Evaluating the Performance of Multiple Imputation by Chained Equations versus Complete Case Analysis through Simulations

Faizan Khalid Mohsin[1,*], **Jasmine Ngyuen**[2], **Hongyang Hu**[2], **and Nathan Taback**[2]

[1]University of Toronto, Dalla Lana School of Public Health, Toronto, Canada
[2]University of Toronto, Department of Statistical Sciences, Toronto, Canada
[*]faizan.mohsin@mail.utoronto.ca

ABSTRACT

Background: The challenge of missing data is encountered by all disciplines of science. This is especially true when working with longitudinal data. One approach has been to use only those observations that have no missing data. This is called complete case analysis (CC). Another popular method is multiple imputation (MI) developed by Rubin. There are several implementations of multiple imputation and among them, multiple imputation by chained equations using predictive mean matching (MICE), has become popular. **Purpose:** We evaluate the performance of MICE and CC using simulated longitudinal data by methodologically introducing missing data. **Methodology:** Firstly, we simulate 100 data sets that are longitudinal in nature. We run mixed-effects model on the complete data sets and obtain the true estimate of the coefficients of the model. Now, in the simulated data we introduce three types of missingness: missing completely at random (MCAR), missing at random (MAR), and not missing at random (MNAR). We also methodologically introducing missingness at varying percentages. After this we use MICE and CC on the data sets and run a mixed-effects model to obtain estimates of the coefficients and also create 95% confidence intervals. Finally, we compare these estimates to the true estimates of the modl. **Results:** We find that for both, MICE and CC, when missing data is at most 50% and the missingness is either MCAR or MAR. Both missing data methods are very accurate. However, for MNAR, complete case analysis' confidence interval does capture the true estimate whereas MICE's does not. For over 50% missing data, we did not even have enough complete cases to perform a complete case analysis. However, MICE did capture the true estimate even at 60% missing data when the data was MCAR or MAR with the point estimates being very close to the true estimate. It did not, however, capture the true estimate in its confidence interval when the missing data was MNAR. **Conclusion:** Both methods perform equally well when missing data is less than 50% with MICE being more precise (smaller CI and a smaller mse). However, when the missing data is MNAR, complete case analysis slightly outperforms MICE when missing data is less that 50%, simply because it is much higher variance and thus bigger confidence intervals. Now, for over 50% missing data there might not even be enough complete cases to perform a complete case analysis and so multiple imputation is very advantageous. Further, MICE performs very well even when missing data is over 50% as long as the missingness is not MNAR. One possible reason CC and MI have similar performance for MAR is that the variables of our data sets were not correlated with one another, hence not allowing MICE to fully take advantage of this (which we know it can) and outperform CC which it theoretically should.

1 Introduction

These days, with more and more data, the problem of missing data is also increasing. Before the advent of computational power, and even today, for computational convenience, missing data is sometimes dealt with by simply keeping the observations with no missing data, that is, the complete cases and then analyzing this data set with only complete cases. This approach is called the complete case analysis. However, there have been made many advancements in the past several years and we will be discussing one of them known as multiple imputation (MI) by Rubin. Before we do so, we will describe the three different types of missing data.

1.1 Types of Missing Data

Missing data is divided into three categories. The first is called missing completely at random (MCAR). As the name suggest, this type of missing data is missing completely at random[1]. There is no associative pattern between the missing data and the study or any aspect of the study. In such a case the missing data can simply be ignored. For example, if in a study where people's blood pressures are being measured and some people are unable to come due to a completely unrelated reason which has nothing to do with the person's blood pressure or anything else that we might be of interest in the study, such as relating to the person's age, gender, or ethnicity. A truly random reason such as being in an accident, or simply forgetting about it, would be considered missing completely at random.

The second is called missing at random (MAR). This happens when the missing data depends on the other variables in the data set but not not the missing values[1]. For instance, if we are interested in people's blood pressure then if older people tend to have more missing values for their blood pressures then this could be a case where the missing data would be considered missing at random since the missingness depends on a person's age and not what their blood pressure is.

The third is called not missing at random (NMAR or equivalently MNAR). This is when the missing data depends on the missing values themselves[1]. For example, in a study where people's blood pressure is being measured, the missing data will be considered not missing at random (NMAR) if the higher the person's blood pressure is, the higher the likelihood that he will not come in to have his blood pressure measured, hence causing the missingness of the blood pressure to depend on the blood pressure itself. If data is not missing at random, then it can have a significant impact on the study and can seriously bias the results.

1.2 Complete Case Analysis

Complete case analysis is one of the methods to deal with missing data. Simply put, only those observations that have no missing data are retained in the data set. All others are deleted, hence it is called complete cases. This method is also sometimes known as listwise deletion. As the reader can surmise, in extreme cases when every observation has some missing data, all the data points would be deleted. Thus, it is typically recommended that variables with a lot of missing data be removed before hand. Further, ideally, the data sets should be large with little missing data so that the total percentage of observations deleted is low. Lastly, one possible issue of only keeping complete cases in the data set, is that it could introduce bias. For example, if the data is longitudinal, and there is missing data due to lack of follow-up for the last few time points because sicker people choose to discontinue (data being not missing at random). Deleting all such people and not using any of their data will introduce bias into the data set and any subsequent analysis. However, despite all of these issues, it is very attractive to use as it is very simple and easy to implement. Hence, one must be careful when using complete case analysis.

1.3 Multiple Imputation

A more advanced method in which all the observations are kept is called multiple imputation. There are many multiple imputation methods, however all of them have three main steps[2]. The first step is to impute the missing values 'm' times creating 'm' complete data sets. Each data set is slightly different from the others, due to the randomness involved in the process, causing there to be variation between the competed data sets. The second step is to do the statistical analysis on each completed data set resulting into 'm' analyses. Hence, if we were estimating a parameter of a model, then we could have 'm' estimates, one from each data set. The third and last step is to combine all the 'm' analyses into the final result. Hence, for our 'm' estimates we could simply take their average[2,3]. Figure 1 illustrates the multiple imputation method.

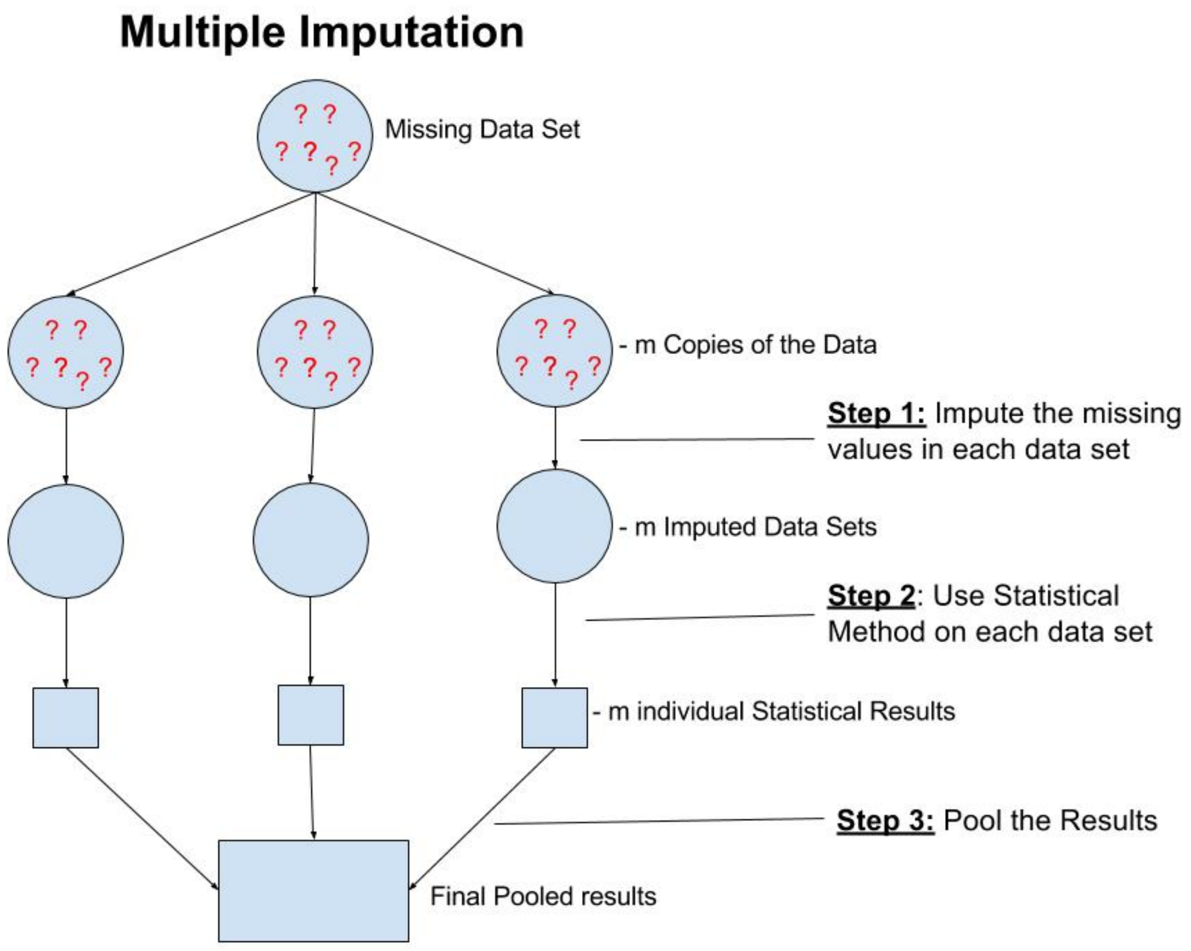

Multiple Imputation
Missing Data Set
- m Copies of the Data
Step 1: Impute the missing values in each data set
- m Imputed Data Sets
Step 2: Use Statistical Method on each data set
- m individual Statistical Results
Step 3: Pool the Results
Final Pooled results

Figure 1. Diagram illustrating the three main steps of the Multiple Imputation Method.

An advantage of this method is that, to a degree, it accounts for the fact that the missing values are not known. I.e. it accounts for the fact that even though we may have an idea of the range of the possible values for the missing value, we are uncertain to what exact the value is. Hence, by creating 'm' different data sets we are introducing a new source of variation in our analysis which causes our estimate to have two sources that contribute towards it variance. Hence, two variance components. One is the usual variance of the data set, the other one is the new variance: the variance caused by the variation between data sets. The new variance component causes the estimate to have a higher variance. This accounts for the fact that we do not know what the missing value is.

