load the data, rename Apollo data and remove the rest of the data
load("UUsummerschool.Rdata")
Apollo <- PartOfApollo_13  %>%
  rename(
    actor1 = sender,        # These are renamed actor1 and actor2 to match the
naming from the 'remify' function used later
    actor2 = receiver
  )

rm(Class, PartOfApollo_13, Twitter_data_rem3, WTCPoliceCalls, ClassIntercept,
   ClassIsFemale, ClassIsTeacher, WTCPoliceIsICR)

head(Apollo)

##       time actor1 actor2
## 1 11849.2     18      2
## 2 11854.2      2     18
## 3 11885.2     18      2
## 4 11890.2      2     18
## 5 12232.2      2     17
## 6 12342.2     17      2

tail(Apollo)

##          time actor1 actor2
## 3877 49995.8     19      2
## 3878 49996.8      4      7
## 3879 50000.8      2     19
## 3880 50001.8      7      4
## 3881 50012.8      7      4
## 3882 50014.8      4      7

summary(Apollo)

##       time           actor1           actor2     
##  Min.   :11849   Min.   : 1.000   Min.   : 1.000  
##  1st Qu.:25295   1st Qu.: 2.000   1st Qu.: 2.000  
##  Median :42154   Median : 7.000   Median : 7.000  
##  Mean   :36506   Mean   : 8.947   Mean   : 8.824  
##  3rd Qu.:46824   3rd Qu.:17.000   3rd Qu.:17.000  
##  Max.   :50015   Max.   :19.000   Max.   :19.000  

str(Apollo)

## 'data.frame':    3882 obs. of  3 variables:
##  $ time  : num  11849 11854 11885 11890 12232 ...
##  $ actor1: num  18 2 18 2 2 17 19 17 19 17 ...
##  $ actor2: num  2 18 2 18 17 2 17 19 17 19 ...

N <- nrow(Apollo)

# Set this as the seed to get the same results every time
set.seed(123)
```

## Visualize Network

With this full data, show the network and its connections

```
# Make sociomatrix from the Dyadic
ApolloNet <- as.sociomatrix.eventlist(Apollo,19)

# Make covariate denoting which actors are astronauts or not (last 3 are
astronauts)
IsAstronaut <- vector("logical",19)
IsAstronaut[17:19] <- TRUE
IsAstronaut

##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE

# And plot it using gplot
gplot(ApolloNet, jitter = TRUE, pad = .075,
              mode = "target", ,vertex.col=ifelse(IsAstronaut ,"blue","red"),
              displaylabels = TRUE, label.pos = 0, label.cex = .75,
              boxed.labels = TRUE, label.pad = .5,
              displayisolates = TRUE, vertex.cex=.6,
              arrowhead.cex = .75, edge.lwd = -.75, edge.col = "gray")
```



## Analyse the full dataset to get the true coefficients

Now that there is no missingness (yet), run the analysis to get the true estimates for the appropriate statistics:

1. Reciprocity (PSAB-BA)
2. Indegree sender
3. Outdegree receiver
4. Same Location; exogenous tie statistic that is 1 when both sender and receiver are at the same location

```
## This function creates a riskset for the given network with calculated
statistics according to the given formula. A riskset is a set of all events
(dyads) that have or could have occurred at each timepoint. It should have M
(dyads) * N (timepoints) rows and columns: {time,actor1,actor2,status} + 1
column for each calculated endogenous or exogenous statistic.
riskset_function <- function(network,formula){

  # Get the Relational Event History from the network
  REH <- remify(network,  model = 'tie')

  # Calculate stats from this REH for each dyad at every timepoint
  dyad_stats <- remstats(REH, tie_effects = formula)

  # Get all the dyads from the network and give them corresponding id's
  network_dyads <- attr(dyad_stats, "riskset")  %>% rename(actor1 = sender,
actor2 = receiver)

  # Add those id's to the Apollo network to denote which dyad is present
  Apollo_dyads <- merge(network,network_dyads) %>% rename(dyad = id)

  # Now we are making our full riskset, this is done by merging the network with
the dataframe containing all dyads
  # The results is a dataframe with all dyads at every timepoint, resulting in
240 (dyads) * 3882 (timepoints) = 931680 rows
  riskset <- merge(Apollo_dyads %>% select(time,dyad),network_dyads, by = NULL)
%>% arrange(time)

  # To this we add the stats calculated for each dyad at every timepoint by
remstats.
  riskset$reciprocity <- c(dyad_stats[,,1])
  riskset$indegreeSender <- c(dyad_stats[,,2])
  riskset$outdegreeReceiver <- c(dyad_stats[,,3])

  # To finish the riskset, a status variable is created. It denotes which dyad
was actually present (1 if so, 0 otherwise) at each timepoint. On top of that,
the sameLoc variable is created which is 1 if both actors are Astronauts or both
not Astronauts.
  riskset <- riskset %>% mutate(id = as.integer(id==dyad)) %>%
                                    rename(status = id) %>% select(-dyad) %>%
                                      mutate(actor1 = as.integer(actor1)) %>%
                                      mutate(actor2 = as.integer(actor2)) %>%
                                      mutate(sameLoc =
ifelse(IsAstronaut[actor1] == IsAstronaut[actor2],1,0))

    return(riskset) # The full riskset is returned
}

# Add order column to Apollo, helps sorting later since the time column needs to
be in order
Apollo$order <- 1:nrow(Apollo)

# These will be the statistics that will be studied
statistics <- ~ -1 + reciprocity(scaling = ("std")) + indegreeSender() +
outdegreeReceiver()

# Create a riskset for the true/full Apollo network using the statistics formula
created above
true_riskset <- riskset_function(Apollo,statistics)


