Enhancing Relational Event History Data Analysis: Multiple Imputation for Missing Values handling in Social Network Research

Jesse J. van der Wensch (5453062)

Mahdi Shafiee Kamalabad

Gerko Vink

Master Applied Data Science, Utrecht University

July 7, 2023

Abstract

Missing data can have significant effects on reliability of results and lead to incorrect conclusions. This study examines the effectiveness of multiple imputation in relational event history data. The study compares the estimates of a relational event model of the complete data with a 100 simulations where missing data was generated using the assumption that the data is missing completely at random (MCAR). It was found that, overall, the imputation method gave accurate estimates. However, the significance of the estimations changed from being not significant to significant. This change in significance should be taken into consideration when interpreting results after imputation.

Keywords: Relational event history, REH, relational event model, REM, missing values, multiple imputation

1. Introduction

Researchers often encounter the problem of missing data, which can have a significant impact on the reliability of results and lead to incorrect conclusions (Little & Rubin, 2019). A simple approach to handling missing data is to conduct a complete case analysis, which involves ignoring the missing data. However, this approach may often lead to bias under certain conditions (Schafer & Graham, 2002). To address this issue, multiple imputation (MI) is commonly used. MI is a method of imputation that estimates parameters to impute a value of the missing data and takes uncertainty in regard by imputing a value multiple times (Schafer & Graham, 2002). However, like all imputation methods, MI has its own limitations which should be taken into account. For example, the appropriate model to impute missing values should be selected. Selecting the wrong model can lead to invalid estimates of the parameters (Nguyen et al. 2017). Therefore, it is important to evaluate the performance of MI and other methods in different contexts to assess their robustness. One area that has received limited attention is how missing data affects social network analysis and the strategies to handle missing data in social networks (Huisman, 2009). The lack of studies on this subject leaves a gap in our understanding of the effects of missing data on social networks as it can have significant implications. This paper focuses on missing values in social networks.

1.1 Missing data mechanisms

An important question to ask when handling missing data is why the data is missing. Rubin (1976) describes three different types of missing data known as, missing at random (MAR), missing completely at random (MCAR) and missing not at random (MNAR). When the data is missing purely by chance and the probability of missing is the same for all cases then it is said to be MCAR.

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

If the probability of missing depends on an external factor that is known to the researcher then it is MAR since the researcher knows why it is missing. An example of MAR is when older people are less likely to complete a survey than younger people. Here age is a variable that increases the probability of missingness.

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

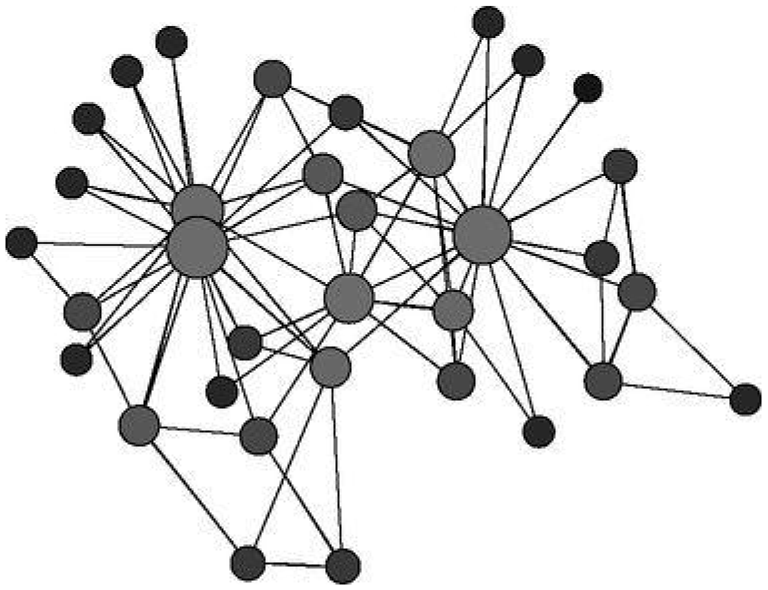
We speak of MNAR when neither MAR or MCAR holds. This means that there is some unobserved variable that influences the probability of missingness. For example a weighing scale might wear out over time which leads to more missing data after long use (van Buuren, 2018).

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

1.2 Social networks

A social network is a network that consists of nodes and edges with nodes that can represent anything that can have a connection and edges, which represent a relation between the two nodes. The relation can be anything from being close friends to a fleeting conversation, a link to a different webpage or any type of connection (Newman, 2018). Figure 1 is a visual representation of a social network.

Figure 1. A network friendship network between members of a club. Every circle represents a member of the club and the line represents the connection between members.



Note. From *Networks*, by M. F. Newman, 2018, Oxford University Press eBooks.

Analysing these networks can tell us something about the dynamics of those networks and see what people are important, what roles they play, what groups are highly interconnected and how things like diseases or rumours spread through a network (Golbeck, 2013). There are numerous ways you can apply network analysis such as, preventing a disease from spreading any further or you can approach people with the most influence within a network to spread a message quickly.

1.3 Relational event history

Relational event history is a type of network data. Relational event history data is a sequence of interactions over time between two individuals in the network (Meijerink-Bosman et al., 2022). The data contains a column of who is at the sending side of the interaction, who is at the receiving end and at what time the interaction occurred. Each row in Table 1 represents a relational event which is defined as a “discrete event generated by a social actor (the sender) towards one or more targets (the receiver)” (Butts, 2008). Essentially representing an interaction between two individuals at a specific moment. For example, sending a message or liking a post.

Table 1. Relational event history data.

time	sender	receiver
10:30	Jesse	Myriam
10:32	Myriam	Jesse
10:34	Jesse	Myriam
10:38	Myriam	Jesse

1.4 Relational event models

A well established model for analysing relational event history data is the relational event model (REM). The REM can predict what event might occur at a certain time in the future. The REH serves as the basis for modelling the probability of the next relational event occurring in the sequence. The probability of who will interact at a certain time is determined by the event rate λ.

$$log\,\lambda = x1 * \beta 1 + x2 * \beta 2 \ldots$$

To calculate the event rate, a risk set of N(N-1) events is constructed. A risk set is a set of all possible interactions at time *t.* Predicting the probability of the subsequent events is influenced by both exogenous and endogenous predictors. Exogenous predictors are variables that provide additional contextual information about the events such as age, gender or personality

(Meijering-Bosman et al., 2023). On the other hand, Endogenous predictors summarise information about the network up to a specific point in time. One example of an endogenous predictor is reciprocity, which is the tendency of a person in the network to initiate the event towards a person given the number of past events with the same person (Leenders et al., 2016). REM allows the study of the dynamics of social networks and estimates how specific factors influence what interaction will happen and when.