Further, if the assumption that the missing data is either MCAR or MAR and the three steps outlined above are done properly, then the estimates produced by the MI will have very desirable statistical properties. They will be consistent, almost asymptotically efficient and asymptotically normal[1].

1.4 Multiple Imputation by Chained Equations

There are many methods as the reader can imagine how the missing values can be imputed in step one of the multiple imputation method. One popular semi-parametric method is called multiple imputation by chained equations[4]. This is also the method we employed for doing multiple imputation.

The simple idea behind multiple imputation by chained equations is performing a regression on the variable with missing values using the other variables as regressors and then using the predicted values plus a random error, from the regression of the missing value to be the new imputed value. And then some other variable with missing values is regressed on using all the other variables including the variable whose missing values were just imputed, and its missing values are imputed in the same fashion. This is done for all the variables with missing data and can be thought of as one cycle. This cycle is then repeated several times, typically ten or until convergence is reached for the missing values.[1,5]

One clear advantage of this method is that if the variables in the data are correlated to one another then missing values in one variable could be fairly well estimated by the other variables through the successive regressions. However, on the flip side if the variables are not related to one another then performing regression would not substantially improve the estimates of the missing value.

Further, we mentioned that we are using predictive mean matching for MICE. It is one of the options in the R ”mice” package. We use this method because it is somewhat robust against the imputation model assumptions. For example the normality assumption, residuals being homoscedastic, and the associations being linear. Hence, predictive mean matching provides some robustness against the violation of these assumptions[6].

We will investigate these advantages of multiple imputation by chained equations via evaluating its performance and will be comparing it to that of the complete case analysis.

2 Methods

We will first give an overview of the entire process and then later go into depth for each step.

2.1 Overview of the Simulation and Analysis Process

We simulate 100 data sets that are longitudinal in nature. We first introduce missing data that is missing completely at random (MCAR), then missing data that is missing at random (MAR) and finally, data that is not missing at random (MNAR). For each type of missing data, we introduce missingness at 20%, 30%, 40% and 60%. After this we use MICE and CC on the data sets to handle the missing data and then run a mixed-effects model to obtain 100 coefficient estimates for each (missing data) method. We take the average of these estimates to obtain the final coefficient estimates. We also create 95% confidence intervals of these estimates. We then compare them to the true estimates that we obtain by running the mixed-effects model on the complete data sets (before introducing any missingness) obtaining 100 true coefficient estimates of which we again take the average and refer it to as the true coefficient estimates and create a 95% confidence intervals for it as well.

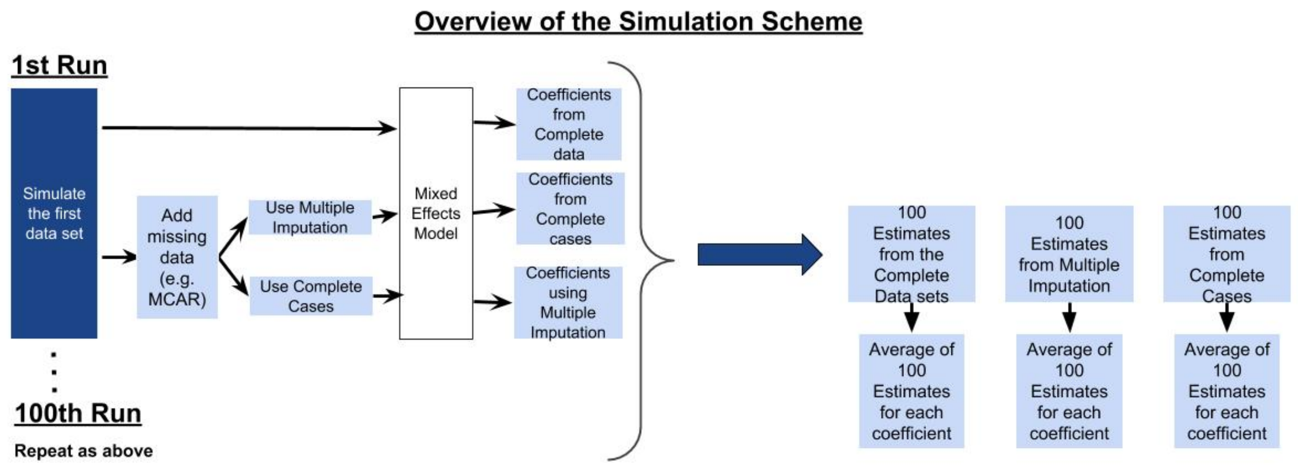

Overview of the Simulation Scheme
1st Run
Simulate the first data set
Add missing data (e.g. MCAR)
Use Multiple Imputation
Use Complete Cases
Mixed Effects Model
Coefficients from Complete data
Coefficients from Complete cases
Coefficients using Multiple Imputation
100 Estimates from the Complete Data sets
100 Estimates from Multiple Imputation
100 Estimates from Complete Cases
Average of 100 Estimates for each coefficient
Average of 100 Estimates for each coefficient
Average of 100 Estimates for each coefficient
100th Run
Repeat as above

Figure 2. General overview of the simulation scheme.

2.2 Simulating the Longitudinal data

We create a longitudinal data set of 221 patients with a total of five time points (one baseline measurement and four follow-up measurements) with 1105 as the total number of data points. The variable of interest we will be using is FACIT-fatigue score (referred to simply as patient's fatigue score) which has the range 0-52[7]. We will also be simulating each person's age, gender, cancer stage and type of cancer. We did this so that the simulated data would resemble real world data sets.

To simulate the longitudinal data we used the R software.

- For the age variable we made it normally distributed with mean and standard deviation approximately 55 and 10 years, respectively.

- For the gender a binomial distribution was used with the probability of 80% of being female and 20% of being male.

- For cancer type, subjects can have one of five of the following cancers: ”Breast”, ”Head and Neck”, ”Leukemia”, ”Prostate” or ”Lung” with females having the probability of 70%, 10%, 15%, 0%, and 5% respectively for each of the cancers and males having the probability of 0%, 10%, 20%, 60% and 10% respectively. As you can see females have zero probability of having prostate cancer and males have zero probability of having breast cancer.

- For the cancer stage, we made the higher cancer stages progressively more unlikely to occur.

- For the fatigue scores we made them normally distributed. To create the association between the patients' fatigue scores and the time points, we increased the mean fatigue score for each subsequent time point such that it would have a slope of 4.

We illustrate the structure of the longitudinal data set below in Figure 3. One can see that there are four levels to the data.

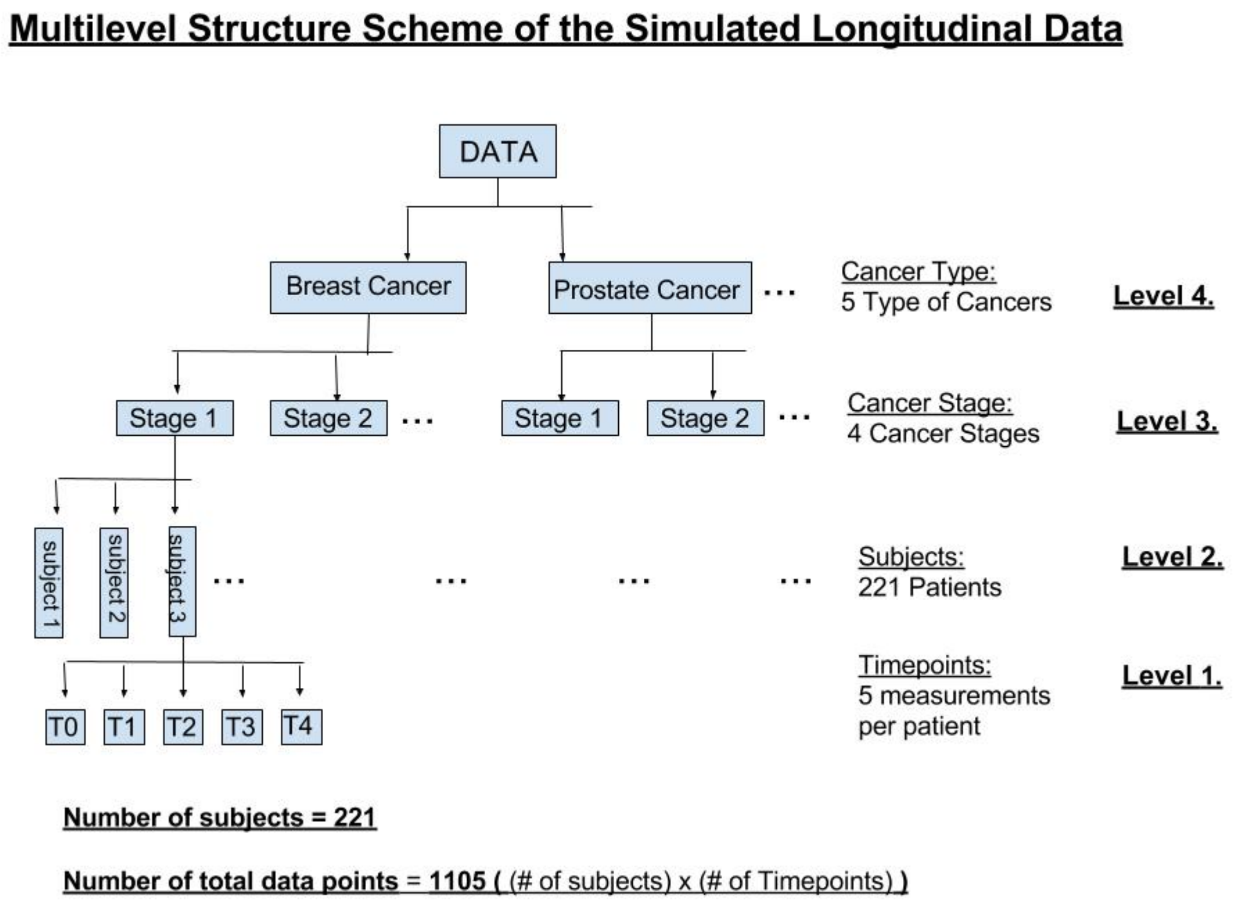


Figure 3. Multilevel structure of the simulated data set.

We also present in Figure 4 below the correlation structure of our simulated data between the different variables. As you can see that time point and the fatigue scores are relatively moderately correlated with a Pearson correlation coefficient of $r = 0.58$. This is by design as you would recall that we set the fatigue scores to be normally distributed with increasing mean for each subsequent time point. Thus, fatigue scores and time points are correlated. You will also observe that the cancer type and the gender are also correlated with correlation coefficient of -0.71. This is also by design since only females can get breast cancer and only males can get prostrate cancer. For all the other variables they are uncorrelated to one another.