## Fit REM model on this riskset using Cox Proportional Hazards model (coxph)
# # Old formula for coxph
# formula_23 <- Surv(time,status) ~ reciprocity + indegreeSender +
outdegreeReceiver
```

```
# New and improved formula for coxph, sameLoc added
formula_24 <- Surv(time,status) ~ reciprocity + indegreeSender +
outdegreeReceiver + sameLoc


true.fit <- coxph(formula = formula_24, data = true_riskset)


# # Fit a Cox Proportional Hazards model on this riskset with both formulas to
get the estimates for the statistics of the true network
# true.fit1 <- coxph(formula = formula_23, data = true_riskset)
# true.fit2 <- coxph(formula = formula_24, data=true_riskset)
#
# summary(true.fit1)
# summary(true.fit2)
#
# # Check model fit, using AIC and BIC values (lower is better).
# AIC(true.fit1)
# AIC(true.fit2)
# BIC(true.fit1)
# BIC(true.fit2)
# # Both the AIC and BIC are lower, so the model fit has improved by adding
sameLoc.


# Get the true coefficients for the statistics, these are compared to the
coefficients from the amputed datasets to evaluate the imputations

true_coefs <- coefficients(true.fit)
true_coefs

##       reciprocity    indegreeSender outdegreeReceiver           sameLoc
##      2.332074e-02      4.314537e-04     -9.023379e-05     -8.629124e-01
```

## Prepare data for amputing

I am puting this secret message here

```
# Check how many unique dyads and actors are in the original dataset
remify(Apollo, model = "tie") %>% dim() # 16 actors and 240 dyads

## events actors  dyads
##   3882     16    240

# Need to make sure that the amputed datasets retain those 16 actors and 240
dyads, otherwise they can't be imputed
# For that we need a random base set of (e.g. 1300) rows that we always keep
M <- 1500
base_indic <- sort(sample(1:nrow(Apollo),M))
```

```
# Test if those still retain all the actors and dyads
remify(Apollo[(base_indic),],model = "tie") %>% dim()

## events actors  dyads
##   1500     16    240

# Success! These indices still have all 16 actors and 240 dyads. These can be
used as the base set.

# Split the dataset in 2. rest_Apollo can be amputed in whatever way and then
combined with base_Apollo and arranged correctly.
base_Apollo <- Apollo[base_indic,]
rest_Apollo <- Apollo[-base_indic,]
```

### Props for the Heist Rocker

This is the function to fix the proportion issue that arose from only running ampute on a part of the dataset

```
## Problem!:
# Since the amputing is done on just a part of the dataset, the missingness
proportion will not be accurate to the prop value given to the ampute function.
This has to be fixed like this:
recalc_prop <- function (desired_prop, totalN, partN){
  rowsMissing <-  totalN * desired_prop # This is the amount of rows that should
be missing in the full dataset
  needed_prop <- rowsMissing / partN  # This is the prop missing needed in the
partial dataset to recreate the correct prop level
  return(needed_prop)
}
```

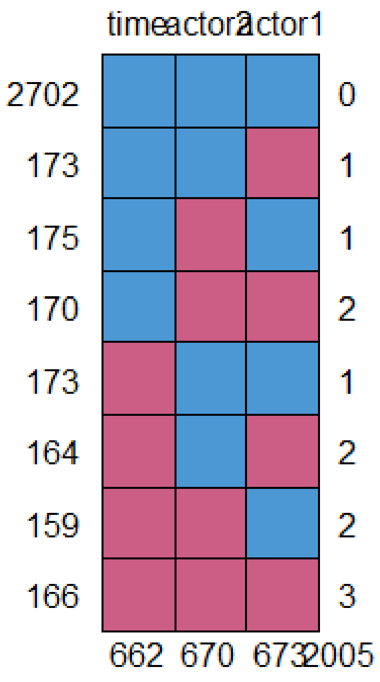
### Amputer? I hardly know her!

Here is the function used to ampute a dataset. These will be the standard arguments used in ampute() for every amputations

- We will have all possible combinations of time, actor1 and actor2 as patterns: 7 patterns
- The frequency of each pattern is the same: 1/7
- Weights will be the default

The other arguments ‘prop’, ‘mechs’and ’types’ will differ between simulations.

```
# These are the missingness patterns, the order column will never be missing of
course
myPatterns <- matrix(c(0,0,0,1,
                       0,0,1,1,
                       0,1,0,1,
                       1,0,0,1,
                       1,1,0,1,
                       1,0,1,1,
                     0,1,1,1),
                   nrow = 7,ncol=4,byrow = T)

ampute_dataset <- function(des_prop, patts, mechs, freqs = NULL, weights = NULL,
types = NULL){

  prop <- recalc_prop(des_prop,N,N-M)
  amp_rest <- ampute(rest_Apollo,prop = prop, patterns = patts, mech = mechs,
freq = freqs, weights = weights, type = types)$amp
  amputed_nw <- rbind(base_Apollo,amp_rest) %>%  arrange(order)

  return(amputed_nw)

}


# Test out the function with different frequencies for the missingness patterns
test_amp <- ampute_dataset(0.3,myPatterns,"MAR")

md.pattern(test_amp[1:3])
```



```
##      time actor2 actor1
## 2702    1      1      1    0
## 173     1      1      0    1
## 175     1      0      1    1
## 170     1      0      0    2
## 173     0      1      1    1
## 164     0      1      0    2
## 159     0      0      1    2
## 166     0      0      0    3
##       662    670    673 2005
```

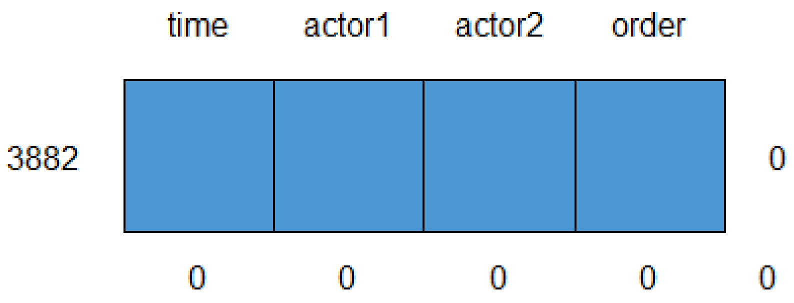
## MIce MIce Baby

The function aimpute is created, which will first make N amputed datasets using the function created above. And then it will impute those using mice.

The standard setting for mice will be: The method will be pmm_conditional, which makes sure that actor1 and actor2 can't be the same id. Imputing the missing time values will be more complicated, since the correct order needs to be maintained, thus these are just interpolated using mean interpolation the NA's in time before running MICE.