The impact of missing data on REM is a topic that has not been extensively researched. However, it is known that missing data can introduce significant biases and errors in the estimation of network properties and parameters. Previous studies have shown that missing data can affect the measurement of network density, centrality, clustering, assortativity, and robustness (Kossinets, 2006; Smith & Moody, 2013). For example, missing data can lead to an underestimation of network density which is a measure of how many connections exist between nodes in a network, and an overestimation of network centralization, meaning that there are a few nodes in the network that have too much influence on the rest of the network (Kossinets, 2006). This can lead to inaccurate conclusions about the structure and properties of the network being studied. Furthermore, Smith and Moody (2013) found that the effects of missing data vary depending on the type of network and the research question being addressed.

This paper explores if the current approach to missing data works in the REH data, in the context of relational event models (REM). The objective of this study is to determine whether multiple imputation is a viable imputation method for REM. We will examine whether MI produces accurate estimations of missing data and evaluate its impact on REM estimates. The findings of this study can provide an effective solution for handling missingness in REM and contribute to further research in social networks where data is missing.

2. Data

The data used for this study is a part of the Apollo 13 mission communications transcript from Kamalabad et al. (2023). The Apollo 13 mission is the infamous moon mission where an exploding oxygen tank forced the astronauts to abandon the moon landing. The data include communication between ground crew and the communication between the flight directors and the astronauts. The contents of the communications are not relevant to this study so the transcript of the messages were excluded. The dataset contains three columns: time, sender and receiver. The data is 3882 rows long, each row is indicative of a communication event with a time point, the sender and the receiver. Both the sender and receiver column consist of 15 unique actors. In total there are 16 unique actors across both sender and receiver this is due to specific actors only occurring in either the sender or receiver column. In the data the names of the actors from the original transcript are replaced with numbers. A digital version of the original Apollo 13 dataset is a publicly available dataset and can be accessed on Github (Tseng, 2020).

Table 2. First four rows of the Apollo data used.

time	sender	receiver
11849.2	18	2
11854.2	2	18
11885.2	18	2
11890.2	2	18
…	…	…

3. Methods

To evaluate the effectiveness of multiple imputation on missing values in relational event history data, the results of a REM on the original data and the multiple imputed data are compared. For the REM some network statistics were calculated using remify (Arena, 2003) and remstats (Meijerink-Bosman et al, 2023). The REM was conducted using the cox proportional hazard (coxph) function from the survival package (Therneau, 2023) in r.

Since a subset from Kamalabad et al. (2023) was used, the same statistics are used as in their study. Which are:

Reciprocity

Reciprocity measures how likely person A is to reply to a message from person B, if person A has received a message from person B recently (Pilny et al. 2016).

Indegree sender

Indegree sender captures the popularity of a sender of a message in the network, based on how many messages they receive.

Outdegree receiver

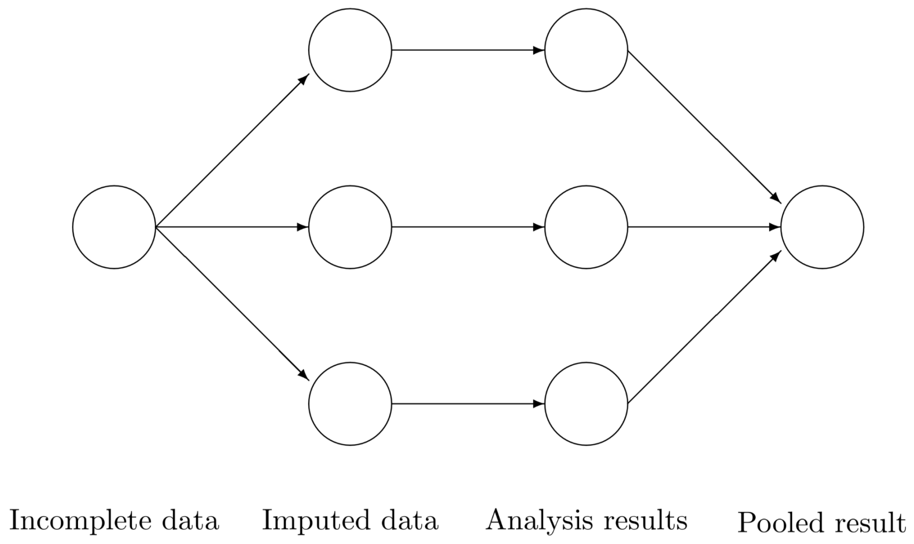
Outdegree receiver reflects the activity level of a person in the network, based on how many messages they send as a receiver.

3.1 Missing data imputation

There are multiple ways to deal with missing data. One can simply ignore the missing data and only analyse complete cases. However, this may result in loss of information, decrease in statistical power and to serious bias when data is not MCAR (Little & Rubin, 1987). Another more popular method is multiple imputation (Rubin, 1987). Multiple imputation is a method where the analyst imputes two or more values for the missing values creating multiple datasets. Then, the desired analysis is conducted on each completed dataset, and the estimates are pooled. Figure 2 is a visual representation of the process.

Multiple imputation has several advantages over other methods. It preserves the variability and uncertainty of the missing data and it allows for valid statistical inference under the assumption of MAR and MCAR (van Buuren, 2018). However, Multiple imputation may not work well when the proportion of missing data is high or when data is MNAR (Sterne et al., 2019).

Figure 2. The steps of multiple imputation. From left to right it starts with the incomplete data and then creates multiple sets of imputed data following an analysis on the data and pools the results in the last step.



Note. Scheme of main steps in multiple imputation. From *Flexible Imputation of Missing Data.* (2nd edition., p. 19), by S. van Buuren, 2018, CRC Press. Copyright by 2018 by Taylor & Francis Group, LLC.