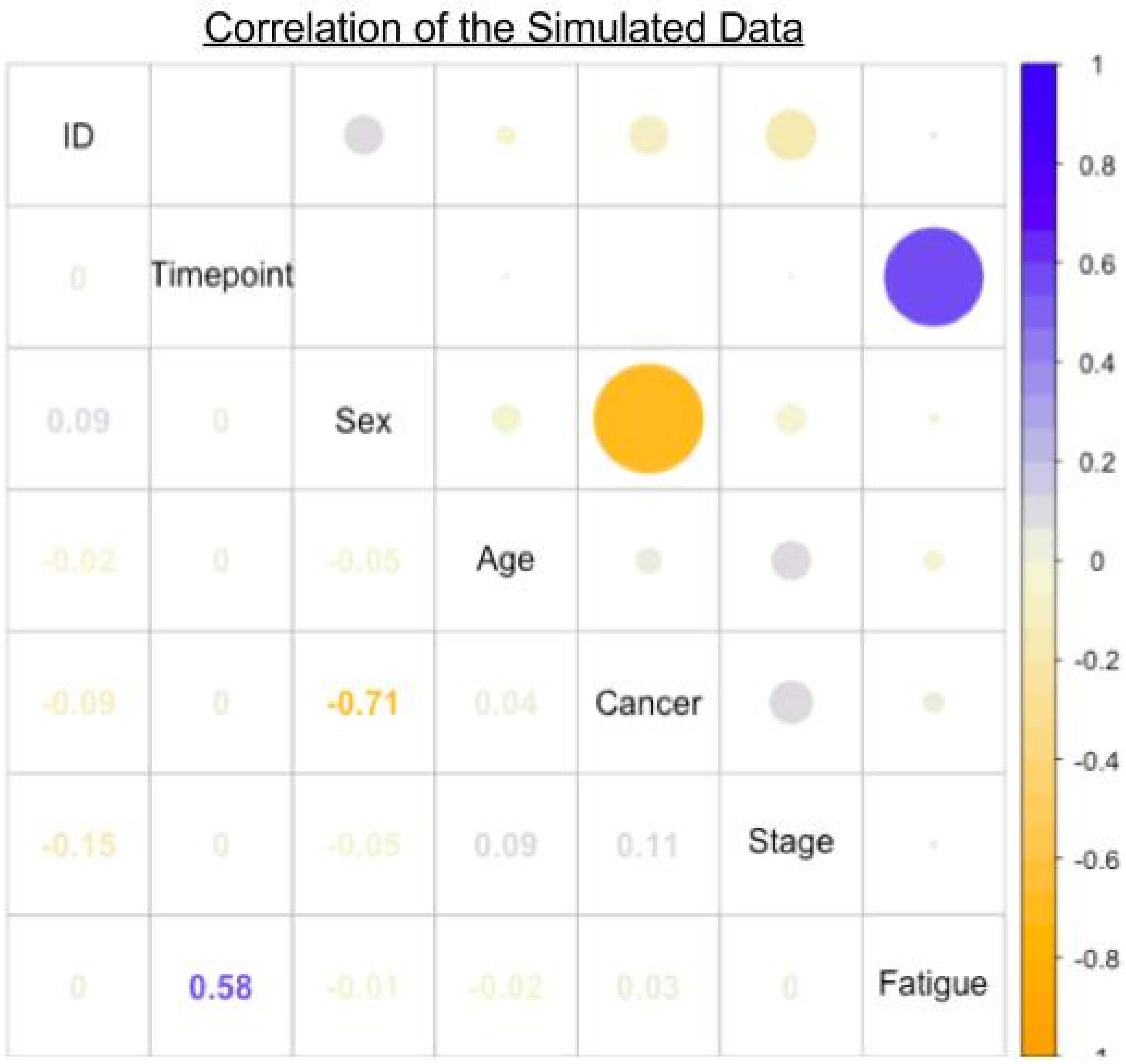


Figure 4. Correlation structure of the simulated data set

2.3 Simulating the Missing Data

After simulating the data we introduce three types of missingness. To simulate missing completely at random we simply randomly make the fatigue scores missing with all the fatigue scores having the same probability of being missing. To introduce missing at random we make the probability of a patient's fatigue score missing depend on the person's age. The higher the age, the greater the probability of having a missing value for the fatigue score. Lastly, for not missing at random we made the probability of having a missing value depend on the fatigues scores themselves with higher fatigue scores being more likely to be missing. We were also interested in testing how the two methods would perform if the MNAR was strong versus a weak relationship. For this, we simply linked more strongly higher probabilities of having missing fatigue scores to the higher fatigue scores. To introduce these three types of missing data we had three functions. By changing the probability of having missing values we could control what percentage of the data was missing. Hence, we were able to created data sets with 20%, 30%, 40% and 60% missing data for the variable of interest. For the reader's reference, sample code is provided in the appendix.

2.4 The Mixed Effect Model

The mixed-effects model is used for modeling data with repeated measures. Hence, can be used to model repeated measures over time. The mixed effects model is:

$$\mathbf{y} = \mathbf{X}\beta + \mathbf{Z}\mathbf{u} + \varepsilon$$

where:

- $\mathbf{y}$ is the vector of the variables of interest, with $E(\mathbf{y}) = \mathbf{X}\beta$;
- β is the vector of the coefficients of the fixed effects;
- u is the vector of random effects where $u \sim N(0, G)$;
- ε is the vector of random errors with $\varepsilon \sim N(0, R)$;
- with $Cov(u, \varepsilon) = 0$;
- $\mathbf{X}$ and $\mathbf{Z}$ are design matrices, where $\mathbf{X}$ is for the fixed effects and $\mathbf{Z}$ is for the random effects.

To model the simulated longitudinal data using mixed-effects model we used the R package lme4.

For our mixed-effect model, our response variable was the patient's fatigue score, which was treated as a continuous variable.The patients were treated as random effects over time. All the other covariates were treated as fixed effects: Cancer stage, cancer type, age, gender and the time points. The R code we used for our model was:

$$lmer(FACTF\_Total \sim Timepoint + Age + Sex + CaDiagnosis\_Generalized + Stage\_Generalized + (Timepoint|ID))$$

Where FACTF_Total is the fatigue score, CaDiagnosis_Generalized is the type of cancer and Stage_Generalized is the cancer stage.

2.5 Obtaining Estimates and Confidence Intervals from the 100 simulated data sets

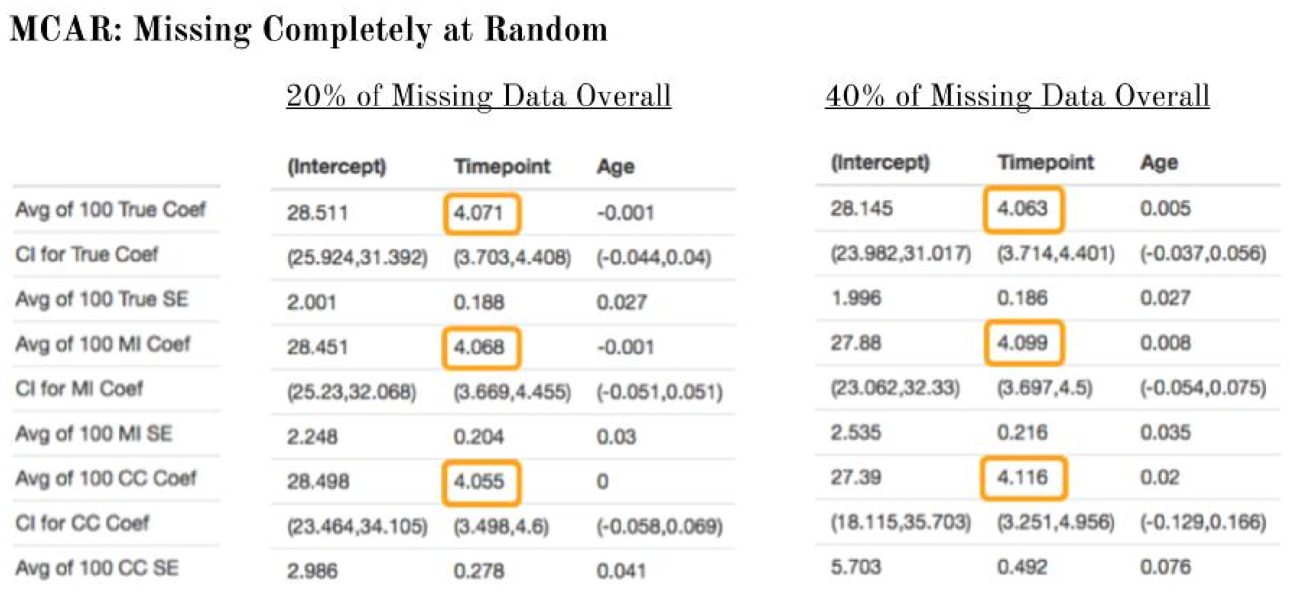
Once we simulated the 100 longitudinal data sets and then added missing data to them, we applied the complete case analysis and then multiple imputation by chained equations on the 100 data sets individually. Once the missing data was taken care of we applied the same mixed-effects model getting 100 estimates for each of the missing data methods. Then, to do the final step we took the average of the 100 estimates for the complete case analysis and we called this the final complete case point estimate. Similarly, we obtained the final MICE point estimate. Now, since we had the complete data sets we again fitted the mixed-effects model to the complete data sets obtaining 100 "true" estimates. Then again, simply took the average of the 100 "true" estimates to calculate the final "true" point estimate.

To create the 95% confidence interval of the MICE estimates of mixed-effects model we computed the 2.5% and the 97.5% quantiles of the 100 estimates we had obtained from the 100 simulations, with 2.5% and 97.5% quantiles being the lower and upper bounds of the confidence interval respectively. The 95% confidence intervals of the complete case analysis estimates and the "true" estimates, were also constructed in the same manner.

3 Results

3.1 Comparing Multiple Imputation with Complete Case Analysis for different types, and over different percentages of missing data.

We first present the results for 20% and 30% missing data with the missingness being missing completely at random, the most benign type.



