

Comparison of parametric and Random Forest MICE in imputation of missing data in survival analysis

Anoop D. Shah, Jonathan W. Bartlett, James Carpenter,
Owen Nicholas and Harry Hemingway

November 25, 2021

Contents

1	Introduction	1
2	Methods	2
2.1	Missingness mechanism	5
3	Results	6
3.1	Fully observed variables	6
3.2	Partially observed variable	8
3.3	Pairwise comparisons between methods	8
3.3.1	Comparison of bias	8
3.3.2	Comparison of precision	9
3.3.3	Comparison of confidence interval length	9
3.3.4	Comparison of confidence interval coverage	9
4	Discussion	10
4.1	CART versus Random Forest MICE	10
4.2	Comparison of Random Forest MICE methods	10
4.3	missForest	10
4.4	Implications for further research	10
5	Appendix: R code	10
5.1	R functions	10
5.1.1	Data generating functions	11
5.1.2	Functions to analyse data	12
5.1.3	Functions to compare methods	15
5.1.4	Functions to compile and display results	17
5.2	R script	19

1 Introduction

This is a simulation study comparing various methods for imputation of missing covariate data in a survival analysis in which there are interactions between the predictor variables. We compare our new Random Forest method for MICE (Multivariate Imputation by Chained Equations) with other imputation methods and full data analysis. In our Random Forest method (RFcont), the conditional mean missing values are predicted using Random Forest and imputed values are drawn from Normal distributions centred on the predicted means [1].

We also perform a comparison with the methods recently published by Doove et al. [2]: `mice.impute.cart` (classification and regression trees in MICE) and `mice.impute.rf` (MICE using Random Forests).

2 Methods

We used the R packages **CALIBERr**, **fimpute**, **survival**, **xtable**, **missForest** and **randomForest**. We created simulated survival datasets with two fully observed predictor variables (x_1 , x_2) and a partially observed predictor (x_3), which depends on x_1 , x_2 and their interaction. They were generated as follows:

x_1 Standard normal distribution

x_2 Standard normal distribution, independent of x_1

x_3 Derived from x_1 and x_2 : $x_3 = 0.5(x_1 + x_2 - x_1.x_2) + e$ where e is normally distributed with mean 0 and variance 1.

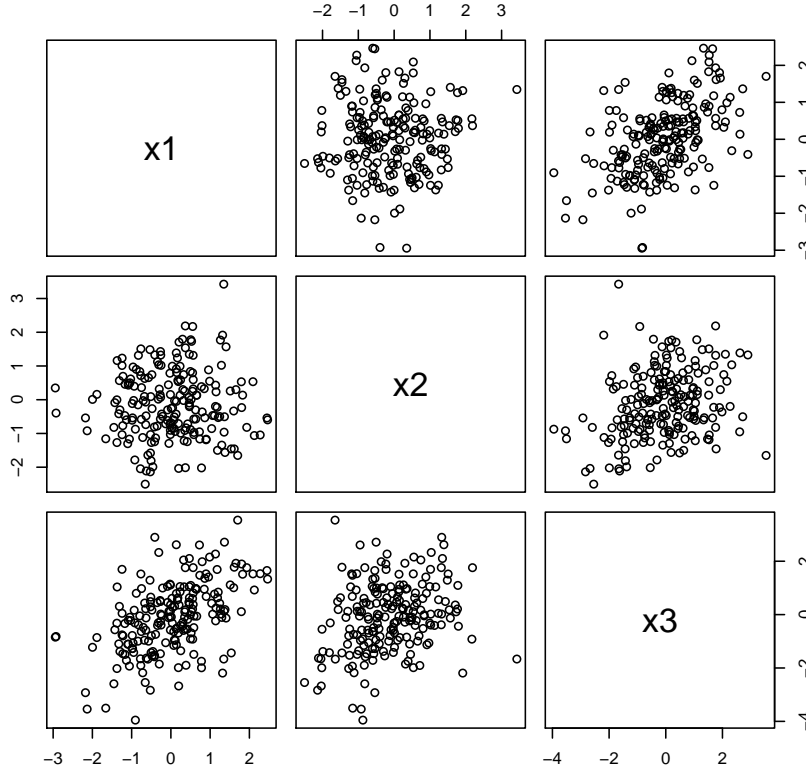
The equation for the log hazard of patient i was given by:

$$h_i = \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 x_{3i} \quad (1)$$

where all the β coefficients were set to 0.5.