```
# These are the methods used to impute actor1 and actor2. Time will be imputed
not with MI but with mean interpolation.
myMethods <- c("","pmm.conditional","pmm.conditional","")

# This is the condition matrix, it tells pmm.conditional which variables will be
conditioned on which covariates.
cond_col <- c("","actor2","actor1","")
names(cond_col) <- colnames(Apollo)

# Here the predictor matrix is created that will be used in every multiple
imputation run.
imp0 <- mice(Apollo,printFlag = F)
predMat <- imp0$predictorMatrix
predMat[,'order'] <- 0            # Order should not help predict actor1 and
actor2

# This function will first create N amputed datasets from the Apollo dataset and
then use MI to fill in the missing values.
# This results in a list of N mids, which consist of m = 5 imputed datasets for
each simulated missing dataset.
aimpute <- function(N,des_prop, patts, mechs, freqs = NULL, weights = NULL,
types = NULL) {

   # Use future_map to run the code to ampute the dataset N times
  AMP_Apollo <- future_map(1:N, ~ {
    ampute_dataset(des_prop, patts, mechs, freqs, weights, types)
  }, .options = furrr_options(seed = 123))

  # Then use future_map to impute all those datasets 5 times with multiple
imputation
  IMP_Apollo <- future_map(AMP_Apollo[1:N], ~ {
      mutate(., time = na_interpolation(.$time)) %>% # Interpolate time before
running MICE, will be changed later
      mice(m = 5, maxit = 5,         # 5 imputations per amputed df
           method = myMethods,      # This one does not impute time using MICE,
been done above
           predictorMatrix = predMat,
           whichcolumn = cond_col,
           printFlag = F)
  }, .options = furrr_options(seed = 123))


  return(list(AMPs = AMP_Apollo, IMPs = IMP_Apollo)) # Return both the amputed
and the imputed datasets separately

}

L = 5
# Test the function with MAR and 5 simulations
MCAR_Apollo <- aimpute(L, 0.2, myPatterns, "MAR")
test_MICE_MCAR <- MCAR_Apollo$IMPs

# As you can see, the imputation has no missing values
md.pattern(complete(test_MICE_MCAR[[1]]))
```

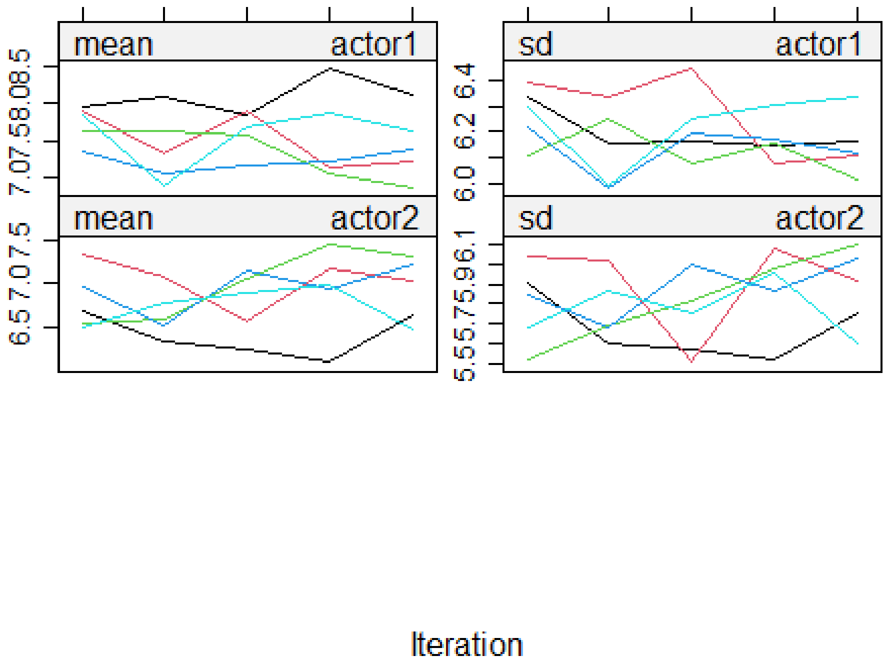
```
##  /\     /\
## {  `---'  }
## {  O   O  }
## ==>  V <==  No need for mice. This data set is completely observed.
##  \  \|/  /
##   `-----'
```



```
##      time actor1 actor2 order
## 3882    1      1      1     1 0
##         0      0      0     0 0
```

## Check convergence

```
## Here we check whether convergence occurred during the multiple imputation of
the amputed datasets
convergence <- lapply(test_MICE_MCAR, plot)

# Plot one of the convergence plots, chosen randomly. Done this multiple times
to check convergence by eye
ind_conv <- sample(1:L,1)
ind_conv

## [1] 2

convergence[ind_conv]

## [[1]]
```



```
# Seems like most converged relatively well.
```

## \Analysis on MICE datasets

Here is the function that uses the mids_list created using aimpute, then for K elements make a riskset and run the REM. These results are then pooled, the evaluation measures are added by comparing them to true_coefs. Finally the K results are averaged to get a final result, if K = N = 100 that result is for the whole mids_list (a simulation study)