MCAR: Missing Completely at Random

	20% of Missing Data Overall			40% of Missing Data Overall		
	(Intercept)	Timepoint	Age	(Intercept)	Timepoint	Age
Avg of 100 True Coef	28.511	4.071	-0.001	28.145	4.063	0.005
CI for True Coef	(25.924,31.392)	(3.703,4.408)	(-0.044,0.04)	(23.982,31.017)	(3.714,4.401)	(-0.037,0.056)
Avg of 100 True SE	2.001	0.188	0.027	1.996	0.186	0.027
Avg of 100 MI Coef	28.451	4.068	-0.001	27.88	4.099	0.008
CI for MI Coef	(25.23,32.068)	(3.669,4.455)	(-0.051,0.051)	(23.062,32.33)	(3.697,4.5)	(-0.054,0.075)
Avg of 100 MI SE	2.248	0.204	0.03	2.535	0.216	0.035
Avg of 100 CC Coef	28.498	4.055	0	27.39	4.116	0.02
CI for CC Coef	(23.464,34.105)	(3.498,4.6)	(-0.058,0.069)	(18.115,35.703)	(3.251,4.956)	(-0.129,0.166)
Avg of 100 CC SE	2.986	0.278	0.041	5.703	0.492	0.076

Figure 5. Average of the 100 estimates of the coefficients obtained from the 100 simulated data for missing completely at random data. In this we are comparing the performance of complete case analysis and multiple imputation at the level of 20% and 40% missing data.

From the above, one can see that for 20% missing data the average of the 100 true coefficients (4.071), the average of the 100 MI coefficients (4.068), and the average of the 100 CC coefficients (4.055), are all very close (the estimates highlighted in orange). The same is true when the missing data is 40%.

We shall now plot the results, drawing the 95% confidence intervals as well and will also plot the results for 30% missing data.

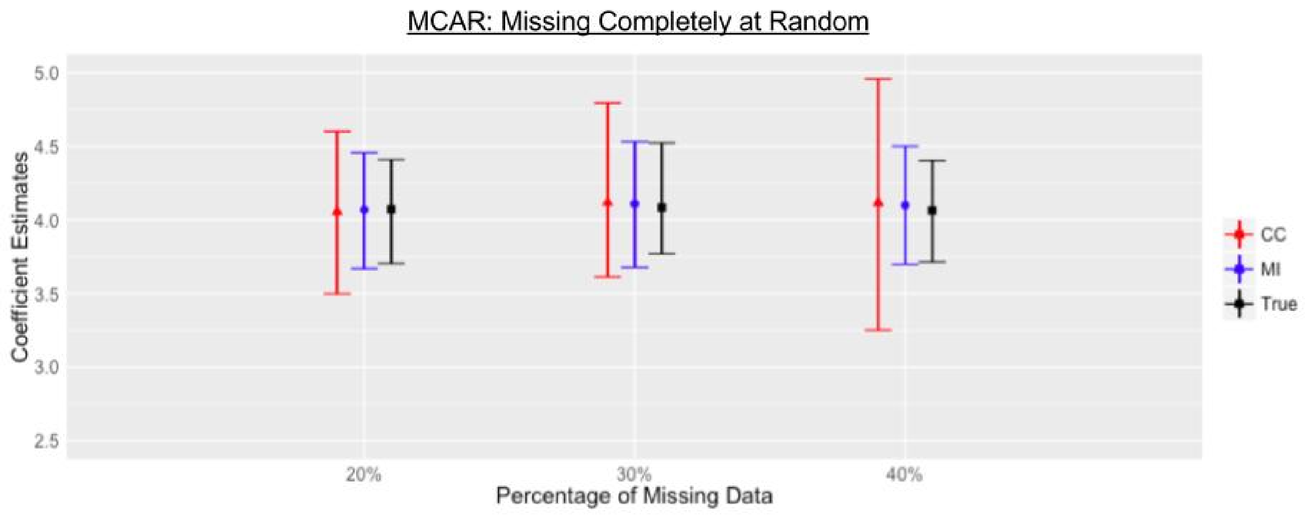


Figure 6. Performance of complete cases analysis and multiple imputation by chained equations for missing complete at random data.

One can see that the point estimates of MICE and CC are almost exactly the same as the true coefficient estimate. Further, their confidence intervals overlap, with complete case analysis having larger confidence intervals than multiple imputation, especially when 40% of the data is missing. Hence, one clear advantage of multiple imputation over CC is that it is more accurate, especialy when missing data is high.

We repeated the above but with the missing data now being missing at random (MAR), where the missingness of the fatigue scores depend on the age of the person.

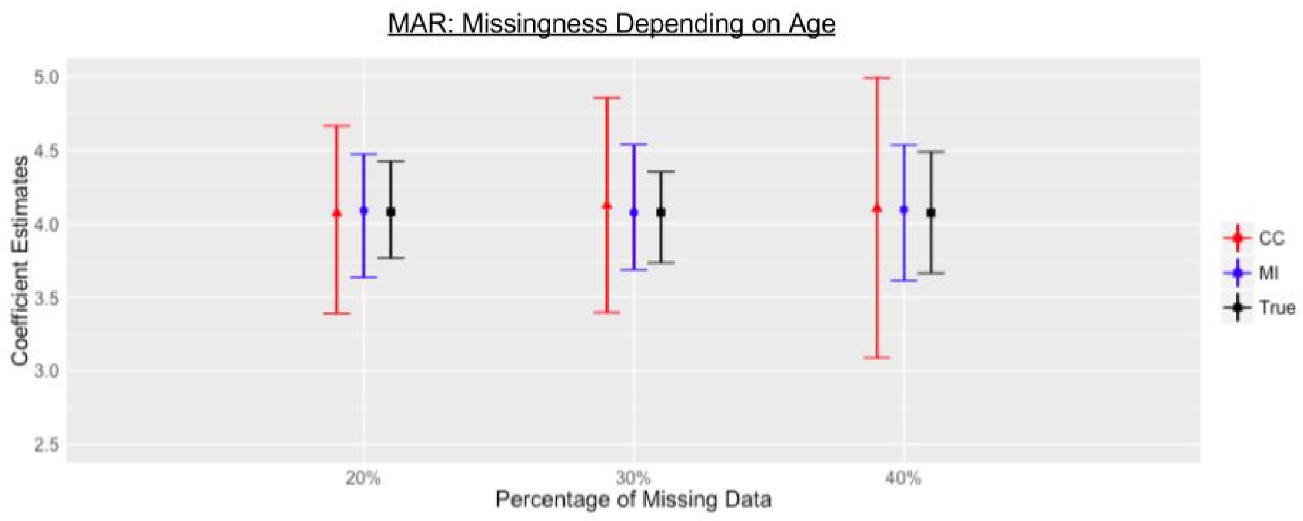


Figure 7. Performance of complete case analysis and multiple imputation with missing at random data.

Here, we see, perhaps a bit surprisingly, that even though the data is missing at random, complete case analysis performs as well as multiple imputation. Theoretocally, complete case analysis should be worse off than multiple imputation, as it should only be able to handle data that is missing completely at random.

Now, we introduce missing data that is not missing at random (NMAR), with the probability of the fatigue score missing depending on the fatigue score itself. We designed this so that higher fatigue scores were more likely to be missing. After running the simulation 100 times we plot the results below.

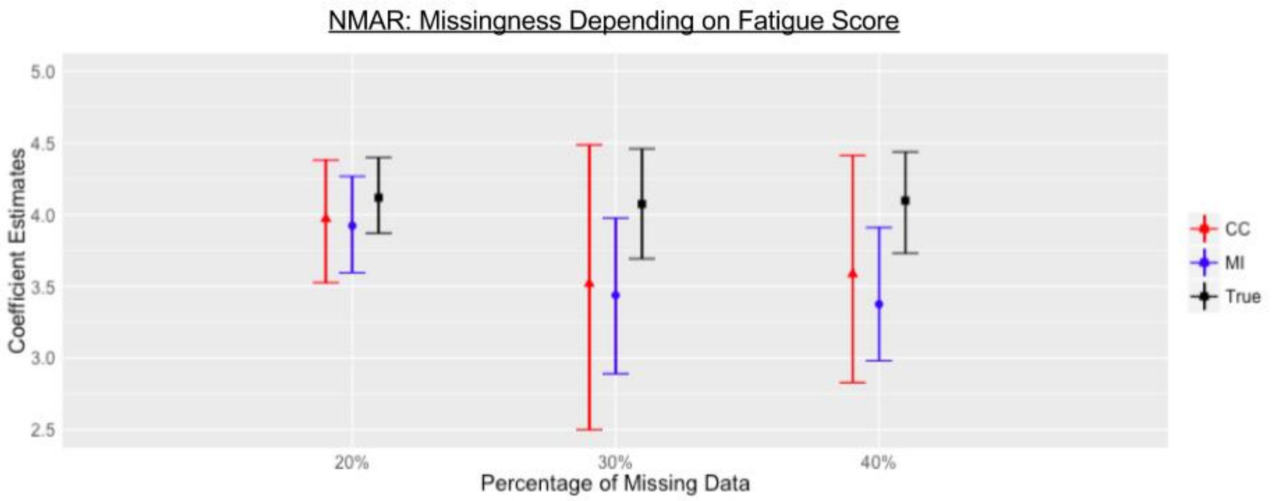


Figure 8. Performance of complete case analysis and multiple imputation for not missing at random data at different percentages of missing data.

We see that even for 20% missing data, the point estimates of the two missing data methods are not as accurate as before. Then, when the missingness increases to 30% and 40% the point estimates are way off and multiple imputation's confidence intervals do not even contain the true point estimates of the coefficient.

3.2 Comparing the performance of Complete Cases and Multiple Imputation for weak vs. strong Not Missing at Random data and evaluating the performance of MI for 60% of missing data.

We will now see how complete case analysis and multiple imputation perform when there is a strong MNAR pattern for the missing data and when there is a weak one. Both weak and strong MNAR are for 40% missing data. We have plotted the results below in Figure 9.

