SUPPLEMENTARY MATERIAL

Code will be provided on the author's Dataverse.

R-packages for Imputation: 3 R-packages used to impute the missing data: Amelia II, MICE, sbgcop

R-code for simulation in Section 4: R-code to replicate simulation study in section 4.

R-code for Application in Section 5: R-code to replicate application in section 5.

A Missing at Random

We now describe a missing data mechanism that always produces **MAR** data. Our goal is to make the simulations as realistic as possible; therefore some variables will be fully observed, and others will have different amounts of missing values.

1. Given a fully observed data set $\mathbf{X}$ randomly select four variables, one from each of the four classes, that will be fully observed; without loss of generality relabel them X_1, X_{11}, X_{21} and X_{31}.

2. Randomly select four variables from the remaining thirty six, one from each of the four classes, that will have a 5-6% missingness; without loss of generality relabel them X_2, X_{12}, X_{22} and X_{32}. The probability that the i^{th} observation for each variable is missing is based on a logistic regression on the fully observed variables, X_1, X_{11}, X_{21} and X_{31}, adjusted so that the mean number of missing variables is between 5-6%. The missingness indicators are then sampled from independent Bernoulli random variables with the appropriate probabilities. Let $\mathbf{X}^{(1)} = (X_1, X_2, X_{11}, X_{12}, X_{21}, X_{22}, X_{31}, X_{32})$ and $\mathbf{X}^{(1)}_{\text{cc}}$ be the complete cases after removing the any rows that have missing values.

3. The probability of the i^{th} observation missing for the remaining thirty two variables is proportional to a logistic regression on the fully observed $\mathbf{X}^{(1)}_{\text{cc}}$. The probabilities are then adjusted so that the mean number of missing variables is equal to the Missingness Coefficient (MC) (see Table 1 for the range of values that we considered). The missingness indicators are sampled from independent Bernoulli random variables with the appropriate probabilities. If the i^{th} row of $\mathbf{X}^{(1)}$ has been removed in $\mathbf{X}^{(1)}_{cc}$

then that row is always observed for the thirty-two variables.

The proportion of missing values is slightly lower than the MC as four variables are fully observed, and four others only have 5-6% of their values missing.

B Missing not at Random

We now describe a missing data mechanism that produces **MNAR** data with extremely high probability.

1. Given a fully observed data set $\mathbf{X}$ randomly select four variables, one from each of the four classes, that will be fully observed; without loss of generality relabel them X_1, X_{11}, X_{21} and X_{31}.

2. Randomly select four variables from the remaining thirty six, one from each of the four classes, that will have a small amount of missingness; without loss of generality relabel them X_2, X_{12}, X_{22} and X_{32}. The probability that the i^{th} observation is missing is given by,

$$P(R_2 = 1|\mathbf{X}) = 1_{X_2>0}p_{MC},$$

$$P(R_{12} = 1|\mathbf{X}) = 1_{X_{12}=0}p_{MC},$$

$$P(R_{22} = 1|\mathbf{X}) = 1_{X_{22}>3}p_{MC},$$

$$P(R_{32} = 1|\mathbf{X}) = 1_{X_{32}=3}p_{MC},$$

   where the value of p_{MC} is given by the MC in Table 1.

3. For the remaining thirty two variables the probability of the i^{th} observation missing is based on a logistic regression on $\mathbf{X}^{(1)}$ adjusted so that the mean number of missing variables is equal to the MC (see Table 1). The missingness indicators are again sampled from independent Bernoulli random variables with the appropriate probabilities. In contrast to the MAAR mechanism if the i^{th} row of $\mathbf{X}^{(1)}$ has missing values then other variables in that row can still be missing.

C Plots of MNAR Simulation Results

[Figure 5 about here.]

[Figure 6 about here.]

D Number of Simulations for which `Amelia II` crashed

[Table 3 about here.]

E Example `sbgcop` Application

In this section, we discuss how to use the ‘sbgcop‘ package for multiple imputation in the context of conducting inferential analysis on data with missingness. Specifically, we show how to conduct regression analysis in the presence of missing data using an example dataset. First we simulate a dataset in which we introduce missingness.

```r
# simulate data
set.seed(6886)
n <- 100
x1 <- rnorm(n) ; x2 <- rnorm(n) ; x3 <- rnorm(n)
y <- 1 + 2*x1 -1*x2 + 1*x3 + rnorm(n)

## organize into matrix
raw <- cbind(y, x1, x2, x3)

## simulate missingness
naMat <- matrix(rbinom(n*4,1,.7),
    nrow=nrow(raw),ncol=ncol(raw))
naMat[naMat==0] <- NA

## remove observations
data <- raw * naMat

## summarize missingness
missStats <- apply(data, 2, function(x){sum(is.na(x))/nrow(data)})
missStats <- matrix(missStats,
    ncol=1,
    dimnames=list(colnames(data),'Prop. Missing')
    )
```