3.2 Missing data generation

Before imputing, missing data had to be generated first. Following the “Strategies for simulated missingness” by Vink (2022), MICE was used to create 100 simulations of missing data. For the missing data generation the percentage missing was set at 20% to introduce a significant amount of missingness without affecting the results of the analysis of the multiple imputations too much. Although it is usually not the case for missing data the missing data mechanism of MCAR was used for the generation. MCAR was chosen because when multiple imputation is not a valid method for imputation in this case then the method will also not hold for MAR.

To avoid losing actors from the dataset due to missing values, we randomly selected 1500 rows as a sufficient set to include all actors, this was done due to time restriction of the study. Then the set was removed from the original dataset and created 33% missingness in the remaining dataset, which resulted in roughly 20% missingness overall when we combined the two sets again. The missing data pattern for the generation was that either sender (actor 1) or receiver (actor 2) was missing at one time but not both. Figure 3 is a plot of the missing pattern of all simulations.

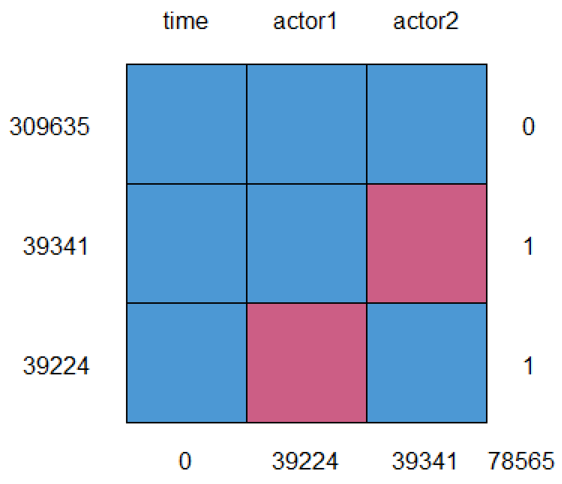


Figure 3. Aggregated missing data pattern of every simulation. The numbers on the left are how often the pattern occurs in the data. Red means the value is missing in the row. The numbers on the bottom are how many values are missing in that column.

3.3 Imputation

For imputation with MICE, the variables sender and receiver were used as predictors for the missing values. The number of imputations were set at 5 and the amount of iterations were set at 5 since inferential validity is often achieved after 5 to 10 iterations (Oberman et al. 2021). The number of iterations is how many times MICE goes through each variable to estimate new values for the missing values. The amount of imputations refer to how many datasets it creates where the missing values were imputed with the given amount of iterations. The method of imputation had to be a custom version of the predictive mean matching (pmm) method. The custom pmm was made to avoid generating loops in the data where actors are initiating interactions with themselves. This conditional mean matching made sure that the imputed value was not the same as the corresponding receiver or sender. Pmm would also ensure imputation of realistic values.

3.4 Analysis

Finally, a relational event model, with Reciprocity, OutdegreeReceiver and IndegreeSender as predictors was fitted and pooled within each simulation, using the coxph function. For the data to work with coxph, the data was altered to look like the data in table 3 where status means whether that interaction happened at time = *t,* yes (1) or no (0). The number of interactions between sender and receiver is the risk set of N(N-1).

Table 3. Example of a risk set. Every row is a possible combination of interaction at a given time. The status column shows whether the combination actually happened in the relational event history.

time	sender	receiver	status
1	a	b	1
1	b	c	0
1	c	a	0
1	a	c	0
1	b	a	0
1	c	b	0
2	a	b	0
…	…	…	…

After running the analysis the results were averaged over a hundred simulations. The imputation method is evaluated using the criteria proposed by Obermann and Vink (2023). The criteria for the assessment are the raw bias, coverage and confidence interval width of the estimates. The raw bias of the estimate is the difference between the estimate and the true estimate. The raw bias should be close to zero, indicating that the estimate is close to the true estimate. Coverage rate is the proportion of confidence intervals that contain the true value. Typically the coverage should not fall below 95%. When the coverage falls below 95%, it indicates an estimate of poor quality. A high coverage rate (> .95) may indicate that the confidence interval is too wide and that the method used is inefficient and leads to inferences that are too conservative. Lastly, the average width of the confidence interval is the difference between the lower and upper end of the confidence interval. A small average width indicates statistical efficiency. However, when the average width is too small it may cause the coverage rate to fall below 95% (van Buuren, 2018).

4. Results and discussion

First a relational event model was performed on the original data to get the true model to compare the simulations to. Table 4 shows the results of the pooled REM. Here, reciprocity has a positive but not significant effect on the chance of an event happening (β = 2.36e-02, p = 0.204) along with outdegreeReceiver, which has a non significant negative effect (β = -9.11e-05, p = .220). However, indegreeSender did have a positive significant effect (β = 4.31e-04, p = < .001).

Table 4

Relational event model on the fully observed Apollo 13 data.

	β	*SE*	z	*p*
reciprocity	2.36e-02	1.86e-02	1.27	.204
indegreeSender	4.31e-04	7.40e-05	5.83	< .001
outdegreeReceiver	-9.11e-05	7.44e-05	-1.23	.220

Based on Obermann and Vink (2023), to evaluate the imputation method the bias, coverage and confidence interval width are calculated from the simulations.

The average REM results of the hundred simulations can be found in Table 5. What is different here is that the estimates of the imputed model are all significant with a very small p value (< 0.001) instead of only indegreeSender being significant as in the true model. The bias of the effects are all near 0 indicating that the bias is negligible. Which was to be expected since the missing mechanism was MCAR (Obermann & Vink, 2023). Further, the coverage for Reciprocity (.99) and IndegreeSender (1.00) are both very high. However, the coverage of OutdegreeReceiver (.97) is somewhat lower but still very high. A coverage that is higher than .95 means that there exists a narrower interval where the confidence interval would still yield valid results. The coverage rates of all variables are very high, which means that the method is effective but there is room for improvement in efficiency. However, this does not affect the validity of the inference (Obermann & Vink, 2023). And lastly, the confidence intervals appear to be very small with small average widths for Reciprocity (4.85e-03), indegreeSender (1.34e-04) and outdegreeReceiver (1.80e-05). The confidence interval width of the three variables appears to be relatively small, which suggests that in combination with the coverage and the bias that multiple imputation is a valid approach for the missing data problem when assuming MCAR in relational event history data. However, the estimates in the simulations are all significant where in the original data only indegreeSender was significant. This can be due to using a fixed set to keep all actors in the data and thus missingness is only generated in a part of the data, resulting in small between variance, smaller confidence intervals and smaller p-values (van Buuren, 2018).