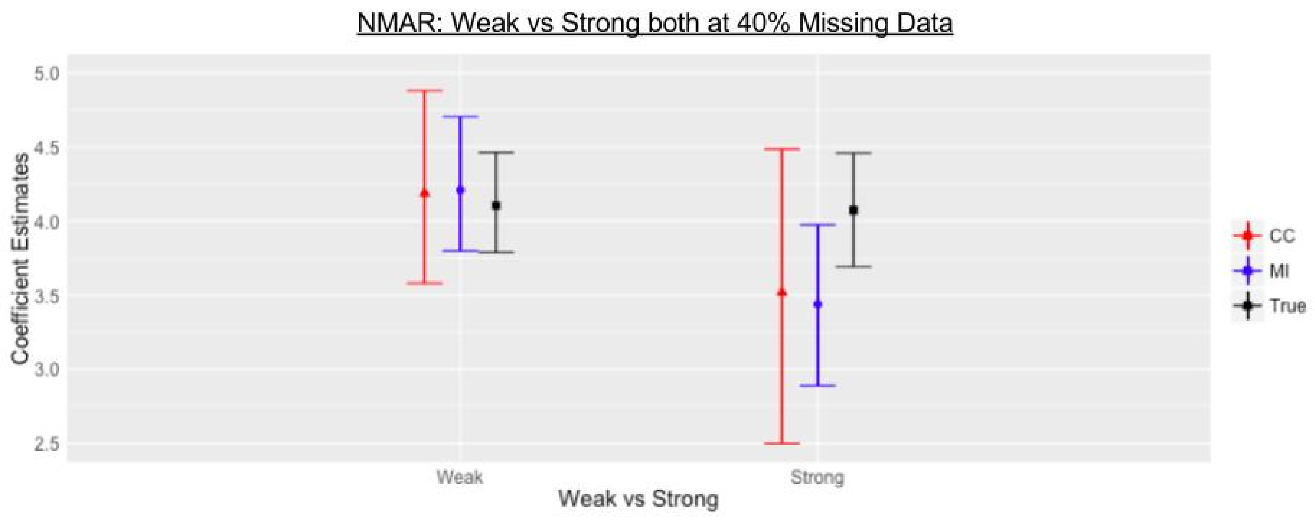


Figure 9. Performance of complete case analysis and multiple imputation for weak and strong NMAR with 40% missing data.

We see that when the MNAR is weak, the performance of complete case analysis and multiple imputation are both very accurate even with 40% missing data. However, both perform poorly when there is a strong MNAR missing data pattern, with complete case analysis doing slightly better that multiple imputation as its confidence interval captures the true coefficients point estimate and its point estimate is slightly closer to the true estimate than that of the multiple imputation.

We tried to compare the performance of multiple imputation and complete case analysis for 60% missing data. However, there were not enough complete cases to perform a complete case analysis. Thus, we only looked at how well multiple imputation performs for the different types of missing data, when 60% of the data is missing.

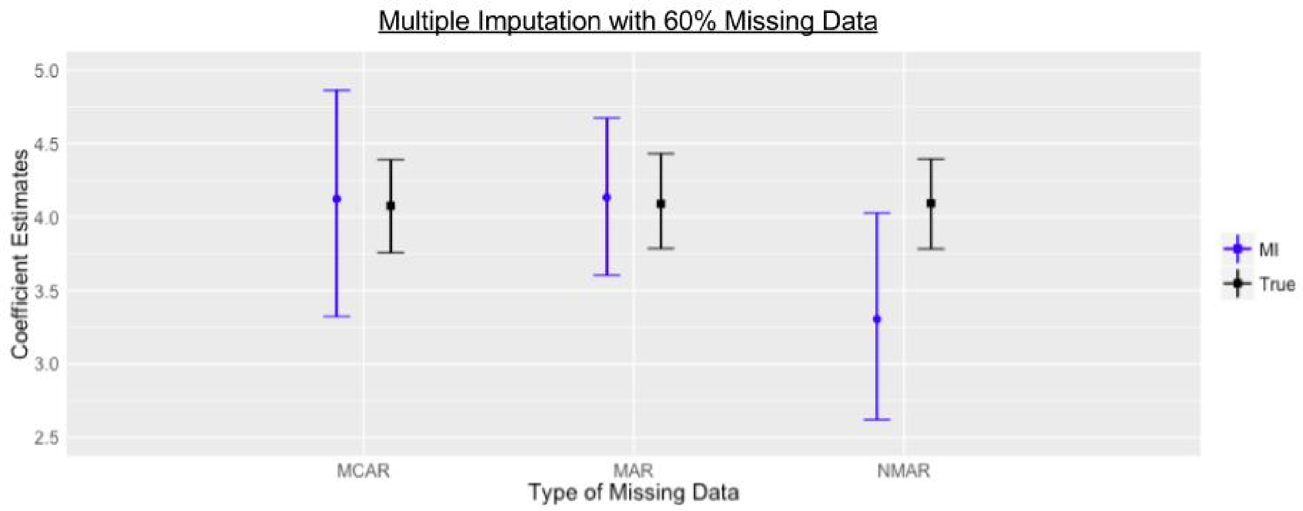


Figure 10. Performance of multiple imputation for MCAR, MAR and NMAR missing data at 60% missingness.

From Figure 10 we see that even when the missing data is very high, 60% as long as the missing data is missing completely at random or missing at random then multiple imputation performs very well. However, when the missing data is not missing at random then, multiple imputation does not perform very well.

4 Discussion

Firstly, one of the most interesting finding we found was that for MAR, in terms of point estimate of the coefficients, complete case analysis and multiple imputation performed equally well (see Figure 7). This is surprising, since theoretically, multiple imputation is able to handle MAR but complete case analysis is not. However, when we simulated the data we made all the variables uncorrelated, except time point and fatigue scores, and gender and cancer type. Hence, as the algorithm of MICE uses to its advantage the correlation between variables, and there was no such relationship for MICE to exploit, both methods yielded similar results for MAR data. This result does make sense and we found one other paper by Mukaka et. al 2016 where researchers using real world data, found CC and MI to perform equally well[8].

Further, for missing data under 50% and with the MCAR and MAR conditions, we found that both MICE and CC confidence intervals capture the true estimate with MICE's point estimate being very slightly closer to the true estimate compared to CC (see Figure 6 and Figure 7, respectively). Also, the CC coefficient estimates have much larger variance compared to MICE and hence much larger confidence intervals. And from our simulation study we can see that for MCAR and MAR, both MICE and CC give unbiased estimates. Therefore, MICE would has smaller mse compared to CC making MICE the better method inspite of the point estimates of the two methods having similarly good performance.

Now for NMAR, both MICE and CC estimates are biased and the bias becomes progressively worse as the percentage of missing data increases (see Figure 8). However, when the missing data is 20% both MICE and CC perform reasonably well as they are able to capture the true estimate in their confidence intervals. Now when the missing data percentage increases, the CC estimate's confidence interval, which is much larger compared to MICE's, is able to capture the true estimate of the coefficient, whereas MICE's does not (although, MICE's confidence interval does overlap with that of the true estimate's confidence interval). Also, the point estimate for the complete cases was slightly closer to the true estimate than that of MICE. Therefore, both methods do not perform very well when the data is NMAR. Though, NMAR has less of an impact when there is little missing data.

Further, we also wanted to see how MICE and CC would perform when the NMAR was weak. By NMAR being weak, we mean that the probability for the Y value to be missing is weakly related to the value of Y (the varaible of interst, fatigue score in our case). For this we kept the missing data at 40%. From Figure 9 we see that MICE actually performs very reasonably when NMAR is weak and performs very badly when it is strong. In fact CC also happens to perform reasonably well. So even if the missing data is NMAR, but there is a weak relationship between the value of the variable of interest and its probability of missing, MICE and CC perform quite well.

Now, when missing data is over 50%, we did not even have enough complete cases to perform a complete case analysis. Where we can see a clear advantage for MICE. Further, MICE was even able to capture the true estimate even at 60% missing data when the data was MCAR or MAR with the point estimates being very close to the true estimate, as seen in Figure 10. It did not however, capture the true estimate in its confidence interval when the missing data was MNAR. This said, it is actually not advised to perform imputation when missing data is over 50%. However, as can be seen, if the missing data is MCAR or MAR and there can not be made any improvement on the missingness of the data set, then for practical reasons, one could use MICE. These results are indeed very intersting. Especially, the fact that MICE predicts the true point estimate accurately with little to no bias for MCAR and MAR. Also, our results show that MICE is not able to handle NMAR data with 60% data, as one would expect.

Final Conclusion: Both methods perform equally well when missing data is less than 50% with MICE being more precise (smaller CI). However, when the missing data is MNAR, complete case analysis somewhat technically outperforms MICE due to its larger confidence interval. For over 50% missing data, there might not be enough complete cases to perform a complete case analysis. MICE, in our particular case, performs very well even when missing data is over 50% as long as the missing data is not MNAR.

References

1. Allison, P. D. Handling missing data by maximum likelihood. In *SAS global forum*, vol. 2012 (Statistical Horizons, Havenford, PA, 2012).

2. Dong, Y. & Peng, C.-Y. J. Principled missing data methods for researchers. *SpringerPlus* **2**, 222 (2013).

3. Yuan, Y. C. Multiple imputation for missing data: Concepts and new development (version 9.0). *SAS Institute Inc, Rockville, MD* **49**, 1–11 (2010).

4. Sterne, J. A. C. *et al.* Multiple imputation for missing data in epidemiological and clinical research: potential and pitfalls. *BMJ* **338** (2009). URL https://www.bmj.com/content/338/bmj.b2393. DOI 10.1136/bmj.b2393. https://www.bmj.com/content.

5. Azur, M. J., Stuart, E. A., Frangakis, C. & Leaf, P. J. Multiple imputation by chained equations: what is it and how does it work? *International Journal of Methods in Psychiatric Research* **20**, 40–49 (2011). URL https://onlinelibrary.wiley.com/doi/abs/10.1002/mpr.329. DOI 10.1002/mpr.329. https://onlinelibrary.wiley.com/doi/pdf/10.1002/mpr.329.

6. Morris, T. P., White, I. R. & Royston, P. Tuning multiple imputation by predictive mean matching and local residual draws. *BMC medical research methodology* **14**, 75 (2014).

7. Acaster, S. *et al.* Qualitative and quantitative validation of the facit-fatigue scale in iron deficiency anemia. *Health and quality of life outcomes* **13**, 60 (2015).

8. Mukaka, M. *et al.* Is using multiple imputation better than complete case analysis for estimating a prevalence (risk) difference in randomized controlled trials when binary outcome observations are missing? *Trials* **17**, 341 (2016).

Author contributions statement

F.K.M. conceived the research question, methodology and experiments. F.K.M., J.N. and H.H. conducted the experiments and the analysis. The final results and analysis were reviewed by all the authors F.K.M., J.N., H.H. and N.T.. F.K.M. wrote and reviewed the manuscript.

Additional information

Competing financial interests

The authors declare no competing interests.

Source Code

To obtain the complete source code for this study, please contact the authors directly.