```r
## Made the whole process into a function for easy reproduction
# This function takes K elements of a list of mids (multiple imputed data sets)
and fits a model on each. It then averages its results and compares to
true_coefs to assess whether valid inference is still possible after using MI.
get_MICE_results <- function(mids_list,K,true_coefs,coxph_formula) {

  start_time <- Sys.time()

  # Get the results per simulation
  sims_results <-  future_map(mids_list[1:K], ~ {    # Use future to run the
following code on the first K mids:
  complete(.x,"all") %>%  # Get all complete datasets
    future_map(~.x %>%    # Run the following code on each imputed complete
dataset:
      riskset_function(.,statistics) %$%        # Use the function to get the
complete riskset of the REH data
        coxph(formula = coxph_formula, data = .)# Fit the Cox Proportional
Hazards model on it with the given formula.
      )  %>%
    pool(custom.t = ".data$b + .data$b / .data$m") %>% # Pool the results from
every dataset in the mids together
  .$pooled %>% # extract table of pooled coefficients
        mutate(true = true_coefs, # 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,  # raw bias
               PB = 100 * abs(bias/true),  # percent bias
               AW = `97.5 %` -`2.5 %`) %>% # average width
        select(term, m, true, estimate, std.error, statistic, p.value,
               riv, `2.5 %`, `97.5 %`, cov, bias, PB, AW) %>%
        column_to_rownames("term")
  })

  # Add all and divide by length to get the average results of the entire mids
list.
  avg_results <- Reduce("+", sims_results) / K

  # Print computational time. This can take a long time to compute, this shows
how long exactly.
  end_time <- Sys.time()
  print(end_time - start_time)

  # Return the results for all sims and the averaged results in 1 named list
  return(list(all = sims_results,avg = avg_results))

}
```

## Simulating a missing data problem, then solving it with MI

In this block all the simulations of missing data are created and then imputed, aimpute is used to easily do this.

```r
setwd("~/Applied Data Science (Master)/Thesis 2024/Code/Results/Sims")

# We will do 100 simulations
K = 100

## Here are our simulation configurations that will be tested. We are first
testing different missingness proportions:
# 0.1, 0.2, 0.3, 0.4 & 0.5;  0.2 is gonna be the standard, used in the other
simulations

#MAR.1_Apollo <- aimpute(K, 0.1, myPatterns, "MAR")
MAR.1_Apollo <- readRDS(file = "C:/Users/kaspe/Documents/Applied Data Science
(Master)/Thesis 2024/Code/Results/Sims/MAR.1_Apollo.rds")

#MAR.2_Apollo <- aimpute(K, 0.2, myPatterns, "MAR")
MAR.2_Apollo <- readRDS(file = "C:/Users/kaspe/Documents/Applied Data Science
(Master)/Thesis 2024/Code/Results/Sims/MAR.2_Apollo.rds")

#MAR.3_Apollo <- aimpute(K, 0.3, myPatterns, "MAR")
MAR.3_Apollo <- readRDS(file = "C:/Users/kaspe/Documents/Applied Data Science
(Master)/Thesis 2024/Code/Results/Sims/MAR.3_Apollo.rds")

#MAR.4_Apollo <- aimpute(K, 0.4, myPatterns, "MAR")
MAR.4_Apollo <- readRDS(file = "C:/Users/kaspe/Documents/Applied Data Science
(Master)/Thesis 2024/Code/Results/Sims/MAR.4_Apollo.rds")

#MAR.5_Apollo <- aimpute(K, 0.5, myPatterns, "MAR")
MAR.5_Apollo <- readRDS(file = "C:/Users/kaspe/Documents/Applied Data Science
(Master)/Thesis 2024/Code/Results/Sims/MAR.5_Apollo.rds")


## Also test out MCAR
#MCAR.2_Apollo <- aimpute(K, 0.2, myPatterns, "MCAR")
MCAR.2_Apollo <- readRDS(file = "C:/Users/kaspe/Documents/Applied Data Science
(Master)/Thesis 2024/Code/Results/Sims/MCAR.2_Apollo.rds")

## Also test other types of missingness: MID
#MAR.2M_Apollo <- aimpute(K, 0.2, myPatterns, "MAR", types = "MID")
MAR.2M_Apollo <- readRDS(file = "C:/Users/kaspe/Documents/Applied Data Science
(Master)/Thesis 2024/Code/Results/Sims/MAR.2M_Apollo.rds")


# ## Save everything!
# setwd("C:/Users/kaspe/Documents/Applied Data Science (Master)/Thesis
2024/Code/Results/Sims/")
# saveRDS(MAR.1_Apollo, file = "C:/Users/kaspe/Documents/Applied Data Science
(Master)/Thesis 2024/Code/Results/Sims/MAR.1_Apollo.rds")
# saveRDS(MAR.2_Apollo, file = "C:/Users/kaspe/Documents/Applied Data Science
(Master)/Thesis 2024/Code/Results/Sims/MAR.2_Apollo.rds")
# saveRDS(MAR.3_Apollo, file = "C:/Users/kaspe/Documents/Applied Data Science
(Master)/Thesis 2024/Code/Results/Sims/MAR.3_Apollo.rds")
# saveRDS(MAR.4_Apollo, file = "C:/Users/kaspe/Documents/Applied Data Science
(Master)/Thesis 2024/Code/Results/Sims/MAR.4_Apollo.rds")
# saveRDS(MAR.5_Apollo, file = "C:/Users/kaspe/Documents/Applied Data Science
(Master)/Thesis 2024/Code/Results/Sims/MAR.5_Apollo.rds")
# saveRDS(MCAR.2_Apollo, file = "C:/Users/kaspe/Documents/Applied Data Science
(Master)/Thesis 2024/Code/Results/Sims/MCAR.2_Apollo.rds")
# saveRDS(MAR.2M_Apollo, file = "C:/Users/kaspe/Documents/Applied Data Science
(Master)/Thesis 2024/Code/Results/Sims/MAR.2M_Apollo.rds")
```

## In the mids of getting results

Get an average results for each simulation study made above.