Using this simulated dataset, our goal is to show how to conduct inference on the effect of x_1, x_2, and x_3 on y after imputing the missing values with the `sbgcop` package in R. `sbgcop` is available on CRAN and can be installed and loaded into your R session just as any other package.

```r
install.packages('sbgcop')
library(sbgcop)
```

The key function in this package is `sbgcop.mcmc` and there are four arguments that should always be set (for a full list of arguments run `?sbgcop.mcmc`):

- Y: a matrix with missing values to be imputed
- `nsamp`: number of iterations of the Markov chain
- `odens`: number of iterations between saved samples
- `seed`: an integer for the random seed

The Y argument specifies the dataset to be imputed. The object passed to the argument must be in **matrix** format. Additionally, users should only include variables that can provide information to the imputation algorithm. For example, this can include lags and leads of a variable in the case of time-series-cross-sectional data. Identification variables, such as actor names, abbreviations, or years, should not be included in the **matrix**.

The imputation procedure in `sbgcop.mcmc` is a Bayesian estimation scheme, so users must pass the number of iterations for which they want the Markov chain to be run to the `nsamp` argument. If `nsamp` is set to 100, then the Markov chain will run for 100 iterations and 100 imputed datasets will be produced. The `odens` argument specifies how often an iteration from the Markov chain should be saved. Thus, if `nsamp` is set to 100 and `odens` is set to 4, 25 imputed datasets will be returned by `sbgcop.mcmc`. Last, since this is a Bayesian model and we will be sampling from distributions to arrive at parameter values, one should always pass an integer to the `seed` argument. This way when users rerun `sbgcop.mcmc` they will arrive at the same results.

To impute missingness in our example dataset, we pass our `data` object to the `sbgcop.mcmc` function. We run the Markov chain for 2000 iterations and save every 10th iteration. We store the output from `sbgcop.mcmc` to `sbgcopOutput`.

```
        sbgcopOutput <- sbgcop.mcmc(Y=data, nsamp=2000, odens=10, seed=6886)
```

This is quite simple to do as the output from `sbgcop.mcmc` is simply a list. The first element in this list is `C.psamp`, which contains posterior samples of the correlation matrix. The `C.psamp` is structured as an array of size p x p x `nsamp/odens`. Where p indicates the number of variables included in the imputation process. In our case, the `data` object includes 4 variables and we ran the Markov chain for 2000 iterations saving every tenth. Thus giving us dimensions of: 4 x 4 x 200.

Each value in this array is providing us with the estimated association between a pair of parameters at every saved iteration of the Markov chain. We show an example below using the 100th and 200th saved iterations.

```
sbgcopOutput$C.psamp[,,c(100,200)]

## , , 100
##
##               y          x1          x2          x3
## y   1.0000000  0.78961179 -0.43494151  0.36593885
## x1  0.7896118  1.00000000 -0.08686933  0.05172101
## x2 -0.4349415 -0.08686933  1.00000000 -0.14619182
## x3  0.3659389  0.05172101 -0.14619182  1.00000000
##
## , , 200
##
##               y         x1          x2         x3
## y   1.0000000 0.68269537 -0.46139236  0.4138161
## x1  0.6826954 1.00000000  0.08754115  0.1495993
## x2 -0.4613924 0.08754115  1.00000000 -0.1278238
## x3  0.4138161 0.14959933 -0.12782384  1.0000000
```

To generate a trace plot of this data we need to restructure our dataframe into a long format. We can do so using the `reshape2` package:

```
 library(reshape2)
 sbgcopCorr = reshape2::melt(sbgcopOutput$'C.psamp')

 # remove cases where variable is the same in both columns
 sbgcopCorr = sbgcopCorr[sbgcopCorr$Var1 != sbgcopCorr$Var2,]

 # construct an indicator for pairs of variables
 sbgcopCorr$v12 = paste(sbgcopCorr$Var1, sbgcopCorr$Var2, sep='-')

 #
 print(head(sbgcopCorr))

##   Var1 Var2 Var3       value   v12
## 2   x1    y    1  0.62439270  x1-y
## 3   x2    y    1 -0.43347850  x2-y
## 4   x3    y    1  0.28013565  x3-y
## 5    y   x1    1  0.62439270  y-x1
## 7   x2   x1    1  0.03581958 x2-x1
## 8   x3   x1    1  0.15626246 x3-x1
```

Using the `reshape2` package we have reformatted the array into a dataframe, in which the first two columns designate the variables for which a correlation is being estimated, the third an indicator of the saved iteration, the fourth the correlation, and the fifth an indicator designating the variables being compared.