Appendix

Sample Code

First we will provide sample code of the script and then code for the helper functions.

```r
###################### Script Code #####################################

#### First simulate the data
sim1 = sim_data(dataset, p_female=.8, p0=p0, p1=p1, p2=p2, p3=p3, p4=p4)
# Look at the correlations among the variables of the correlated data
MM <- cor(sim1, use = "pairwise.complete.obs")
corrplot(MM, method="circle")
corrplot.mixed(MM)

# MCAR 20% overall missing data
p_absent = c(.4, .05, .05)
p_missing = c(.2, .2, .2, .1)
# Simulate missing data according to the different p's for MCAR
sim1_data = mcar_final(sim1, p_absent, p = p_missing)
# Find total Missing % and Missing % at Each Timepoint for MCAR
perc_miss_vec(sim1_data)
perc_miss(sim1_data)
# See how the results are affected for the MCAR
Test_MCAR_20per = simulation_mcar(dataset, n.sim = n_sim, m = 5, p_absent = p_absent,
                                  p = p_missing, p_female=.8, p0=p0, p1=p1, p2=p2,
                                  p3=p3, p4=p4)
Test_MCAR_20


# MAR 30% overall missing data
p_absent = c(.2, .1, .05)
p_missing = c(.1, .1, .1, .4)
# Simulate missing data according to the p's for MAR
sim1_data = mar_final(sim1, p_absent = p_absent, p_missing, pattern_age)
# Find total Missing % and Missing % at Each Timepoint for MAR
perc_miss_vec_results = perc_miss_vec(sim1_data); perc_miss_vec_results
perc_miss(sim1_data)
# See how the results are affected for the MAR
Test_MAR_30perc = simulation_mar(dataset, n.sim = n_sim, m = 5, p_absent = p_absent,
                                 p_missing = p_missing, pattern = pattern_age,
                                 p_female=.8, p0=p0, p1=p1, p2=p2, p3=p3, p4=p4)
Test_MAR_30perc


# Strong NMAR 20% overall Missing data
p_absent =  c(.3, .05, .05)
p_missing = c(.001, .001, .05, .3)
# Simulate missing data according to the p's for NMAR
sim1_data = nmar_final(sim1, p_absent, p_missing, pattern)
```

```r
# Find total Missing % and Missing % at Each Timepoint for MAR
perc_miss(sim1_data)
perc_miss_vec(sim1_data)
# See how the results are affected for the MAR
Test_strong_NMAR_20per = simulation_nmar(dataset, n.sim = n_sim, m = 5, p_absent=p_absent,
                                         p_missing, pattern = pattern, p_female=.8, p0=p0,
                                         p1=p1, p2=p2, p3=p3, p4=p4)
Test_NMAR_20per


# Weak NMAR 30% overall Missing data
p_absent =  c(.3, .1, .1)
p_missing = c(.2, .25, .3, .4)
# Simulate missing data according to the p's for NMAR
sim1_data = nmar_final(sim1, p_absent, p_missing, pattern)
# Find total Missing % and Missing % at Each Timepoint for MAR
perc_miss(sim1_data)
perc_miss_vec(sim1_data)
# See how the results are affected for the MAR
Test_weak_NMAR_30per = simulation_nmar(dataset, n.sim = n_sim, m = 5, p_absent = p_absent,
                                       p_missing, pattern = pattern, p_female=.8, p0=p0,
                                       p1=p1, p2=p2, p3=p3, p4=p4)
Test_NMAR_30per


###################### Helper Functions #####################################

######################## Simulating the data #################################

######
###### Function that simulates the data
######

sim_data = function(data, p_female=.8, p0=rep(1,52), p1=rep(1,52),
                    p2=rep(1,52), p3=rep(1,52), p4=rep(1,52)){

  f_vec = 0:52

  # female is 1 and male is 0
  # breast = 1, Head and neck = 2, Leukemia = 3, Prostrate = 4, Lung = 5
  #cancer = c("Breast", "Head and Neck", "Leukemia", "Prostrate", "Lung")

  cancer = c(1, 2, 3, 4, 5)

  j = 1
  while (j < nrow(data) ){

    a1 = sample(c(1,0), size = 1, prob = c(p_female, 1-p_female))
    a2 = round(rnorm(n = 1, mean = 54.93, sd = 9.829))
    a3 = if (a1 == 1) sample(cancer, size = 1, prob = c(0.7, .10, .15, 0, .05) )
         else  sample(cancer, size = 1, prob = c(0, .10, .20, .60, .10) )
    a4 = sample(1:4, size = 1, prob = c(43, 49, 59, 19) )

    a = c(a1, a2, a3, a4)
    data[j, c(3, 4, 5, 6)] = a
    data[j+1, c(3, 4, 5, 6)] = a
    data[j+2, c(3, 4, 5, 6)] = a
    data[j+3, c(3, 4, 5, 6)] = a
    data[j+4, c(3, 4, 5, 6)] = a

    data[j, ncol(data)] = sample(f_vec, size = 1, prob = p0)
    data[j+1, ncol(data)] = sample(f_vec, size = 1, prob = p1)
    data[j+2, ncol(data)] = sample(f_vec, size = 1, prob = p2)
    data[j+3, ncol(data)] = sample(f_vec, size = 1, prob = p3)
    data[j+4, ncol(data)] = sample(f_vec, size = 1, prob = p4)

    j = j + 5

  }

  data
}

# # Test
# f_vec = 0:52
# p0 = dnorm(f_vec, mean = 27, sd = 10)
# p1 = dnorm(f_vec, mean = 35, sd = 9)
# p2 = dnorm(f_vec, mean = 42, sd = 10)
# p3 = dnorm(f_vec, mean = 45, sd = 10)
# p4 = dnorm(f_vec, mean = 47, sd = 10)
# sim1 = sim_data(data = simdata, p_female = .83, p0=p0, p1=p1, p2=p2, p3=p3, p4=p4)


###################### Introducing missingness to data. #######################


######
###### Function that creates MCAR data at the rate of p_i% at each Timepoint i.
######

mcar_final = function(data, p_absent, p, seed){

  n = nrow(data)

  for (i in 1:n){

    f_score = data[i,ncol(data)]
    p1 = 0
    p2 = 0
    p3 = 0


    if (data[i, ]$Timepoint == 1){
      data[i,ncol(data)]=sample(c(NA, f_score), size = 1, prob = c(p[1], 1-p[1]))
    }

    if (data[i, ]$Timepoint == 2){

      if(is.na(data$FACTF_Total[i-1])){
        p1 = p_absent[1]
      }

      data[i,ncol(data)]=sample(c(NA, f_score), size = 1, prob = c(p[2] + p1 , 1 - (p1+p[2])
    }

    if (data[i, ]$Timepoint == 3){

      if(is.na(data$FACTF_Total[i-1])){
        p1 = p_absent[1]
      }

      if(is.na(data$FACTF_Total[i-2])){
        p2 = p_absent[2]
      }
      pp = p1 + p2

      data[i,ncol(data)]=sample(c(NA, f_score), size =  1, prob = c(pp+p[3], 1-(pp+p[3])))
    }

    if (data[i, ]$Timepoint == 4){


      if(is.na(data$FACTF_Total[i-1])){
        p1 = p_absent[1]
      }

      if(is.na(data$FACTF_Total[i-2])){
        p2 = p_absent[2]
      }

      if(is.na(data$FACTF_Total[i-3])){
        p3 = p_absent[3]
      }

      pp = p1 + p2 + p3


      data[i,ncol(data)]=sample(c(NA, f_score), size = 1, prob = c(pp+p[4], 1-(pp+p[4])))
    }

  }
  data
}

# Test
# vecdata = mcar_final(data = sim2, p_absent = c(.1, .1, .1), p = c(.4, .4, .4, .4) )
# perc_miss_vec(vecdata)
# perc_miss(vecdata)



######
###### Final Function that creates MAR data at the rate of p%
######
```

```r
# Creating Missing at Random Condition. Assumption: the missingness is associated with Age
mar_final<-function(df, p_absent, p_missing, pattern){
  for(i in 1:nrow(df)){

    if(df$Timepoint[i] != 0) {

      pp = 0
      p1 = 0
      p2 = 0
      p3 = 0

      if (data[i, ]$Timepoint == 2){

        if(is.na(data$FACTF_Total[i-1])){
          pp = p_absent[1]
        }
      }

      if (data[i, ]$Timepoint == 3){

        if(is.na(data$FACTF_Total[i-1])){
          p1 = p_absent[1]
        }

        if(is.na(data$FACTF_Total[i-2])){
          p2 = p_absent[2]
        }
        pp = p1 + p2
      }

      if (data[i, ]$Timepoint == 4){

        if(is.na(data$FACTF_Total[i-1])){
          p1 = p_absent[1]
        }

        if(is.na(data$FACTF_Total[i-2])){
          p2 = p_absent[2]
        }

        if(is.na(data$FACTF_Total[i-3])){
          p3 = p_absent[3]
        }

        pp = p1 + p2 + p3
      }

      if(pattern[1,1]<=df$Age[i] && df$Age[i]<=pattern[1,2]){
        # Removing the data for for certain chances if the 0<= fatigue score <=12
        df$FACTF_Total[i]<-sample(c(NA, df$FACTF_Total[i]), size = 1,
                                  prob = c(p_missing[1] + pp , 1-(pp + p_missing[1])))
        next
      }
      if(pattern[2,1]<=df$Age[i] && df$Age[i]<=pattern[2,2]){
        # Removing the data for for certain chances if the 13<= fatigue score <=25
        df$FACTF_Total[i]<-sample(c(NA, df$FACTF_Total[i]), size = 1,
                                  prob = c(p_missing[2] + pp, 1 -(pp + p_missing[2])))
        next
      }
      if(pattern[3,1]<=df$Age[i] && df$Age[i]<=pattern[3,2]){
        # Removing the data for for certain chances if the 26<= fatigue score <=38
        df$FACTF_Total[i]<-sample(c(NA, df$FACTF_Total[i]), size = 1,
                                  prob = c(p_missing[3] + pp , 1-(pp + p_missing[3])))
        next
      }
      if(pattern[4,1]<=df$Age[i] && df$Age[i]<=pattern[4,2]){
        # Removing the data for for certain chances if the 39<= fatigue score <=52
        df$FACTF_Total[i]<-sample(c(NA, df$FACTF_Total[i]), size = 1,
                                  prob = c(p_missing[4] + pp , 1-(pp + p_missing[4])))
      }

    }

  }
  df
}

## Test
# mar_data1 = mar_final(sim2)
# perc_miss_vec(mar_data1)
# perc_miss(mar_data1)


######
###### Function that creates Not Missing at Random data.
######

# Creating Not Missing at Random Condition.
# Assumption: the missingness is associated with Total Fatigue Score
nmar_final = function(data, p_absent, p_missing,  pattern){