```
## For all our results, we will be using the true_coefs and formula_24;

# Use the function get_MICE_results to get an average result for each simulation
made above here.

## Results from the different proportions.
#MAR.1_results <- get_MICE_results(MAR.1_Apollo$IMPs,K,true_coefs,formula_24)
MAR.1_results <- readRDS(file = "C:/Users/kaspe/Documents/Applied Data Science
(Master)/Thesis 2024/Code/Results/MAR.1_results.rds")

#MAR.2_results <- get_MICE_results(MAR.2_Apollo$IMPs,K,true_coefs,formula_24)
MAR.2_results <- readRDS(file = "C:/Users/kaspe/Documents/Applied Data Science
(Master)/Thesis 2024/Code/Results/MAR.2_results.rds")

#MAR.3_results <- get_MICE_results(MAR.3_Apollo$IMPs,K,true_coefs,formula_24)
MAR.3_results <- readRDS(file = "C:/Users/kaspe/Documents/Applied Data Science
(Master)/Thesis 2024/Code/Results/MAR.3_results.rds")

#MAR.4_results <- get_MICE_results(MAR.4_Apollo$IMPs,K,true_coefs,formula_24)
MAR.4_results <- readRDS(file = "C:/Users/kaspe/Documents/Applied Data Science
(Master)/Thesis 2024/Code/Results/MAR.4_results.rds")

#MAR.5_results <- get_MICE_results(MAR.5_Apollo$IMPs,K,true_coefs,formula_24)
MAR.5_results <- readRDS(file = "C:/Users/kaspe/Documents/Applied Data Science
(Master)/Thesis 2024/Code/Results/MAR.5_results.rds")


### Now run with different proportions
#MCAR.2_results <- get_MICE_results(MCAR.2_Apollo$IMPs,K,true_coefs,formula_24)
MCAR.2_results <- readRDS(file = "C:/Users/kaspe/Documents/Applied Data Science
(Master)/Thesis 2024/Code/Results/MCAR.2_results.rds")

#MAR.2M_results <- get_MICE_results(MAR.2M_Apollo$IMPs,K,true_coefs,formula_24)
```

```r
MAR.2M_results <- readRDS(file = "C:/Users/kaspe/Documents/Applied Data Science
(Master)/Thesis 2024/Code/Results/MAR.2M_results.rds")


## Save all the results
# saveRDS(MAR.1_results, file = "C:/Users/kaspe/Documents/Applied Data Science
(Master)/Thesis 2024/Code/Results/MAR.1_results.rds")
# saveRDS(MAR.2_results, file = "C:/Users/kaspe/Documents/Applied Data Science
(Master)/Thesis 2024/Code/Results/MAR.2_results.rds")
# saveRDS(MAR.3_results, file = "C:/Users/kaspe/Documents/Applied Data Science
(Master)/Thesis 2024/Code/Results/MAR.3_results.rds")
# saveRDS(MAR.4_results, file = "C:/Users/kaspe/Documents/Applied Data Science
(Master)/Thesis 2024/Code/Results/MAR.4_results.rds")
# saveRDS(MAR.5_results, file = "C:/Users/kaspe/Documents/Applied Data Science
(Master)/Thesis 2024/Code/Results/MAR.5_results.rds")
# saveRDS(MCAR.2_results, file = "C:/Users/kaspe/Documents/Applied Data Science
(Master)/Thesis 2024/Code/Results/MCAR.2_results.rds")
# saveRDS(MAR.2M_results, file = "C:/Users/kaspe/Documents/Applied Data Science
(Master)/Thesis 2024/Code/Results/MAR.2M_results.rds")


# ## BEEP BOOP when done, helps to know when the long computation time is over
# library(beepr)
# for(i in 1:10){
#   beep()
# }
```

## Cya Competent Analysis (CCA)

CCA, or Complete Case Analysis does not use imputation to fill in missing values. It just ignores the rows with missingness.

This block creates a new results function, this one uses the list of amputed datasets instead of the mids_list. From these datasets the rows with missing values are removed with na.omit().

After that the process is similar, create a riskset, run the REM, get a result per dataset and then average those.

```r
## For complete case analysis we also need to ampute the network 100 times, but
NOT impute them with multiple imputation.

# This function takes K elements of a list of amputed datasets and fits a model
on each with CCA. It then averages its results and compares to true_coefs to
assess whether valid inference is still possible after using MI.
get_CCA_results <- function(amp_list,K,true_coefs,coxph_formula) {

  start_time <- Sys.time()

  # Get the coefficients per simulation
  CCA.fit <- amp_list[1:K] %>%
  future_map(~.x %>%
               na.omit() %>%
               riskset_function(statistics) %$%
               coxph(formula = coxph_formula,.))