Next, we use `ggplot2` to construct a simple trace plot shown in Figure E.7.

```r
library(ggplot2)

ggplot(sbgcopCorr, aes(x=Var3, y=value, color=v12)) +
    geom_line() +
    ylab('Correlation') + xlab('Iteration') +
    facet_wrap(~v12) +
    theme(legend.position='none')
```

[Figure 7 about here.]

Based on these trace plots we can see that the Markov chain tends to converge rather quickly in this example. The `coda` package provides an excellent set of diagnostics to test convergence in more depth.

After conducting the imputation and evaluating convergence, our goal is now to use the imputed datasets to conduct inferential analysis. For the purpose of this example, we estimate the effect of x_1, x_2, and x_3 on y. By using `sbgcop` as above we have generated 200 copies of our original dataset in which posterior samples of the original missing values have been included. Each of these copies are saved in the output from `sbgcop.mcmc`, which has dimensions of 100 x 4 x 200.

The first two dimensions of this object correspond to the original dimensions of our `data` object, and the third corresponds to the number of saved iterations from the Markov chain.

Having generated a set of imputed datasets, our next step is to use a regression model to estimate the effect of our independent variables on y. We cannot just use one of the imputed datasets – as this would not take into account the uncertainty in our imputations. Instead we run several regression on as many of the imputed datasets generated by `sbgcop.mcmc` that we think are appropriate. For the sake of this example, we utilize all 200 imputed datasets, but typically randomly sampling around 20 imputed datasets should be be sufficient.

Each time we run the regression model, we will save the coefficient and standard errors for the independent variables and organize the results into a matrix as shown below.

```
coefEstimates <- NULL
serrorEstimates <- NULL
for( copy in 1:dim(sbgcopOutput$'Y.impute')[3]){
        # extract copy from sbgcopOutput
        copyDf <- data.frame(sbgcopOutput$'Y.impute'[,,copy])
        names(copyDf) <- colnames(sbgcopOutput$Y.pmean)
        # run model
        model <- lm(y~x1+x2+x3,data=copyDf)
        # extract coefficients
        beta <- coef(model)
        coefEstimates <- rbind(coefEstimates, beta)
        # extract standard errors
        serror <- sqrt(diag(vcov(model)))
        serrorEstimates <- rbind(serrorEstimates, serror)
}

print(head(coefEstimates))

##      (Intercept)       x1         x2        x3
## beta   0.6576411 1.449662 -1.1290934 0.4569379
## beta   0.7436243 1.661250 -1.0542155 0.6866980
## beta   0.8299671 1.613892 -1.1363969 0.7454211
## beta   0.8073597 1.513452 -0.7512275 0.6331863
## beta   0.8112010 1.583065 -0.9608251 0.6529509
## beta   0.7882072 1.509635 -0.5152139 0.8897130
```

The last step is to combine each of the estimates using using Rubin's rule. Many existing packages have implemented functions to aid in this last step, one could use the `pool` function from `mice` or the `mi.meld` function from `Amelia II` as below.

```
  paramEstimates <- Amelia::mi.meld(q=coefEstimates, se=serrorEstimates)
  print(paramEstimates)

## £q.mi
##       (Intercept)      x1          x2         x3
## [1,]     0.892732 1.70032 -0.9023761 0.7235922

## £se.mi
##       (Intercept)         x1         x2         x3
## [1,]   0.1680402 0.1965969 0.2213771 0.1588638
```

The resulting parameter estimates take into account the uncertainty introduced through the imputation process, and we can interpret them just as we would interpret the results from a typical regression.

Below we show the full set of steps required to conduct a regression analysis in the context of missing data using `sbgcop`.

```
library(sbgcop)
sbgcopOutput <- sbgcop.mcmc(Y=data, nsamp=2000, odens=10, seed=6886)

## restructure posterior samples of correlation matrix
library(reshape2)
sbgcopCorr = reshape2::melt(sbgcopOutput$'C.psamp')
sbgcopCorr = sbgcopCorr[sbgcopCorr$Var1 != sbgcopCorr$Var2,]
sbgcopCorr$v12 = paste(sbgcopCorr$Var1, sbgcopCorr$Var2, sep='-')

## trace plot of C.psamp
library(ggplot2)
```

```
ggplot(sbgcopCorr, aes(x=Var3, y=value, color=v12)) +
        geom_line() +
        ylab('Correlation') + xlab('Iteration') +
        facet_wrap(~v12) +
        theme(legend.position='none')

## conduct regression analysis
coefEstimates <- NULL
serrorEstimates <- NULL
for( copy in 1:dim(sbgcopOutput$'Y.impute')[3]){
                copyDf <- data.frame(sbgcopOutput$'Y.impute'[,,copy])
        names(copyDf) <- colnames(sbgcopOutput$Y.pmean)
                model <- lm(y~x1+x2+x3,data=copyDf)
                beta <- coef(model)
                coefEstimates <- rbind(coefEstimates, beta)
                serror <- sqrt(diag(vcov(model)))
                serrorEstimates <- rbind(serrorEstimates, serror) }

## combine estimates using Rubin's rules
paramEstimates <- Amelia::mi.meld(q=coefEstimates, se=serrorEstimates)
```

References

Bojinov, I., N. Pillai, and D. Rubin (2017). Diagnosing missing always at random in multivariate data. *arXiv preprint arXiv:1710.06891*.