Table 5. Relational event model averaged over 100 simulations with 5 imputations

	True estimates	estimates	*SE*	*p*	95% CI [LL, UL]	cov	bias
Reciprocity	2.36e-02	2.41e-02	0.0009	< .001	[2.17e-02, 2.65e-02]	0.99	5.35e-04
IndegreeSender	4.31e-04	4.22e-04	2.41e-05	< .001	[3.36e-04, 4.89e-04]	1.00	-9.07e-06
OutdegreeReceiver	-9.11e-05	-9.16e-05	3.24e-06	< .001	[-1.00e-04, -8.26e-05]	0.97	4.13e-07

The distribution of all estimates of the simulations can be found in the appendix (Figure 4-6). All the distributions appear to be normally distributed. With only indegreeSender being slightly left skewed.

5. Conclusion

The aim of the study was to see if multiple imputation is a valid method for handling missing data in relational event history data given MCAR. The findings suggest that the method does have some utility and seems to give unbiased estimates when under the MCAR assumption.

However, after imputation the significance of some parameters changed. This could lead to incorrect conclusions from the analysis.

This study has some limitations. First, the assumption of MCAR is not always a reasonable assumption to make when using the imputation on real missing data. However, the MCAR assumption helps to test if multiple imputation can generate accurate estimations. The next step will be to see if multiple imputation still gives accurate estimations when under the MAR assumption.

For the missing data generation 1500 rows were randomly selected to function as a sufficient set so that the risk sets would be equal in length. The problem arises when one actor is infrequent in the dataset. When an actor only appears once or twice in the data it could be that, when generating missing data with the MCAR mechanism, the actor could be eliminated in its entirety purely by chance. Future research should look if this has an effect on the estimates and if it does look for a solution for this problem.

Moreover, the rather big change in significance is a field of interest. The reason why this happens could be due to the fixed rows previously mentioned. More imputations than five could also benefit the results of the imputation method.

Lastly, this study only looked at when one of the sender or receiver was missing, an interesting case to study will be to see if multiple imputation is still effective when complete edges are missing from the network.

Overall the multiple imputation does seem to give accurate estimates when assuming MCAR but the change from not significant to significant estimates is something to be taken into consideration before interpreting the results.

References

Arena, G. (2023). *GitHub - TilburgNetworkGroup/remify: A package that transforms REH data from/to other formats or other sources to a REH structure that is suitable for the packages in remverse*. GitHub. https://github.com/TilburgNetworkGroup/remify

Butts, C. T. (2008). 4. A Relational Event Framework for Social Action. *Sociological Methodology*, *38*(1), 155–200. https://doi.org/10.1111/j.1467-9531.2008.00203.x

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

Kossinets, G. (2006). Effects of missing data in social networks. *Social Networks*, *28*(3), 247–268. https://doi.org/10.1016/j.socnet.2005.07.002

Meijerink-Bosman M, Arena G, Karimova D, Lakdawala R, Shafiee Kamalabad M, Generoso Vieira F. (2023). TilburgNetworkGroup/remstats: Computes Statistics For Relational Event History Data. *rdrr.io*. https://rdrr.io/github/TilburgNetworkGroup/remstats/

Meijerink-Bosman, M., Back, M. D., Geukes, K., Leenders, R. T. a. J., & Mulder, J. (2022). Discovering trends of social interaction behavior over time: An introduction to relational event modeling. *Behavior Research Methods*. https://doi.org/10.3758/s13428-022-01821-8

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

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

Nguyen, C. D., Carlin, J. B., & Lee, K. J. (2017). Model checking in multiple imputation: an overview and case study. *Emerging Themes in Epidemiology*, *14*(1). https://doi.org/10.1186/s12982-017-0062-6

Oberman, H. I., Van Buuren, S., & Vink, G. (2021). Missing the Point: Non-Convergence in Iterative Imputation Algorithms. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2110.11951

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

Pilny, A., Schecter, A., Poole, M. S., & Contractor, N. (2016). An illustration of the relational event model to analyze group interaction processes. *Group Dynamics: Theory, Research, and Practice*, *20*(3), 181–195. https://doi.org/10.1037/gdn0000042

Rubin, D. B. (1987). Multiple Imputation for Nonresponse in Surveys. (1987c). In *Wiley series in probability and statistics*. Wiley. https://doi.org/10.1002/9780470316696

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

Tseng. (2020). *GitHub - issa-tseng/apollo13rt: Apollo 13 in real-time. Flight director's loop, transcript, information, and more.* GitHub. https://github.com/issa-tseng/apollo13rt

Smith, J. C., Moody, J., & Morgan, J. (2017). Network sampling coverage II: The effect of non-random missing data on network measurement. *Social Networks*, *48*, 78–99. https://doi.org/10.1016/j.socnet.2016.04.005

Sterne, J. A., White, I. R., Carlin, J. B., Spratt, M., Royston, P., Kenward, M. G., Wood, A. M., & Carpenter, J. R. (2009). Multiple imputation for missing data in epidemiological and clinical research: potential and pitfalls. *BMJ (Clinical research ed.)*, *338*, b2393. https://doi.org/10.1136/bmj.b2393

Van Buuren, S. (2018). Flexible Imputation of Missing Data, Second Edition. In *Chapman and Hall/CRC eBooks*. https://doi.org/10.1201/9780429492259