  n = nrow(data)

  for (i in 1:n){

    p1 = 0
    p2 = 0
    p3 = 0
    pp = 0

    if (data[i, ]$Timepoint == 2){

      if(is.na(data$FACTF_Total[i-1])){
        p1 = p_absent[1]
      }

      pp = p1
    }

    if (data[i, ]$Timepoint == 3){

      if(is.na(data$FACTF_Total[i-1])){
        p1 = p_absent[1]
      }

      if(is.na(data$FACTF_Total[i-2])){
        p2 = p_absent[2]
      }

      pp = p1 + p2



    }

    if (data[i, ]$Timepoint == 4){


      if(is.na(data$FACTF_Total[i-1])){
        p1 = p_absent[1]
      }

      if(is.na(data$FACTF_Total[i-2])){
        p2 = p_absent[2]
      }

      if(is.na(data$FACTF_Total[i-3])){
        p3 = p_absent[3]
      }

      pp = p1 + p2 + p3

    }

    if(data$Timepoint[i] != 0){

      if(pattern[1,1]<=data$FACTF_Total[i] && data$FACTF_Total[i]<=pattern[1,2]){
        # Removing the data for for certain chances if the 0<= fatigue score <=12
        data$FACTF_Total[i]<-sample(c(data$FACTF_Total[i],NA), size = 1,
                                    prob = c(1-(pp+ p_missing[1]), pp + p_missing[1]))
        next
      }
      if(pattern[2,1]<=data$FACTF_Total[i] && data$FACTF_Total[i]<=pattern[2,2]){
        # Removing the data for for certain chances if the 13<= fatigue score <=25
        data$FACTF_Total[i]<-sample(c(data$FACTF_Total[i],NA), size = 1,
                                    prob = c(1-( pp + p_missing[2]), pp + p_missing[2]))
        next
      }
      if(pattern[3,1]<=data$FACTF_Total[i] && data$FACTF_Total[i]<=pattern[3,2]){
        # Removing the data for for certain chances if the 26<= fatigue score <=38
        data$FACTF_Total[i]<-sample(c(data$FACTF_Total[i],NA), size = 1,
                                    prob = c(1-(pp + p_missing[3]), pp+ p_missing[3]))
```

```r
        next
      }
      if(pattern[4,1]<=data$FACTF_Total[i] && data$FACTF_Total[i]<=pattern[4,2]){
        # Removing the data for for certain chances if the 39<= fatigue score <=52
        data$FACTF_Total[i]<-sample(c(data$FACTF_Total[i],NA), size = 1, prob = c(1-(pp + p_m
      }
    }
  }
  data
}

# Test
# vecdata = nmar_final(data = sim2, p_absent = c(.1, .1, .1), p_missing = c(.4, .5, .5, .5),
# perc_miss_vec(vecdata)
# perc_miss(vecdata)


################# MI and MIXED-EFFECTS models #######################################


# Function That combines everything for
#MCAR

simulation_mcar = function(dataset, n.sim = 10, m = 5 , p_absent, p, p_female=.8,
                           p0=rep(1,52), p1=rep(1,52), p2=rep(1,52), p3=rep(1,52),
                           p4=rep(1,52)){

  true_coef = matrix(nrow = n.sim, ncol = 6)
  true_se = matrix(nrow = n.sim, ncol = 6)
  mi_est_coef = matrix(nrow = n.sim, ncol = 6)
  mi_est_se = matrix(nrow = n.sim, ncol = 6)
  cc_est_coef = matrix(nrow = n.sim, ncol = 6)
  cc_est_se = matrix(nrow = n.sim, ncol = 6)

  for (i in 1:n.sim){

    # Simulating the data
    sim1 = sim_data(dataset, p_female = p_female, p0=p0, p1=p1, p2=p2, p3=p3, p4=p4)

    # Calculating true Coef and their se's from the simulated (complete) data

    mixed_model = lmer(FACTF_Total ~ Timepoint + Age + Sex + CaDiagnosis_Generalized +
                         Stage_Generalized + (Timepoint|ID), data = sim1)

    summary(mixed_model)

    Vcov <- vcov(mixed_model, useScale = FALSE)
    betas <- fixef(mixed_model)
    se <- sqrt(diag(Vcov))

    true_coef[i, ] = betas
    true_se[i, ] = se

    # Estimating Coef from the imputed data
    vecdata = mcar_final(sim1, p_absent = p_absent, p = p) # Creating the missing data

    imputed_mi = mice(vecdata, m = m) # Imputing the missing data

    mixed_model_imp <- with(imputed_mi,lmer(FACTF_Total ~ Timepoint + Age + Sex +
                            CaDiagnosis_Generalized + Stage_Generalized + (Timepoint|ID)))

    results_mi = summary(pool(mixed_model_imp))

    # confint(mixed_model_imp)

    mi_est_coef[i,] = results_mi[, 1 ]
    mi_est_se[i, ] = results_mi[, 2 ]

#     # Checking if True coef's are within CI's for Multiple Imputation
#
#     for (j in 1:6){
#       mi_count[i,j] = if (results_mi[,6][j] <= betas[j] && betas[j] <= results_mi[, 7][j])
#     }

    imputed_cc = create_completecasedf(vecdata) # Imputing the missing data

    mixed_model_imp <- lmer(FACTF_Total ~ Timepoint + Age + Sex + CaDiagnosis_Generalized +
                              Stage_Generalized + (Timepoint|ID), data = imputed_cc)
    results_cc = summary(mixed_model_imp)

    cc_est_coef[i,] = coef(results_cc)[,1]
    cc_est_se[i, ] = coef(results_cc)[,2]

#     # Checking if True coef's are within CI's for Complete Cases
#
#     for (j in 1:6){
#       cc_count[i,j] = if (coef(results_cc)[, 6][j] <= betas[j] && betas[j] <= coef(results_
#     }

    if (i==1){
      name = names(betas)
    }
    print(paste("num of Sims = ", i))

  }

    qtrue1 = quantile(true_coef[,1],
                      c(0.025, 0.975)); qtrue2 = quantile(true_coef[,2], c(0.025, 0.975))
    qtrue3 = quantile(true_coef[,3],
                      c(0.025, 0.975)); qtrue4 = quantile(true_coef[,4], c(0.025, 0.975))
    qtrue5 = quantile(true_coef[,5],
                      c(0.025, 0.975)); qtrue6 = quantile(true_coef[,6], c(0.025, 0.975))

    qtrue_final = cbind(qtrue1, qtrue2, qtrue3, qtrue4, qtrue5, qtrue6)

    qmi_est1 = quantile(mi_est_coef[,1], c(0.025, 0.975))
    qmi_est2 = quantile(mi_est_coef[,2], c(0.025, 0.975))
    qmi_est3 = quantile(mi_est_coef[,3], c(0.025, 0.975))
    qmi_est4 = quantile(mi_est_coef[,4], c(0.025, 0.975))
    qmi_est5 = quantile(mi_est_coef[,5], c(0.025, 0.975))
    qmi_est6 = quantile(mi_est_coef[,6], c(0.025, 0.975))

    qmi_est_final = cbind(qmi_est1, qmi_est2, qmi_est3, qmi_est4, qmi_est5, qmi_est6)

    qcc_est1 = quantile(cc_est_coef[,1], c(0.025, 0.975))
    qcc_est2 = quantile(cc_est_coef[,2], c(0.025, 0.975))
    qcc_est3 = quantile(cc_est_coef[,3], c(0.025, 0.975))
    qcc_est4 = quantile(cc_est_coef[,4], c(0.025, 0.975))
    qcc_est5 = quantile(cc_est_coef[,5], c(0.025, 0.975))
    qcc_est6 = quantile(cc_est_coef[,6], c(0.025, 0.975))

    qcc_est_final = cbind(qcc_est1, qcc_est2, qcc_est3, qcc_est4, qcc_est5, qcc_est6)


    result_matrix = as.data.frame(rbind(round(colMeans(true_coef), digits = 3),
                                    paste("(",round(qtrue_final[1,], digits = 3),",",
                                      round(qtrue_final[2,], digits = 3),")", sep = ""),
                                      round(colMeans(true_se), digits = 3),
                                      round(colMeans(mi_est_coef), digits = 3),
                                    paste("(",round(qmi_est_final[1,], digits = 3),",",
                                     round(qmi_est_final[2,], digits = 3),")", sep = ""),
                                      round(colMeans(mi_est_se), digits = 3),
                                      round(colMeans(cc_est_coef), digits = 3),
                                    paste("(",round(qcc_est_final[1,], digits = 3),",",
                                     round(qcc_est_final[2,], digits = 3),")", sep = ""),
                                      round(colMeans(cc_est_se), digits = 3)))

    dimnames(result_matrix) = list(c(paste("Avg of", n.sim, "True Coef"),
                                     "CI for True Coef", paste("Avg of", n.sim, "True SE"),
                                   paste("Avg of", n.sim, "MI Coef"), "CI for MI Coef",
                                   paste("Avg of", n.sim, "MI SE"),
                                   paste("Avg of", n.sim, "CC Coef"), "CI for CC Coef",
                                   paste("Avg of", n.sim, "CC SE")), name)

    print(paste("total num of Sims = ", i))
    kable(result_matrix)
  }


###################################################

# Function That combines everything for
# MAR

simulation_mar = function(dataset, n.sim = 10, m = 5, p_absent, p_missing, pattern,
                          p_female=.8, p0=rep(1,52), p1=rep(1,52), p2=rep(1,52),
                          p3=rep(1,52), p4=rep(1,52)) {

  true_coef = matrix(nrow = n.sim, ncol = 6)
  true_se = matrix(nrow = n.sim, ncol = 6)
  mi_est_coef = matrix(nrow = n.sim, ncol = 6)
  mi_est_se = matrix(nrow = n.sim, ncol = 6)
  cc_est_coef = matrix(nrow = n.sim, ncol = 6)
  cc_est_se = matrix(nrow = n.sim, ncol = 6)
```

```r
for (i in 1:n.sim){

  # Simulating the data
  sim1 = sim_data(dataset, p_female = p_female, p0=p0, p1=p1, p2=p2, p3=p3, p4=p4)

  # Calculating true Coef and their se's from the simulated (complete) data

  mixed_model = lmer(FACTF_Total ~ Timepoint + Age + Sex + CaDiagnosis_Generalized + Stage_
  summary(mixed_model)