Casella, G. and E. I. George (1992). Explaining the Gibbs Sampler. *The American Statistician 46*(3), 167–174.

Chen, S.-H. and E. H. Ip (2015). Behaviour of the Gibbs sampler when conditional distributions are potentially incompatible. *Journal of statistical computation and simulation 85*(16), 3266–3275.

Chen, X., Y. Fan, and V. Tsyrennikov (2006). Efficient Estimation of Semiparametric Multivariate Copula Models. *Journal of the American Statistical Association 101*(475), 1228–1240.

Di Lascio, F., S. Giannerini, and A. Reale (2015). Exploring copulas for the imputation of complex dependent data. *Statistical Methods & Application 24*(1), 159–174.

Erler, N. S., D. Rizopoulos, J. v. Rosmalen, V. W. Jaddoe, O. H. Franco, and E. M. Lesaffre (2016). Dealing with Missing Covariates in Epidemiologic Studies: A Comparison Between Multiple Imputation and a Full Bayesian Approach. *Statistics in Medicine*.

Goodrich, B., J. Kropko, A. Gelman, and J. Hill (2012). mi: Iterative Multiple Imputation from Conditional Distributions. R package.

Graham, J. W. (2009). Missing Data Analysis: Making it Work in the Real World. *Annual Review of Psychology 60*(1), 549–576.

Hoff, P. (2010). *sbgcop: Semiparametric Bayesian Gaussian Copula Estimation and Imputation.* R package version 0.975. `https://CRAN.R-project.org/package=sbgcop`.

Hoff, P. D. (2007). Extending the Rank Likelihood for Semiparametric Copula Estimation. *Annals of Applied Statistics 1*(1), 265–283.

Hoff, P. D., X. Niu, and J. A. Wellner (2014). Information Bounds for Gaussian Copulas. *Bernoulli: official journal of the Bernoulli Society for Mathematical Statistics and Probability 20*(2), 604.

Honaker, J. and G. King (2010, April). What to do About Missing Values in Time-Series Cross-Section Data. *American Journal of Political Science 54*(2), 561–581.

Honaker, J., G. King, and M. Blackwell (2012). AMELIA II: A Program for Missing Data – Documentation.

Imai, K., G. King, and O. Lau (2008). Toward A Common Framework for Statistical Analysis and Development. *Journal of Computational and Graphical Statistics 17*(4), 892–913.

Käärik, E. (2006). Imputation algorithm using copulas. *Metodoloski zvezki 3*(1), 109.

Käärik, E. and M. Käärik (2009). Modeling dropouts by conditional distribution, a copula-based approach. *Journal of Statistical Planning and Inference 139*, 3830–3835.

King, G., J. Honaker, A. Joseph, and K. Scheve (2001, March). Analyzing Incomplete Political Science Data: An Alternative Algorithm for Multiple Imputation. *American Politial Science Review 95*(1), 49–69.

Klaassen, C. A., J. A. Wellner, et al. (1997). Efficient Estimation in the Bivariate Normal Copula Model: Normal Margins are Least Favourable. *Bernoulli 3*(1), 55–77.

Krieckhaus, J., B. Son, N. Bellinger, and J. Wells (2014). Economic Inequality and Democratic Support. *The Journal of Politics 76*(1), 139–151.

Kropko, J., B. Goodrich, A. Gelman, and J. Hill (2014). Multiple Imputation for Continuous and Categorical Data: Comparing Joint Multivariate Normal and Conditional Approaches. *Political Analysis 22*(4), 497–519.

Li, F., Y. Yu, and D. B. Rubin (2012). Imputing Missing Data by Fully Conditional Models: Some Cautionary Examples and Guidelines. *Duke University Department of Statistical Science Discussion Paper 1124*.

Little, R. J. and D. B. Rubin (2002). *Statistical Analysis with Missing Data* (second ed.). New York: Wiley.

Liu, J., A. Gelman, J. Hill, Y.-S. Su, and J. Kropko (2013). On the Stationary Distribution of Iterative Imputations. *Biometrika 101*(1), 155–173.

Marini, M. M., A. R. Olsen, and D. B. Rubin (1980). Maximum-likelihood estimation in panel studies with missing data. *Sociological methodology 11*, 314–357.

Mealli, F. and D. B. Rubin (2015). Clarifying Missing at Random and Related Definitions, and Implications when Coupled with Exchangeability. *Biometrika 102*(4), 995–1000.