Appendix

Figure 4

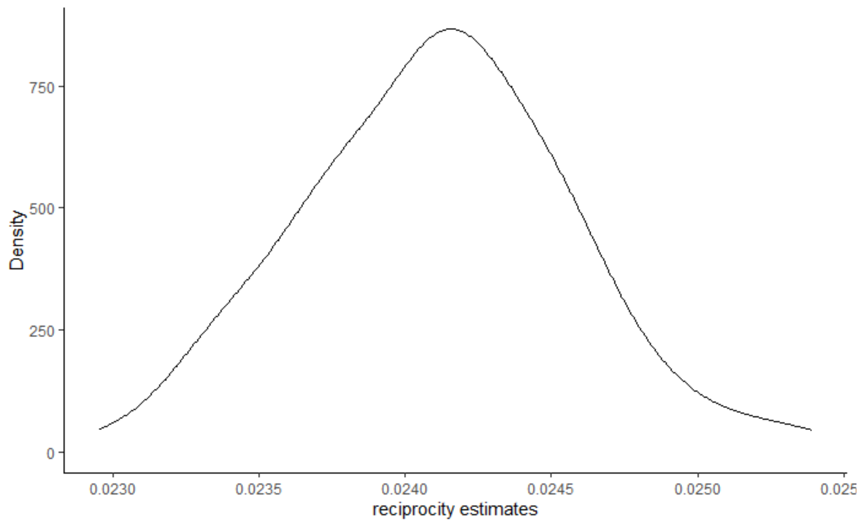


Figure 5

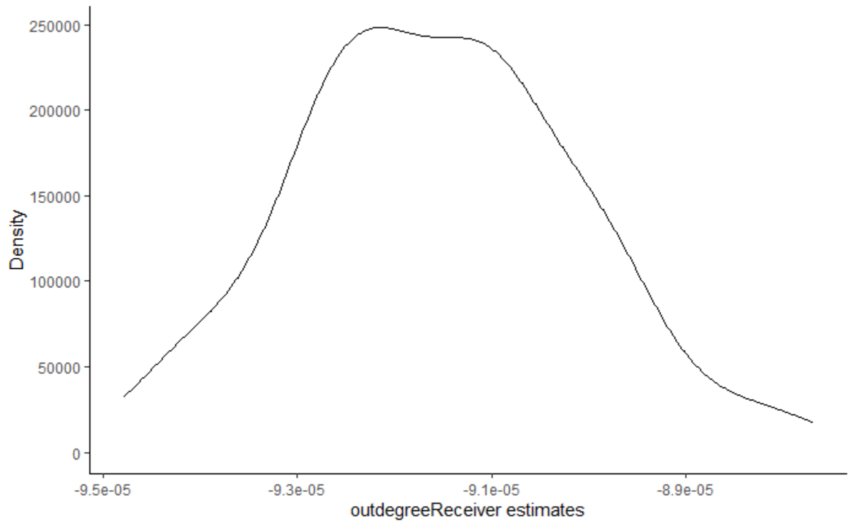
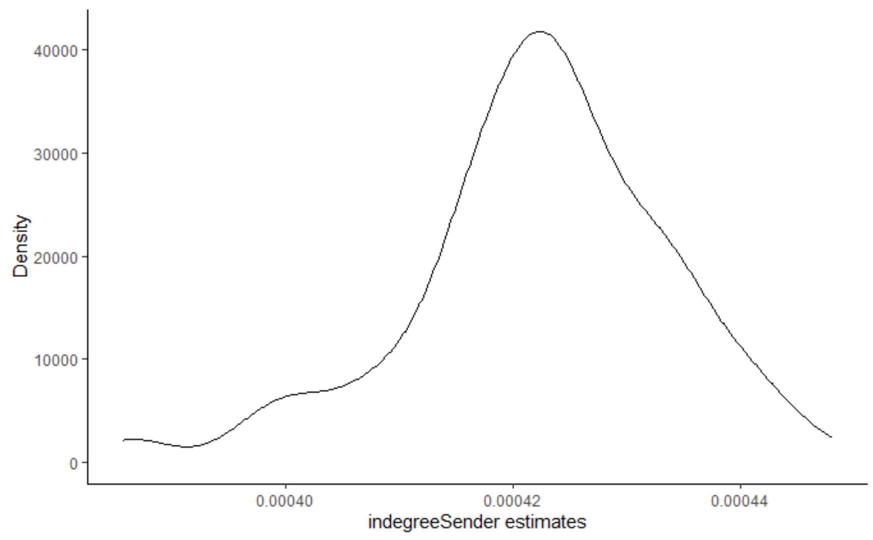


Figure 6



~~~
```{r}
devtools::install_github("TilburgNetworkGroup/remify")
devtools::install_github("TilburgNetworkGroup/remstats")
devtools::install_github("gerkovink/mice@match_conditional")
```

```{r}
##### Load packages
library(mice, warn.conflicts = FALSE)     # for imputation and
amputation
library(purrr, warn.conflicts = FALSE)    # for functional
programming
library(furrr, warn.conflicts = FALSE)    # for functional futures
library(magrittr, warn.conflicts = FALSE) # for pipes
library(dplyr, warn.conflicts = FALSE)    # for data manipulation
library(tibble, warn.conflicts = FALSE)     # for tibbles
library(remstats, warn.conflicts = FALSE) # for REM statistics
~~~

```
library(remify, warn.conflicts = FALSE)   # for converting
library(data.table, warn.conflicts = FALSE)
library(survival, warn.conflicts = FALSE) # for REM analysis
library(tidyverse, warn.conflicts = FALSE)
set.seed(123)

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

whichcol <- c("", "actor2", "actor1")
names(whichcol) <- colnames(apollo)

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


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

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



#### Combine the sufficient set and the incomplete set
make_missing <- function(x, indic){
  sufficient <- x[indic, ]
  miss <- x[-c(indic), ] |>
    # prop is set to .4 to get close to .2 missingness since almost
half of the dataset is used for the sufficient set
```

```
    ampute(prop = .33,
           mech = "MCAR",
           patterns = matrix(c(1,0,1,
                               1,1,0),
                             nrow=2,
                             byrow=TRUE)) %>%
    .$amp
  combined <- rbind(sufficient,
                    miss)
  return(combined[order(combined$time), ]) # sort it all like apollo
}

##### Missing pattern
pattern <- matrix(c(1,0,1,1,1,0), nrow=2, byrow=TRUE)

##### predictor matrix
predictormatrix <- matrix(c(0,0,0,0,0,1,0,1,0), nrow=3, byrow=TRUE)

##### Model-based finite populations
mbased_finite_apollo <-
  furrr::future_map(1:100, ~ {
    make_missing(apollo, indic) %>%
      mice(m = 5,
           maxit = 5,
           method = method,
           whichcolumn = whichcol,
           predictorMatrix = predictormatrix,
           print = FALSE)
  }, .options = furrr_options(seed = 123))