  CCA_results <- list()
  for(i in 1:K){
    cox <- CCA.fit[[i]]

    df <- data.frame(true = true_coefs,
                     estimate = cox$coefficients,
                     std.error = coef(summary(cox))[, "se(coef)"],
                     p.value = coef(summary(cox))[, "Pr(>|z|)"])  %>%
              mutate(`2.5 %`  = confint(cox)[,'2.5 %'],
                     `97.5 %`  = confint(cox)[,'97.5 %'],
                    cov = `2.5 %` < true & true < `97.5 %`,          # coverage
                    bias = abs(estimate - true),                          # raw
bias
                    PB = 100 * abs(bias/true),                       # percent
bias
                    AW = `97.5 %` -`2.5 %`)                          # average
width

    CCA_results[[i]] <- df
  }

  # Add all and divide by length to get the average results of the entire mids
list.
  avg_results <- Reduce("+", CCA_results) / K

  # Print computational time
  end_time <- Sys.time()
  print(end_time - start_time)

  # Return the results for all sims and the averaged results in 1 named list
  return(list(all = CCA.fit,avg = avg_results))

}




## Run the model on all amputed datasets with CCA

#CCA_MAR.1_results <- get_CCA_results(MAR.1_Apollo$AMPs,K,true_coefs,formula_24)
 CCA_MAR.1_results <- readRDS(file = "C:/Users/kaspe/Documents/Applied Data
Science (Master)/Thesis 2024/Code/Results/CCA_MAR.1_results.rds")

#CCA_MAR.2_results <- get_CCA_results(MAR.2_Apollo$AMPs,K,true_coefs,formula_24)
CCA_MAR.2_results <- readRDS(file = "C:/Users/kaspe/Documents/Applied Data
Science (Master)/Thesis 2024/Code/Results/CCA_MAR.2_results.rds")

#CCA_MAR.3_results <- get_CCA_results(MAR.3_Apollo$AMPs,K,true_coefs,formula_24)
CCA_MAR.3_results <- readRDS(file = "C:/Users/kaspe/Documents/Applied Data
Science (Master)/Thesis 2024/Code/Results/CCA_MAR.3_results.rds")
```

```r
#CCA_MAR.4_results <- get_CCA_results(MAR.4_Apollo$AMPs,K,true_coefs,formula_24)
CCA_MAR.4_results <- readRDS(file = "C:/Users/kaspe/Documents/Applied Data
Science (Master)/Thesis 2024/Code/Results/CCA_MAR.4_results.rds")

#CCA_MAR.5_results <- get_CCA_results(MAR.5_Apollo$AMPs,K,true_coefs,formula_24)
CCA_MAR.5_results <- readRDS(file = "C:/Users/kaspe/Documents/Applied Data
Science (Master)/Thesis 2024/Code/Results/CCA_MAR.5_results.rds")

#CCA_MCAR.2_results <-
get_CCA_results(MCAR.2_Apollo$AMPs,K,true_coefs,formula_24)
 CCA_MCAR.2_results <- readRDS(file = "C:/Users/kaspe/Documents/Applied Data
Science (Master)/Thesis 2024/Code/Results/CCA_MCAR.2_results.rds")

#CCA_MAR.2M_results <-
get_CCA_results(MAR.2M_Apollo$AMPs,K,true_coefs,formula_24)
CCA_MAR.2M_results <- readRDS(file = "C:/Users/kaspe/Documents/Applied Data
Science (Master)/Thesis 2024/Code/Results/CCA_MAR.2M_results.rds")

## save results

# saveRDS(CCA_MAR.1_results, file = "C:/Users/kaspe/Documents/Applied Data
Science (Master)/Thesis 2024/Code/Results/CCA_MAR.1_results.rds")
# saveRDS(CCA_MAR.2_results, file = "C:/Users/kaspe/Documents/Applied Data
Science (Master)/Thesis 2024/Code/Results/CCA_MAR.2_results.rds")
# saveRDS(CCA_MAR.3_results, file = "C:/Users/kaspe/Documents/Applied Data
Science (Master)/Thesis 2024/Code/Results/CCA_MAR.3_results.rds")
# saveRDS(CCA_MAR.4_results, file = "C:/Users/kaspe/Documents/Applied Data
Science (Master)/Thesis 2024/Code/Results/CCA_MAR.4_results.rds")
# saveRDS(CCA_MAR.5_results, file = "C:/Users/kaspe/Documents/Applied Data
Science (Master)/Thesis 2024/Code/Results/CCA_MAR.5_results.rds")
# saveRDS(CCA_MCAR.2_results, file = "C:/Users/kaspe/Documents/Applied Data
Science (Master)/Thesis 2024/Code/Results/CCA_MCAR.2_results.rds")
# saveRDS(CCA_MAR.2M_results, file = "C:/Users/kaspe/Documents/Applied Data
Science (Master)/Thesis 2024/Code/Results/CCA_MAR.2M_results.rds")

# ## BEEP BOOP when done, helps to know when the long computation time is over
# for(i in 1:10){
#   beep()
# }
```

### Extract results

From each results df, extract the needed variables into a smaller df. This is later used for making tables and plotting the results. To visualize the results in a way that is easier to understand.

```r
# Function to extract the results into a single table with the relevant
statistics and measures
extract_res <- function(results){
  if(length(results) == 2){ ## MICE results
```

```
    outcome <- results$avg %>% mutate(Statistic = c("Reciprocity" , "Indegree
Sender", "Outdegree Receiver", "Same Location"),
                             Estimate = estimate, SE = std.error,
                             bias = abs(estimate - true),
                             PB = 100 * abs(bias/true)) %>%
                      select(c(Statistic,Estimate,SE,p.value,bias,PB,cov,AW))
  } else { # CCA results
     outcome <- results %>% mutate(Statistic = c("Reciprocity" , "Indegree
Sender", "Outdegree Receiver", "Same Location"),
                             Estimate = estimate, SE = std.error,
                             bias = abs(estimate - true),
                             PB = 100 * abs(bias/true)) %>%
                      select(c(Statistic,Estimate,SE,p.value,bias,PB,cov,AW))
  }
  return(outcome)
}