Molenberghs, G., G. Fitzmaurice, M. G. Kenward, A. Tsiatis, and G. Verbeke (2014). *Handbook of Missing Data Methodology.* Boca Raton, FL: Chapman and Hall/CRC.

Murray, J. S. (2013). Some Recent Advances in Non- and Semiparametric Bayesian Modeling with Copulas, Mixtures, and Latent Variables. Dissertation. Department of Statistical Science Duke University. `http://dukespace.lib.duke.edu/dspace/handle/10161/8253`.

Murray, J. S., D. B. Dunson, L. Carin, and J. E. Lucas (2013). Bayesian Gaussian Copula Factor Models for Mixed Data. *Journal of the American Statistical Association 108*(502), 656–665.

Pettitt, A. (1982). Inference for the Linear Model using a Likelihood Based on Ranks. *Journal of the Royal Statistical Society. Series B (Methodological) 44*(2), 234–243.

Pitt, M., D. Chan, and R. Kohn (2006). Efficient Bayesian Inference for Gaussian Copula Regression Models. *Biometrika 93*(3), 537–554.

R Development Core Team (2004). *R: A language and environment for statistical computing.* Vienna, Austria.

Robbins, M. W., S. K. Ghosh, and J. D. Habiger (2013). Imputation in High-Dimensional Economic Data as Applied to the Agricultural Resource Management Survey. *Journal of the American Statistical Association 108*(501), 81–95.

Rubin, D. B. (1976). Inference and Missing Data. *Biometrika 63*(3), 581–592.

Rubin, D. B. (1987). *Multiple Imputation for Nonresponse in Surveys.* John Wiley & Sons.

Rubin, D. B. (1996). Multiple Imputation After 18+ Years. *Journal of the American statistical Association 91*(434), 473–489.

Rubin, D. B. (2004). *Multiple imputation for Nonresponse in Surveys*, Volume 81. John Wiley & Sons.

Sklar, A. (1959). Fonctions de Répartition à N Dimensions et Leur Marges. *Publications de l'Institut Statistique de l'Université Paris 8*, 229–231.

Van Buuren, S. (2012). *Flexible Imputation of Missing Data.* Boca Raton, FL: Chapman & Hall/CRC Press.

Van Buuren, S. and K. Groothuis-Oudshoorn (2011). MICE: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software 45*(3), 1–67.

Van Buuren, S. and K. Groothuis-Oudshoorn (2011). MICE: Multivariate imputation by chained equations in R. *Journal of statistical software 45*(3), 1–67.

White, I. R. and J. B. Carlin (2010). Bias and Efficiency of Multiple Imputation Compared with Complete-Case Analysis for Missing Covariate Values. *Statistics in Medicine 29*(28), 2920–2931.

World Values Survey (2012). 1981-2008 Integrated Questionnaire.

Yucel, R. M. (2011). State of the Multiple Imputation Software. *Journal of Statistical Software 45*(1), 1 – 7.

Zhu, J. and T. E. Raghunathan (2015). Convergence Properties of a Sequential Regression Multiple Imputation Algorithm. *Journal of the American Statistical Association 110*(511), 1112–1124.

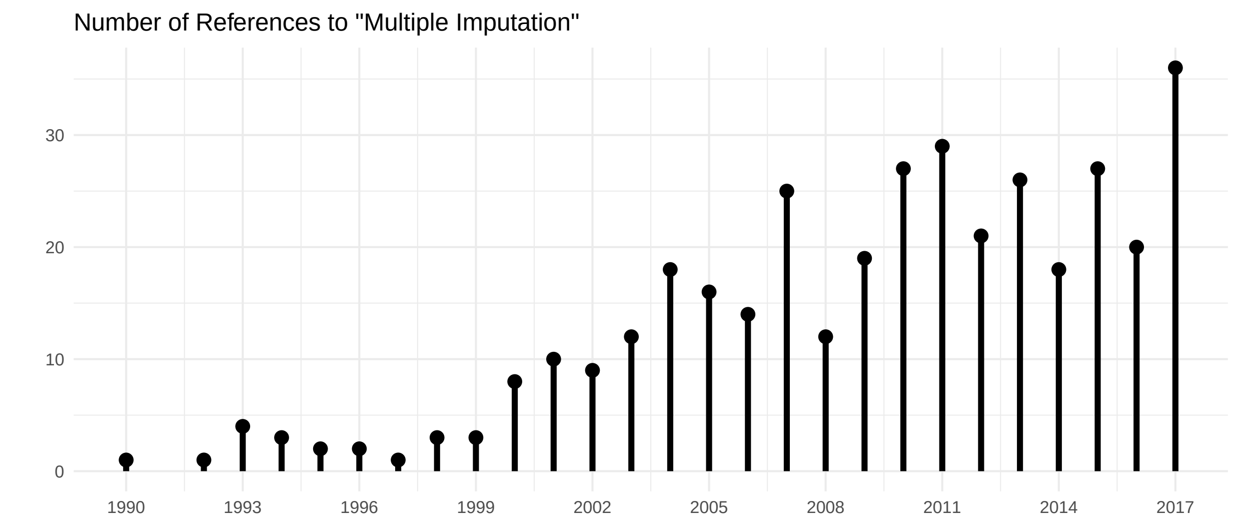

Figure E.1: Number of references to “multiple imputation” in articles from five top sociology and political science journals since 1990.