##### Missing data pattern of all simulations.
missing_pattern <- mbased_finite_apollo %>%
  map(~.x %>% .$data) %>%
  do.call("rbind", .) %>%
  md.pattern()

##### Defining effects for the relational event model
effects <- ~ -1 + reciprocity(scaling = ("std")) + indegreeSender() +
outdegreeReceiver()

##### Function to get the statistics of the previously defined
effects.
stats_function <- function(data) {

  # remify the data
  reh <- remify::remify(edgelist = data, model = "tie")

  # calculate effect statistics
  statsObject_imp <- remstats(reh = reh, tie_effects = effects)

  # Return the statistics
  return(statsObject_imp)
}

##### Function for making the data compatible with coxph()
prepare_coxph_data <- function(statsObject, apollo) {
  risk_sets <- attr(statsObject, "riskset")
  risk_sets <- risk_sets %>% select(-'id')

  # Get the times
  time <- apollo$time

  # merge riskset with each timepoint
  combined <- merge(risk_sets, time, by = NULL)

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

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

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

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

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

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

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

  return(combined)
}



###### TRUE ANALYSIS
true.reh <- remify(edgelist = apollo,
                   model = "tie")
# calculate stats
stats <- remstats(tie_effects = effects,
                  reh = true.reh)
# use the function to create the correct format of the dataframe
true.cox.set <- prepare_coxph_data(stats, PartOfApollo_13)
# fit cox model
true.cox.fit <- coxph(Surv(time, status) ~ reciprocity +
indegreeSender +
                        outdegreeReceiver,
                      data=true.cox.set)
```

```
true <- coefficients(true.cox.fit)

true.cox.fit

###### Running the REM on all simulations divided into 10 blocks.
Results1 <-
  mbased_finite_apollo[1:10] %>%
  map(~.x %>% # for every simulation
        complete("all") %>%
        map(~.x %>% # for every imputation
              stats_function() %>% # do stats function
              prepare_coxph_data(apollo = apollo) %$% # prepare cox
ph
              coxph(Surv(time, status) ~
                      reciprocity +
                      indegreeSender +
                      outdegreeReceiver)) %>%
        pool(custom.t = ".data$b + .data$b / .data$m") %>% ## pool
coefficients
        .$pooled %>% ## extract table of pooled coefficients
          mutate(true = true, # add true
                 df = m-1,  # correct df
                 riv = Inf, # correct riv
                 std.error = sqrt(t), # standard error
                 statistic = estimate / std.error, # test statistic
                 p.value = 2 * (pt(abs(statistic),
                                   pmax(df, 0.001),
                                   lower.tail = FALSE)), # correct
p.value
                 `2.5 %` = estimate - qt(.975, df) * std.error, #
lower bound CI
                 `97.5 %` = estimate + qt(.975, df) * std.error, #
upper bound CI
                 cov = `2.5 %` < true & true < `97.5 %`, # coverage
                 bias = estimate - true) %>% # bias
          select(term, m, true, estimate, std.error, statistic,
p.value,
                 riv, `2.5 %`, `97.5 %`, cov, bias) %>%
          column_to_rownames("term") # `term` as rownames
    )


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

Results3 <-
  mbased_finite_apollo[21:30] %>%
  map(~.x %>% # for every simulation
        complete("all") %>%
        map(~.x %>% # for every imputation
              stats_function() %>% # do stats function
              prepare_coxph_data(apollo = apollo) %$% # prepare cox
ph
              coxph(Surv(time, status) ~
                      reciprocity +
                      indegreeSender +
                      outdegreeReceiver)) %>%
        pool(custom.t = ".data$b + .data$b / .data$m") %>% # pool
coefficients
        .$pooled %>% # extract table of pooled coefficients
          mutate(true = true, # add true
                 df = m-1,  # correct df
                 riv = Inf, # correct riv
                 std.error = sqrt(t), # standard error
                 statistic = estimate / std.error, # test statistic
                 p.value = 2 * (pt(abs(statistic),
                                   pmax(df, 0.001),
                                   lower.tail = FALSE)), # correct
p.value
                 `2.5 %` = estimate - qt(.975, df) * std.error, #
lower bound CI
                 `97.5 %` = estimate + qt(.975, df) * std.error, #
upper bound CI
                 cov = `2.5 %` < true & true < `97.5 %`, # coverage
                 bias = estimate - true) %>% # bias
          select(term, m, true, estimate, std.error, statistic,
p.value,
                 riv, `2.5 %`, `97.5 %`, cov, bias) %>%
          column_to_rownames("term") # `term` as rownames
  )
Results4 <-
```

```
  mbased_finite_apollo[31:40] %>%
  map(~.x %>% # for every simulation
        complete("all") %>%
        map(~.x %>% # for every imputation
              stats_function() %>% # do stats function
              prepare_coxph_data(apollo = apollo) %$% # prepare cox
ph
              coxph(Surv(time, status) ~
                      reciprocity +
                      indegreeSender +
                      outdegreeReceiver)) %>%
        pool(custom.t = ".data$b + .data$b / .data$m") %>% # pool
coefficients
        .$pooled %>% # extract table of pooled coefficients
          mutate(true = true, # add true
                 df = m-1,  # correct df
                 riv = Inf, # correct riv
                 std.error = sqrt(t), # standard error
                 statistic = estimate / std.error, # test statistic
                 p.value = 2 * (pt(abs(statistic),
                                   pmax(df, 0.001),
                                   lower.tail = FALSE)), # correct
p.value
                 `2.5 %` = estimate - qt(.975, df) * std.error, #
lower bound CI
                 `97.5 %` = estimate + qt(.975, df) * std.error, #
upper bound CI
                 cov = `2.5 %` < true & true < `97.5 %`, # coverage
                 bias = estimate - true) %>% # bias
          select(term, m, true, estimate, std.error, statistic,
p.value,
                 riv, `2.5 %`, `97.5 %`, cov, bias) %>%
          column_to_rownames("term") # `term` as rownames
  )
Results5 <-
  mbased_finite_apollo[41:50] %>%
  map(~.x %>% # for every simulation
        complete("all") %>%
        map(~.x %>% # for every imputation
              stats_function() %>% # do stats function
              prepare_coxph_data(apollo = apollo) %$% # prepare cox
ph
              coxph(Surv(time, status) ~
                      reciprocity +
                      indegreeSender +
                      outdegreeReceiver)) %>%
        pool(custom.t = ".data$b + .data$b / .data$m") %>% # pool
coefficients
        .$pooled %>% # extract table of pooled coefficients
          mutate(true = true, # add true
                 df = m-1,  # correct df
                 riv = Inf, # correct riv
                 std.error = sqrt(t), # standard error
                 statistic = estimate / std.error, # test statistic
                 p.value = 2 * (pt(abs(statistic),
                                   pmax(df, 0.001),
                                   lower.tail = FALSE)), # correct
p.value
                 `2.5 %` = estimate - qt(.975, df) * std.error, #
lower bound CI
                 `97.5 %` = estimate + qt(.975, df) * std.error, #
upper bound CI
                 cov = `2.5 %` < true & true < `97.5 %`, # coverage
                 bias = estimate - true) %>% # bias
          select(term, m, true, estimate, std.error, statistic,
p.value,
                 riv, `2.5 %`, `97.5 %`, cov, bias) %>%
          column_to_rownames("term") # `term` as rownames
  )
Results6 <-
  mbased_finite_apollo[51:60] %>%
  map(~.x %>% # for every simulation
        complete("all") %>%
        map(~.x %>% # for every imputation
              stats_function() %>% # do stats function
              prepare_coxph_data(apollo = apollo) %$% # prepare cox
```