## Extract all results

# Mice
Mar.1 <- extract_res(MAR.1_results)
Mar.2 <- extract_res(MAR.2_results)
Mar.3 <- extract_res(MAR.3_results)
Mar.4 <- extract_res(MAR.4_results)
Mar.5 <- extract_res(MAR.5_results)
Mcar <- extract_res(MCAR.2_results)
Mar.M <- extract_res(MAR.2M_results)

# CCA
CCA.1 <- extract_res(CCA_MAR.1_results)
CCA.2 <- extract_res(CCA_MAR.2_results)
CCA.3 <- extract_res(CCA_MAR.3_results)
CCA.4 <- extract_res(CCA_MAR.4_results)
CCA.5 <- extract_res(CCA_MAR.5_results)
CCA.mcar <- extract_res(CCA_MCAR.2_results)
CCA.M <- extract_res(CCA_MAR.2M_results)


## Get the true results in a nice table
true_results <- data.frame(Statistic = c("Reciprocity" , "Indegree Sender",
"Outdegree Receiver", "Same Location"),
                     Estimate = true.fit$coefficients,
                     SE = coef(summary(true.fit))[, "se(coef)"],
                     p.value = coef(summary(true.fit))[, "Pr(>|z|)"])
true_results

##                             Statistic      Estimate           SE         p.value
## reciprocity               Reciprocity  2.332074e-02 1.856480e-02  2.090504e-01
## indegreeSender        Indegree Sender  4.314537e-04 7.397946e-05  5.474252e-09
## outdegreeReceiver Outdegree Receiver -9.023379e-05 7.437120e-05  2.250192e-01
## sameLoc                 Same Location -8.629124e-01 3.216983e-02 1.711966e-158
```

## Compare results

This block makes DF’s containing all the results of the simulation studies per statistic.

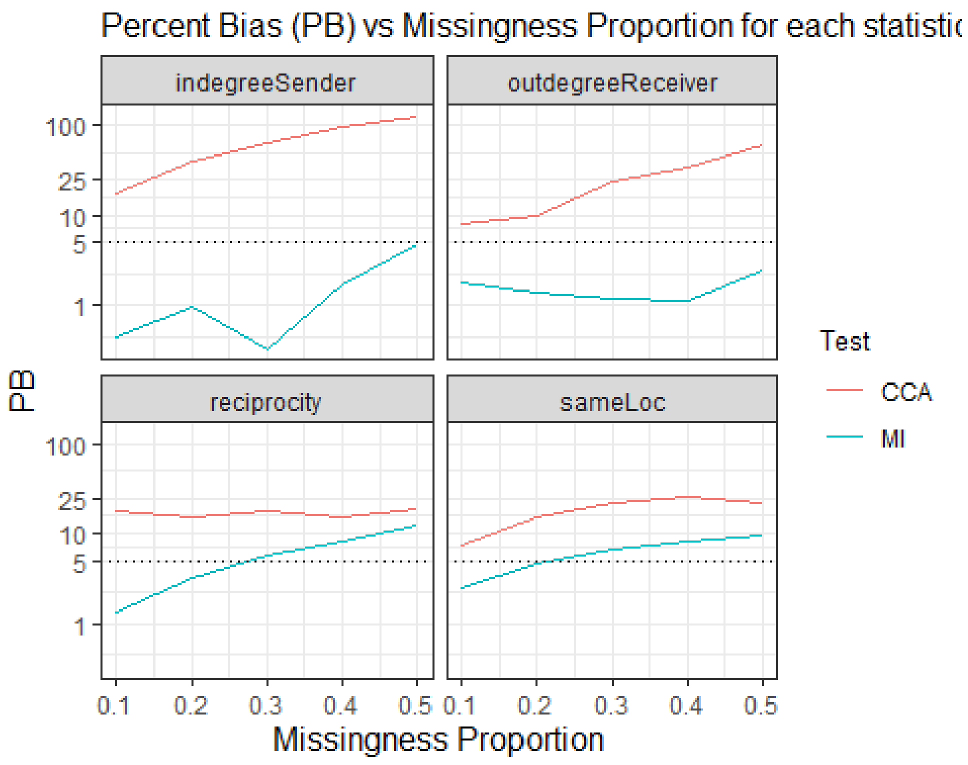
1 DF is made cntaining all the simulations with differing proportions; constant: “MAR” 1 DF containing the different mechanisms; constant: prop = 0.2

```
## Add all results in a single dataframe using the function below, each row will
be the result for 1 statistics, of 1 sim study.
# It fills the DF with different results based on whether 'mechs' is true or
not.
add_results <- function(DF,statistic,mech){

  if(mech){
    DF[nrow(DF)+1,] <- c(statistic,"MI","MCAR",0.2,c(Mcar[statistic,2:8]))
    DF[nrow(DF)+1,] <- c(statistic,"MI","MAR",0.2,c(Mar.2[statistic,2:8]))
    DF[nrow(DF)+1,] <- c(statistic,"MI","MAR.M",0.2,c(Mar.M[statistic,2:8]))

    DF[nrow(DF)+1,] <- c(statistic,"CCA","MCAR",0.2,c(CCA.mcar[statistic,2:8]))
    DF[nrow(DF)+1,] <- c(statistic,"CCA","MAR",0.2,c(CCA.2[statistic,2:8]))
    DF[nrow(DF)+1,] <- c(statistic,"CCA","MAR.M",0.2,c(CCA.M[statistic,2:8]))
  }

  else {
  DF[nrow(DF)+1,] <- c(statistic,"MI","MAR",0.1,c(Mar.1[statistic,2:8]))
  DF[nrow(DF)+1,] <- c(statistic,"MI","MAR",0.2,c(Mar.2[statistic,2:8]))
  DF[nrow(DF)+1,] <- c(statistic,"MI","MAR",0.3,c(Mar.3[statistic,2:8]))
  DF[nrow(DF)+1,] <- c(statistic,"MI","MAR",0.4,c(Mar.4[statistic,2:8]))
  DF[nrow(DF)+1,] <- c(statistic,"MI","MAR",0.5,c(Mar.5[statistic,2:8]))