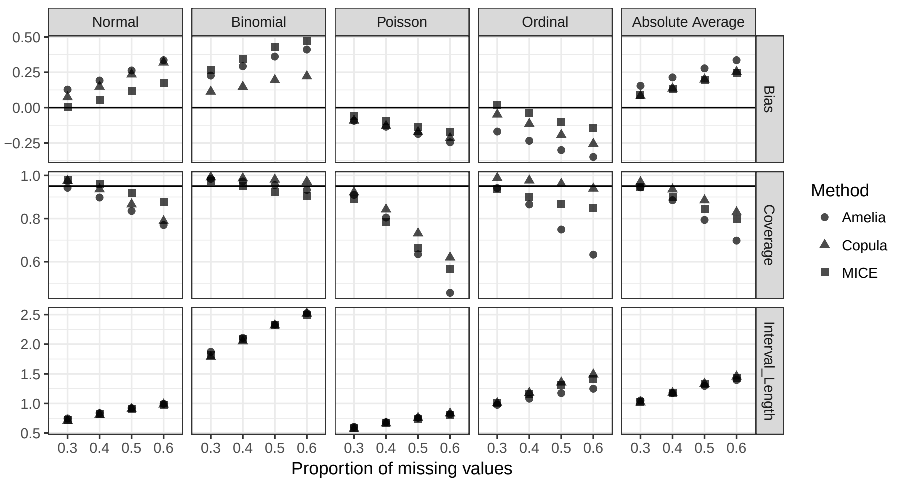

Figure E.2: Simulation study results for the **MAR** data as a function of the missingness coefficient, averaging over the correlation. The plot is split by the different variable types (normal, binomial, Poisson and ordinal) and the three outcomes of interested (the bias, coverage and interval length). The rightmost panel shows the result averaging over the different variable types.

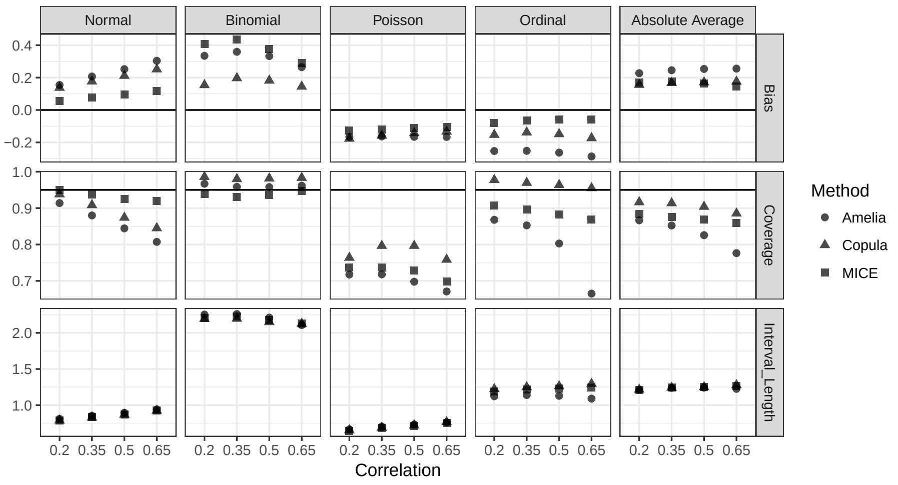

Figure E.3: Simulation study results for the **MAR** data as a function of the correlation, averaging over the missingness coefficient. The plot is split by the different variable types (normal, binomial, Poisson and ordinal) and the three outcomes of interested (the bias, coverage and interval length). The rightmost panel shows the result averaging over the different variable types.