We used an exponential distribution to generate a survival time for each patient. We also generated an observation time for each patient, as a random draw from a uniform distribution bounded by zero and the 50th percentile of survival time. If the observation time was less than the survival time, the patient was considered as censored (event indicator 0, and the patient's follow-up ends on their censoring date), otherwise the event indicator was 1, with follow-up ending on the date of event.

Associations between predictor variables in a sample dataset



Linear regression model relating x_3 to x_1 and x_2 :

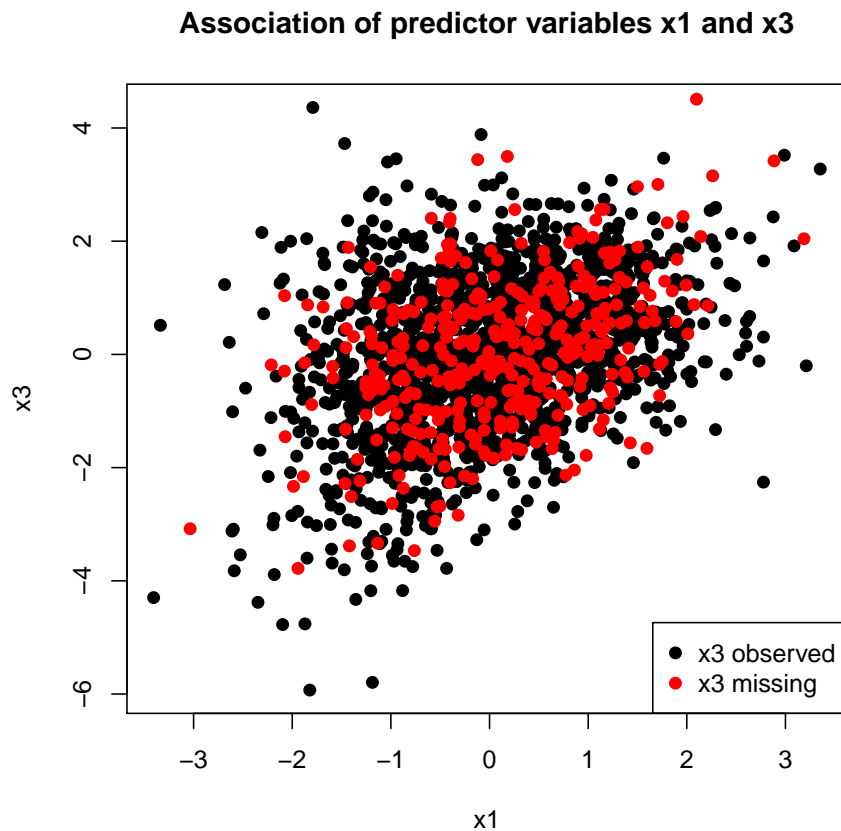
```
> # Chunk 4
>
> summary(lm(x3 ~ x1*x2, data = mydata))

Call:
lm(formula = x3 ~ x1 * x2, data = mydata)

Residuals:
    Min       1Q   Median       3Q      Max
-4.2640 -0.6702 -0.0022  0.6705  4.7581

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.006736   0.007044  -0.956   0.339
x1           0.492275   0.006997  70.359 <2e-16 ***
x2           0.499378   0.007096  70.372 <2e-16 ***
x1:x2        -0.498340   0.006959 -71.608 <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9962 on 19996 degrees of freedom
Multiple R-squared:  0.4311,    Adjusted R-squared:  0.431
F-statistic: 5050 on 3 and 19996 DF,  p-value: < 2.2e-16
```