```
ph
              coxph(Surv(time, status) ~
                      reciprocity +
                      indegreeSender +
                      outdegreeReceiver)) %>%
        pool(custom.t = ".data$b + .data$b / .data$m") %>% # pool
coefficients
        .$pooled %>% # extract table of pooled coefficients
          mutate(true = true, # add true
                 df = m-1,  # correct df
                 riv = Inf, # correct riv
                 std.error = sqrt(t), # standard error
                 statistic = estimate / std.error, # test statistic
                 p.value = 2 * (pt(abs(statistic),
                                   pmax(df, 0.001),
                                   lower.tail = FALSE)), # correct
p.value
                 `2.5 %` = estimate - qt(.975, df) * std.error, #
lower bound CI
                 `97.5 %` = estimate + qt(.975, df) * std.error, #
upper bound CI
                 cov = `2.5 %` < true & true < `97.5 %`, # coverage
                 bias = estimate - true) %>% # bias
          select(term, m, true, estimate, std.error, statistic,
p.value,
                 riv, `2.5 %`, `97.5 %`, cov, bias) %>%
          column_to_rownames("term") # `term` as rownames
  )
Results7 <-
  mbased_finite_apollo[61:70] %>%
  map(~.x %>% # for every simulation
        complete("all") %>%
        map(~.x %>% # for every imputation
              stats_function() %>% # do stats function
              prepare_coxph_data(apollo = apollo) %$% # prepare cox
ph
              coxph(Surv(time, status) ~
                      reciprocity +
                      indegreeSender +
                      outdegreeReceiver)) %>%
        pool(custom.t = ".data$b + .data$b / .data$m") %>% # pool
coefficients
        .$pooled %>% # extract table of pooled coefficients
          mutate(true = true, # add true
                 df = m-1,  # correct df
                 riv = Inf, # correct riv
                 std.error = sqrt(t), # standard error
                 statistic = estimate / std.error, # test statistic
                 p.value = 2 * (pt(abs(statistic),
                                   pmax(df, 0.001),
                                   lower.tail = FALSE)), # correct
p.value
                 `2.5 %` = estimate - qt(.975, df) * std.error, #
lower bound CI
                 `97.5 %` = estimate + qt(.975, df) * std.error, #
upper bound CI
                 cov = `2.5 %` < true & true < `97.5 %`, # coverage
                 bias = estimate - true) %>% # bias
          select(term, m, true, estimate, std.error, statistic,
p.value,
                 riv, `2.5 %`, `97.5 %`, cov, bias) %>%
          column_to_rownames("term") # `term` as rownames
  )
Results8 <-
  mbased_finite_apollo[71:80] %>%
  map(~.x %>% # for every simulation
        complete("all") %>%
        map(~.x %>% # for every imputation
              stats_function() %>% # do stats function
              prepare_coxph_data(apollo = apollo) %$% # prepare cox
ph
              coxph(Surv(time, status) ~
                      reciprocity +
                      indegreeSender +
                      outdegreeReceiver)) %>%
        pool(custom.t = ".data$b + .data$b / .data$m") %>% # pool
```