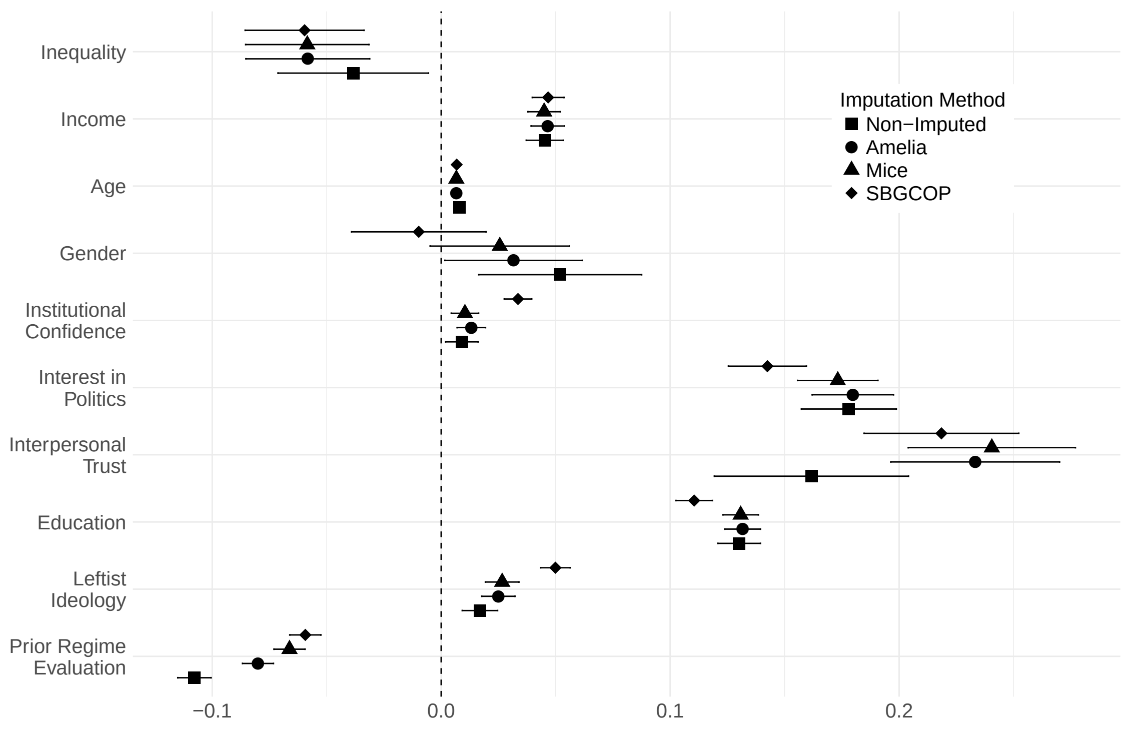

Figure E.4: Coefficient estimates and confidence intervals for *Model 1* in *Table 1* in Krieckhaus et al. (2014) based on three imputation techniques and list-wise deletion

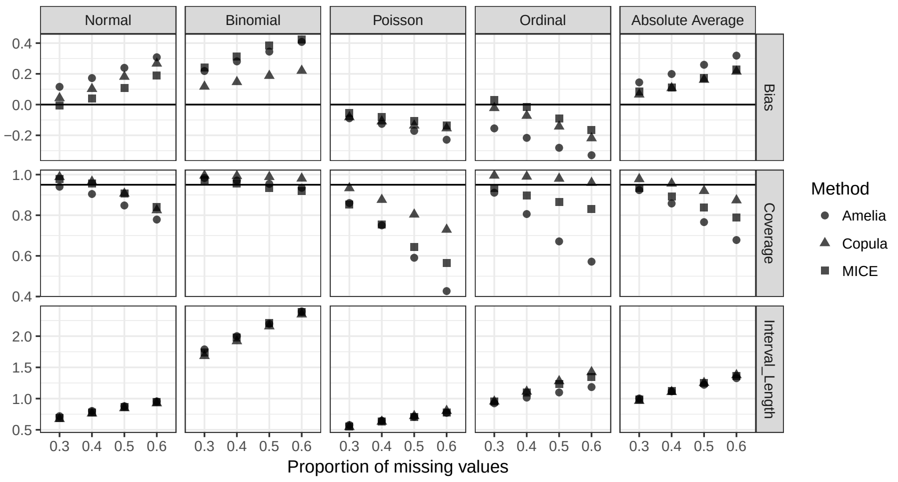

Figure E.5: Simulation study results for the **MNAR** data as a function of the missingness coefficient, averaging over the correlation. The plot is split by the different variable types (normal, binomial, Poisson and ordinal) and the three outcomes of interested (the bias, coverage and interval length). The rightmost panel shows the result averaging over the different variable types.