  DF[nrow(DF)+1,] <- c(statistic,"CCA","MAR",0.1,c(CCA.1[statistic,2:8]))
  DF[nrow(DF)+1,] <- c(statistic,"CCA","MAR",0.2,c(CCA.2[statistic,2:8]))
  DF[nrow(DF)+1,] <- c(statistic,"CCA","MAR",0.3,c(CCA.3[statistic,2:8]))
  DF[nrow(DF)+1,] <- c(statistic,"CCA","MAR",0.4,c(CCA.4[statistic,2:8]))
  DF[nrow(DF)+1,] <- c(statistic,"CCA","MAR",0.5,c(CCA.5[statistic,2:8]))
  }

  return(DF)
}

## Initialize the results_DF's, then fill them with the corresponding results.
results_DF_prop <- results_DF_mech <- data.frame(Stat = row.names(true_results),
Test = "True", Mech = NA, Prop = 0, c(true_results[,2:4]),bias = NA, PB = NA, CR
= NA, AW = NA)

# DF for prop results
for (stat in row.names(true_results)) {
  results_DF_prop <- add_results(results_DF_prop,stat,FALSE)
}
results_DF_prop <- results_DF_prop[-c(1:4),] # remove the true_results

# DF for mechanism results
```

```
for (stat in row.names(true_results)) {
  results_DF_mech <- add_results(results_DF_mech,stat,TRUE)
}
results_DF_mech <- results_DF_mech[-c(1:4),] # remove the true_results
```

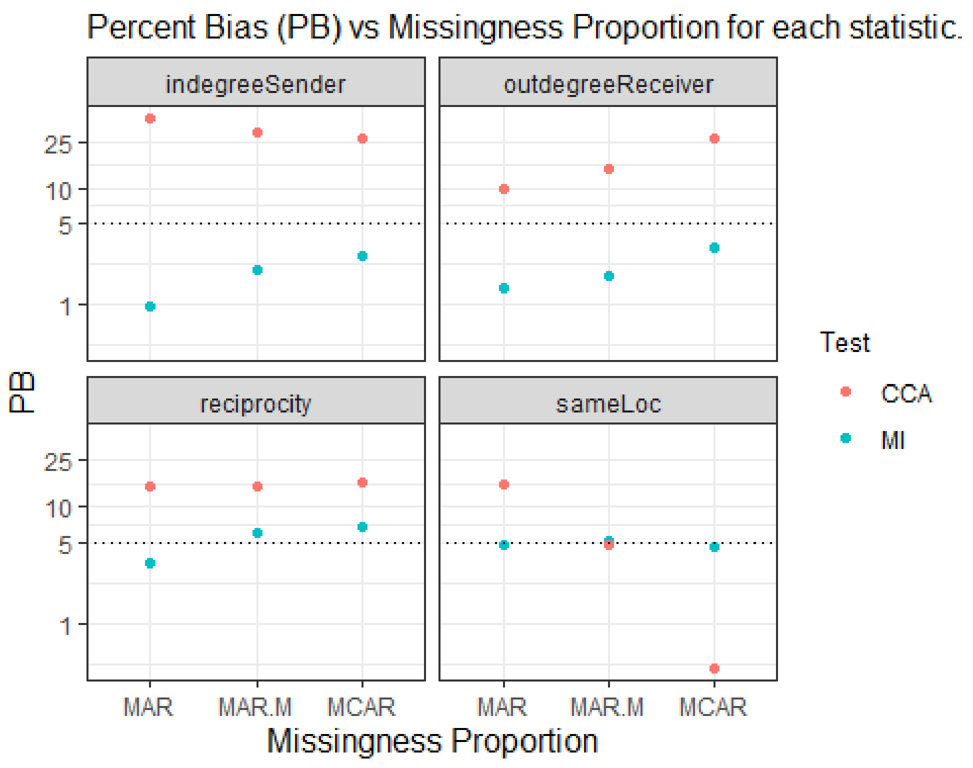
### Plot PB vs Missingness Proportions, 1 plot for each of the 4 statistics

```
# Plot PB vs Missingness Proportion for each statistic
plot_PB.Prop <- ggplot(results_DF_prop, aes(x = Prop, y = PB, colour = Test)) +
  geom_line() +
  geom_hline(aes(yintercept = 5), linetype = "dotted") +
  labs(title = "Percent Bias (PB) vs Missingness Proportion for each statistic.
PB should be <5%",y = "PB",x="Missingness Proportion") +
  facet_wrap(. ~ Stat) +
  scale_y_log10(breaks = c(0,1,5,10,25,100)) +
  theme_bw()

my_theme <- theme(title = element_text(size=10),
axis.title=element_text(size=12))

plot_PB.Prop + my_theme
```



### Plot PB vs Mechanisms, 1 plot for each of the 4 statistics

```
# Plot PB vs Mechanisms for each statistic
plot_PB.Mech <- ggplot(results_DF_mech, aes(x = Mech, y = PB, colour = Test)) +
  geom_point() +
  geom_hline(aes(yintercept = 5), linetype = "dotted") +
  labs(title = "Percent Bias (PB) vs Missingness Proportion for each statistic.
PB should be <5%",y = "PB",x="Missingness Proportion") +
  facet_wrap(. ~ Stat) +
  scale_y_log10(breaks = c(0,1,5,10,25,100)) +
  theme_bw()

plot_PB.Mech + my_theme
```


Percent Bias (PB) vs Missingness Proportion for each statistic.
indegreeSender
outdegreeReceiver
reciprocity
sameLoc
PB
MAR
MAR.M
MCAR
Missingness Proportion
Test
CCA
MI