  Vcov <- vcov(mixed_model, useScale = FALSE)
  betas <- fixef(mixed_model)
  se <- sqrt(diag(Vcov))

  true_coef[i, ] = betas
  true_se[i, ] = se

  # Estimating Coef from the imputed data

  vecdata = mar_final(sim1, p_absent = p_absent, p_missing = p_missing, pattern = pattern)

  imputed_mi = mice(vecdata, m = m) # Imputing the missing data

  mixed_model_imp <- with(imputed_mi,lmer(FACTF_Total ~ Timepoint + Age + Sex +
                      CaDiagnosis_Generalized + Stage_Generalized + (Timepoint|ID)))

  results_mi = summary(pool(mixed_model_imp))

  mi_est_coef[i,] = results_mi[, 1 ]
  mi_est_se[i, ] = results_mi[, 2 ]

  imputed_cc = create_completecasedf(vecdata) # Imputing the missing data

  mixed_model_imp <- lmer(FACTF_Total ~ Timepoint + Age + Sex + CaDiagnosis_Generalized +
                          Stage_Generalized + (Timepoint|ID), data = imputed_cc)

  results_cc = summary(mixed_model_imp)

  cc_est_coef[i,] = coef(results_cc)[,1]
  cc_est_se[i, ] = coef(results_cc)[,2]


  if (i==1){
    name = names(betas)
  }
  print(paste("num of Sims = ", i))
}

qtrue1 = quantile(true_coef[,1], c(0.025, 0.975));
qtrue2 = quantile(true_coef[,2], c(0.025, 0.975))
qtrue3 = quantile(true_coef[,3], c(0.025, 0.975));
qtrue4 = quantile(true_coef[,4], c(0.025, 0.975))
qtrue5 = quantile(true_coef[,5], c(0.025, 0.975));
qtrue6 = quantile(true_coef[,6], c(0.025, 0.975))
```

```
  qtrue_final = cbind(qtrue1, qtrue2, qtrue3, qtrue4, qtrue5, qtrue6)

  qmi_est1 = quantile(mi_est_coef[,1], c(0.025, 0.975));
  qmi_est2 = quantile(mi_est_coef[,2], c(0.025, 0.975))
  qmi_est3 = quantile(mi_est_coef[,3], c(0.025, 0.975));
  qmi_est4 = quantile(mi_est_coef[,4], c(0.025, 0.975))
  qmi_est5 = quantile(mi_est_coef[,5], c(0.025, 0.975));
  qmi_est6 = quantile(mi_est_coef[,6], c(0.025, 0.975))

  qmi_est_final = cbind(qmi_est1, qmi_est2, qmi_est3, qmi_est4, qmi_est5, qmi_est6)

  qcc_est1 = quantile(cc_est_coef[,1], c(0.025, 0.975));
  qcc_est2 = quantile(cc_est_coef[,2], c(0.025, 0.975))
  qcc_est3 = quantile(cc_est_coef[,3], c(0.025, 0.975));
  qcc_est4 = quantile(cc_est_coef[,4], c(0.025, 0.975))
  qcc_est5 = quantile(cc_est_coef[,5], c(0.025, 0.975));
  qcc_est6 = quantile(cc_est_coef[,6], c(0.025, 0.975))

  qcc_est_final = cbind(qcc_est1, qcc_est2, qcc_est3, qcc_est4, qcc_est5, qcc_est6)


  result_matrix = as.data.frame(rbind(round(colMeans(true_coef), digits = 3),
                                       paste("(",round(qtrue_final[1,], digits = 3),",",
                                        round(qtrue_final[2,], digits = 3),")", sep = ""),
                                        round(colMeans(true_se), digits = 3),
                                        round(colMeans(mi_est_coef), digits = 3),
                                       paste("(",round(qmi_est_final[1,], digits = 3),",",
                                        round(qmi_est_final[2,], digits = 3),")", sep = ""),
                                        round(colMeans(mi_est_se), digits = 3),
                                        round(colMeans(cc_est_coef), digits = 3),
                                       paste("(",round(qcc_est_final[1,], digits = 3),",",
                                        round(qcc_est_final[2,], digits = 3),")", sep = ""),

  dimnames(result_matrix) = list(c(paste("Avg of", n.sim, "True Coef"),
                                    "CI for True Coef", paste("Avg of", n.sim, "True SE"),
                                    paste("Avg of", n.sim, "MI Coef"), "CI for MI Coef",
                                    paste("Avg of", n.sim, "MI SE"),
                                    paste("Avg of", n.sim, "CC Coef"), "CI for CC Coef",
                                    paste("Avg of", n.sim, "CC SE")), name)

  print(paste("total num of Sims = ", i))
  kable(result_matrix)
}

#####################################################

# Function That combines everything for
# NMAR

simulation_nmar = function(dataset, p_absent, p_missing, pattern, n.sim = 10, m = 5 ,
                           p, p_female=.8, p0=rep(1,52), p1=rep(1,52), p2=rep(1,52),
                           p3=rep(1,52), p4=rep(1,52)){

  true_coef = matrix(nrow = n.sim, ncol = 6)
```

```r
true_se = matrix(nrow = n.sim, ncol = 6)
mi_est_coef = matrix(nrow = n.sim, ncol = 6)
mi_est_se = matrix(nrow = n.sim, ncol = 6)
cc_est_coef = matrix(nrow = n.sim, ncol = 6)
cc_est_se = matrix(nrow = n.sim, ncol = 6)

for (i in 1:n.sim){

  # Simulating the data
  sim1 = sim_data(dataset, p_female = p_female, p0=p0, p1=p1, p2=p2, p3=p3, p4=p4)

  # Calculating true Coef and their se's from the simulated (complete) data

  mixed_model = lmer(FACTF_Total ~ Timepoint + Age + Sex + CaDiagnosis_Generalized +
                       Stage_Generalized + (Timepoint|ID), data = sim1)

  summary(mixed_model)

  Vcov <- vcov(mixed_model, useScale = FALSE)
  betas <- fixef(mixed_model)
  se <- sqrt(diag(Vcov))

  true_coef[i, ] = betas
  true_se[i, ] = se

  # Estimating Coef from the imputed data

  vecdata = nmar_final(sim1, p_absent = p_absent, p_missing = p_missing,
                       pattern = pattern) # Creating the missing data # Faizan
  #vecdata = nmar_removal(sim1, p_missing = p, pattern = pattern) # Torres

  imputed_mi = mice(vecdata, m = m) # Imputing the missing data

  mixed_model_imp <- with(imputed_mi,lmer(FACTF_Total ~ Timepoint + Age + Sex +
                      CaDiagnosis_Generalized + Stage_Generalized + (Timepoint|ID)))

  results_mi = summary(pool(mixed_model_imp))

  mi_est_coef[i,] = results_mi[, 1 ]
  mi_est_se[i, ] = results_mi[, 2 ]

  imputed_cc = create_completecasedf(vecdata) # Imputing the missing data

  mixed_model_imp <- lmer(FACTF_Total ~ Timepoint + Age + Sex + CaDiagnosis_Generalized
                          + Stage_Generalized + (Timepoint|ID), data = imputed_cc)

  results_cc = summary(mixed_model_imp)

  cc_est_coef[i,] = coef(results_cc)[,1]
  cc_est_se[i, ] = coef(results_cc)[,2]


  if (i==1){
    name = names(betas)
  }
```

```
    print(paste("num of Sims = ", i))
  }

  qtrue1 = quantile(true_coef[,1], c(0.025, 0.975));
  qtrue2 = quantile(true_coef[,2], c(0.025, 0.975))
  qtrue3 = quantile(true_coef[,3], c(0.025, 0.975));
  qtrue4 = quantile(true_coef[,4], c(0.025, 0.975))
  qtrue5 = quantile(true_coef[,5], c(0.025, 0.975));
  qtrue6 = quantile(true_coef[,6], c(0.025, 0.975))

  qtrue_final = cbind(qtrue1, qtrue2, qtrue3, qtrue4, qtrue5, qtrue6)

  qmi_est1 = quantile(mi_est_coef[,1], c(0.025, 0.975));
  qmi_est2 = quantile(mi_est_coef[,2], c(0.025, 0.975))
  qmi_est3 = quantile(mi_est_coef[,3], c(0.025, 0.975));
  qmi_est4 = quantile(mi_est_coef[,4], c(0.025, 0.975))
  qmi_est5 = quantile(mi_est_coef[,5], c(0.025, 0.975));
  qmi_est6 = quantile(mi_est_coef[,6], c(0.025, 0.975))

  qmi_est_final = cbind(qmi_est1, qmi_est2, qmi_est3, qmi_est4, qmi_est5, qmi_est6)

  qcc_est1 = quantile(cc_est_coef[,1], c(0.025, 0.975));
  qcc_est2 = quantile(cc_est_coef[,2], c(0.025, 0.975))
  qcc_est3 = quantile(cc_est_coef[,3], c(0.025, 0.975));
  qcc_est4 = quantile(cc_est_coef[,4], c(0.025, 0.975))
  qcc_est5 = quantile(cc_est_coef[,5], c(0.025, 0.975));
  qcc_est6 = quantile(cc_est_coef[,6], c(0.025, 0.975))

  qcc_est_final = cbind(qcc_est1, qcc_est2, qcc_est3, qcc_est4, qcc_est5, qcc_est6)


  result_matrix = as.data.frame(rbind(round(colMeans(true_coef), digits = 3),
                            paste("(",round(qtrue_final[1,], digits = 3),",",
                                  round(qtrue_final[2,], digits = 3),")", sep = ""),
                                  round(colMeans(true_se), digits = 3),
                                  round(colMeans(mi_est_coef), digits = 3),
                            paste("(",round(qmi_est_final[1,], digits = 3),",",
                                  round(qmi_est_final[2,], digits = 3),")", sep = ""),
                                      round(colMeans(mi_est_se), digits = 3),
                                      round(colMeans(cc_est_coef), digits = 3),
                            paste("(",round(qcc_est_final[1,], digits = 3),",",
                                  round(qcc_est_final[2,], digits = 3),")", sep = ""),
                                  round(colMeans(cc_est_se), digits = 3)))

  dimnames(result_matrix) = list(c(paste("Avg of", n.sim, "True Coef"), "CI for True Coef",
                                   paste("Avg of", n.sim, "True SE"),
                                   paste("Avg of", n.sim, "MI Coef"), "CI for MI Coef",
                                   paste("Avg of", n.sim, "MI SE"),
                                   paste("Avg of", n.sim, "CC Coef"), "CI for CC Coef",
                                   paste("Avg of", n.sim, "CC SE")), name)

  print(paste("total num of Sims = ", i))
  kable(result_matrix)
}
```