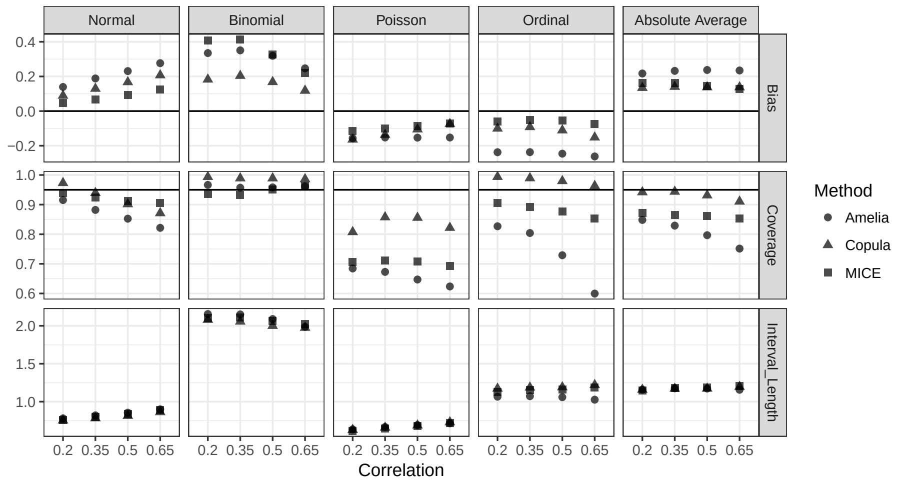

Figure E.6: Simulation study results for the **MNAR** data as a function of the correlation, averaging over the missingness coefficient. The plot is split by the different variable types (normal, binomial, Poisson and ordinal) and the three outcomes of interested (the bias, coverage and interval length). The rightmost panel shows the result averaging over the different variable types.

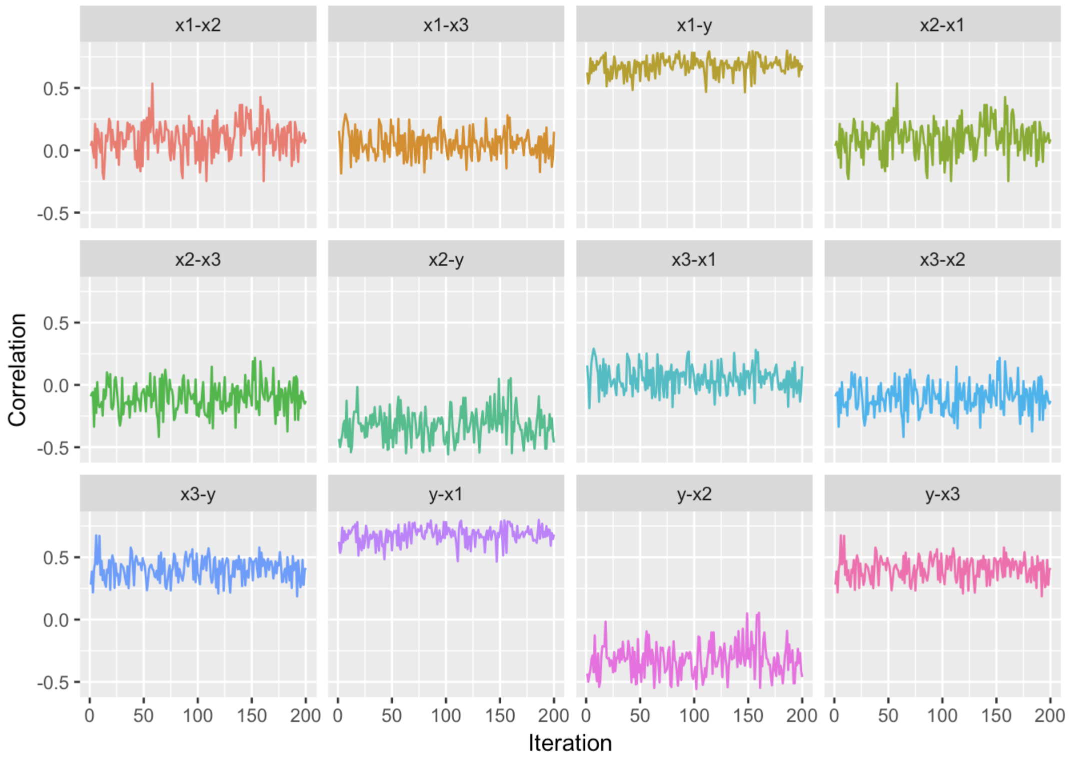

Figure E.7: Trace plot of correlation between variables.

	Missingness
Correlation (ρ)	Coefficient (MC)
0.2	0.3
0.35	0.4
0.5	0.5
0.65	0.6

Table E.1: Simulation Study configurations.

Table E.2: Share of Missingness in Variables of Interest

Democracy Support	Inequality	Income	Age
19.9	1.8	12.9	0.2
Gender	Institutional Confidence	Interest in Politics	Interpersonal Trust
0.1	11.7	2.5	3.7
Education	Leftist Ideology	Prior Regime Evaluation	
3.9	18.5	21.3	

		Correlation			
		0.2	0.35	0.5	0.65
	0.3	2	0	0	7
Share of	0.4	93	16	8	0
Missingness	0.5	285	138	37	13
	0.6	485	305	159	72

Table E.3: The number of `Amelia II` crashes out of the 1000 simulations under each of the possible scenarios.