```
coefficients
        .$pooled %>% # extract table of pooled coefficients
          mutate(true = true, # add true
                 df = m-1,  # correct df
                 riv = Inf, # correct riv
                 std.error = sqrt(t), # standard error
                 statistic = estimate / std.error, # test statistic
                 p.value = 2 * (pt(abs(statistic),
                                   pmax(df, 0.001),
                                   lower.tail = FALSE)), # correct
p.value
                 `2.5 %` = estimate - qt(.975, df) * std.error, #
lower bound CI
                 `97.5 %` = estimate + qt(.975, df) * std.error, #
upper bound CI
                 cov = `2.5 %` < true & true < `97.5 %`, # coverage
                 bias = estimate - true) %>% # bias
          select(term, m, true, estimate, std.error, statistic,
p.value,
                 riv, `2.5 %`, `97.5 %`, cov, bias) %>%
          column_to_rownames("term") # `term` as rownames
  )
Results9 <-
  mbased_finite_apollo[81:90] %>%
  map(~.x %>% # for every simulation
        complete("all") %>%
        map(~.x %>% # for every imputation
              stats_function() %>% # do stats function
              prepare_coxph_data(apollo = apollo) %$% # prepare cox
ph
              coxph(Surv(time, status) ~
                      reciprocity +
                      indegreeSender +
                      outdegreeReceiver)) %>%
        pool(custom.t = ".data$b + .data$b / .data$m") %>% # pool
coefficients
        .$pooled %>% # extract table of pooled coefficients
          mutate(true = true, # add true
                 df = m-1,  # correct df
                 riv = Inf, # correct riv
                 std.error = sqrt(t), # standard error
                 statistic = estimate / std.error, # test statistic
                 p.value = 2 * (pt(abs(statistic),
                                   pmax(df, 0.001),
                                   lower.tail = FALSE)), # correct
p.value
                 `2.5 %` = estimate - qt(.975, df) * std.error, #
lower bound CI
                 `97.5 %` = estimate + qt(.975, df) * std.error, #
upper bound CI
                 cov = `2.5 %` < true & true < `97.5 %`, # coverage
                 bias = estimate - true) %>% # bias
          select(term, m, true, estimate, std.error, statistic,
p.value,
                 riv, `2.5 %`, `97.5 %`, cov, bias) %>%
          column_to_rownames("term") # `term` as rownames
  )
Results10 <-
  mbased_finite_apollo[91:100] %>%
  map(~.x %>% # for every simulation
        complete("all") %>%
        map(~.x %>% # for every imputation
              stats_function() %>% # do stats function
              prepare_coxph_data(apollo = apollo) %$% # prepare cox
ph
              coxph(Surv(time, status) ~
                      reciprocity +
                      indegreeSender +
                      outdegreeReceiver)) %>%
        pool(custom.t = ".data$b + .data$b / .data$m") %>% # pool
coefficients
        .$pooled %>% # extract table of pooled coefficients
          mutate(true = true, # add true
                 df = m-1,  # correct df
                 riv = Inf, # correct riv
                 std.error = sqrt(t), # standard error
                 statistic = estimate / std.error, # test statistic
                 p.value = 2 * (pt(abs(statistic),
                                   pmax(df, 0.001),
                                   lower.tail = FALSE)), # correct 
p.value
                 `2.5 %` = estimate - qt(.975, df) * std.error, # 
lower bound CI
                 `97.5 %` = estimate + qt(.975, df) * std.error, # 
upper bound CI
                 cov = `2.5 %` < true & true < `97.5 %`, # coverage
                 bias = estimate - true) %>% # bias
          select(term, m, true, estimate, std.error, statistic, 
p.value,
                 riv, `2.5 %`, `97.5 %`, cov, bias) %>%
          column_to_rownames("term") # `term` as rownames
  )

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

#saving Results
save(Results, file = "simulations.RData")

##### Adding percentage bias and average width to the Results

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

~~~
df[["true"]])


Results_with_extra <- lapply(Results, function(df) {
  df$PB <- PB(df)
  df$AW <- AW(df)
  df
})

###### Average sims
Reduce("+", Results_with_extra) / length(mbased_finite_apollo)

```
```{r}
##### Creating a list of all estimates for every statistic
reciprocity <- Results %>%
  map(~.x %>% .["reciprocity", ]) %>%
  do.call("rbind", .)

indegreeSender <- Results %>%
  map(~.x %>% .["indegreeSender", ]) %>%
        do.call("rbind", .)

outdegreeReceiver <- Results %>%
  map(~.x %>% .["outdegreeReceiver", ]) %>%
        do.call("rbind", .)

```
```{r}
##### A density plot of all estimates for every statistic
outdegreeReceiver %>%
  ggplot(aes(x = estimate)) +
  geom_density() +
  labs(x = "outdegreeReceiver estimates", y = "Density") +
  theme_classic()

indegreeSender %>%
  ggplot(aes(x = estimate)) +
~~~

```
  geom_density() +
  labs(x = "indegreeSender estimates", y = "Density") +
  theme_classic()

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

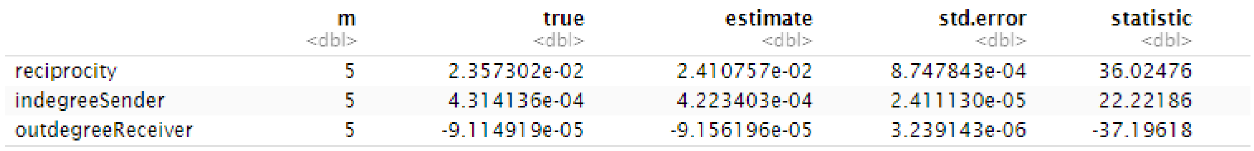
True model

```
coxph(formula = Surv(time, status) ~ reciprocity + indegreeSender +
    outdegreeReceiver, data = true.cox.set)

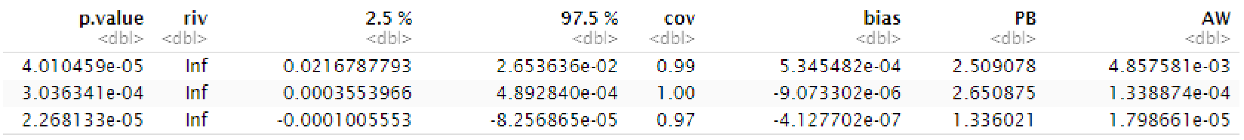
                        coef  exp(coef)   se(coef)      z        p
reciprocity        2.357e-02  1.024e+00  1.858e-02  1.269    0.204
indegreeSender     4.314e-04  1.000e+00  7.400e-05  5.830 5.55e-09
outdegreeReceiver -9.115e-05  9.999e-01  7.437e-05 -1.226    0.220

Likelihood ratio test=41.29  on 3 df, p=5.673e-09
n= 931680, number of events= 3882
```

Evaluation of multiple imputation



	m <dbl>	true <dbl>	estimate <dbl>	std.error <dbl>	statistic <dbl>
reciprocity	5	2.357302e-02	2.410757e-02	8.747843e-04	36.02476
indegreeSender	5	4.314136e-04	4.223403e-04	2.411130e-05	22.22186
outdegreeReceiver	5	-9.114919e-05	-9.156196e-05	3.239143e-06	-37.19618



p.value <dbl>	riv <dbl>	2.5 % <dbl>	97.5 % <dbl>	cov <dbl>	bias <dbl>	PB <dbl>	AW <dbl>
4.010459e-05	Inf	0.0216787793	2.653636e-02	0.99	5.345482e-04	2.509078	4.857581e-03
3.036341e-04	Inf	0.0003553966	4.892840e-04	1.00	-9.073302e-06	2.650875	1.338874e-04
2.268133e-05	Inf	-0.0001005553	-8.256865e-05	0.97	-4.127702e-07	1.336021	1.798661e-05
```