All true log hazard ratios were assumed to be 0.5, with hazard ratios = 1.65. We checked that the hazard ratios in the simulated data were as expected for a large sample:

```

> # Chunk 6
>
> # Cox proportional hazards analysis
> myformula <- as.formula(Surv(time, event) ~ x1 + x2 + x3)
> # Analysis with 10,000 simulated patients (or more
> # if the variable REFERENCE_SAMPLESIZE exists)
> if (!exists('REFERENCE_SAMPLESIZE')){
+   REFERENCE_SAMPLESIZE <- 10000
+ }
> # Use parallel processing, if available, to create
> # datasets more quickly.
> if ('parallel' %in% loadedNamespaces() &&
+   !is.null(getOption('mc.cores')) &&
+   .Platform$OS.type == 'unix'){
+   REFERENCE_SAMPLESIZE <- REFERENCE_SAMPLESIZE %/%
+     getOption('mc.cores')
+   simdata <- parallel::mclapply(1:getOption('mc.cores'),
+     function(x) makeSurv(REFERENCE_SAMPLESIZE))
+   simdata <- do.call('rbind', simdata)
+ } else {
+   simdata <- makeSurv(REFERENCE_SAMPLESIZE)
+ }
> summary(coxph(myformula, data = simdata))

```

Call:

```
coxph(formula = myformula, data = simdata)
```

n= 10000, number of events= 3157

	coef	exp(coef)	se(coef)	z	Pr(> z)
x1	0.49611	1.64232	0.01894	26.19	<2e-16 ***
x2	0.51660	1.67632	0.01926	26.82	<2e-16 ***
x3	0.51870	1.67985	0.01701	30.49	<2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

	exp(coef)	exp(-coef)	lower .95	upper .95
x1	1.642	0.6089	1.582	1.704
x2	1.676	0.5965	1.614	1.741
x3	1.680	0.5953	1.625	1.737

Concordance= 0.766 (se = 0.004)

Likelihood ratio test= 3041 on 3 df, p=<2e-16

Wald test = 2562 on 3 df, p=<2e-16

Score (logrank) test = 2533 on 3 df, p=<2e-16

We created datasets containing 200 simulated patients. For each dataset, we first analysed the complete dataset with no values missing, then artificially created missingness in variable x_3 , imputed the missing values using various methods, and analysed the imputed datasets. We combined parameter estimates from multiply imputed datasets using Rubin's rules.

2.1 Missingness mechanism

Missingness was imposed in x_3 dependent on x_1 , x_2 , the event indicator and the marginal Nelson-Aalen cumulative hazard, using a logistic regression model. The linear predictors were offset by an amount chosen to make the overall proportion of each variable missing approximately 0.2, i.e.:

$$P(\text{miss})_i = \frac{\exp(lp_i + \text{offset})}{1 + \exp(lp_i + \text{offset})} \quad (2)$$

$$lp_i = 0.1x_{1i} + 0.1x_{2i} + 0.1 \times \text{cumhaz}_i + 0.1 \times \text{event}_i \quad (3)$$

where ‘event’ is the event indicator and ‘cumhaz’ is the marginal Nelson-Aalen cumulative hazard.

We analysed the datasets with missing data using different methods of multiple imputation. We calculated the marginal Nelson-Aalen cumulative hazard and included it in all imputation models, along with the event indicator and follow-up time.

```
> # Chunk 7
>
> # Setting analysis parameters: To analyse more than 3 samples,
> # set N to the desired number before running this program
> if (!exists('N')){
+   N <- 3
+ }
> # Number of imputations (set to at least 10 when
> # running an actual simulation)
> if (!exists('NIMPS')){
+   NIMPS <- 4
+ }
> # Use parallel processing if the 'parallel' package is loaded
> if ('parallel' %in% loadedNamespaces() &&
+     .Platform$OS.type == 'unix'){
+   cat('Using parallel processing\n')
+   results <- parallel::mclapply(1:N, doanalysis)
+ } else {
+   results <- lapply(1:N, doanalysis)
+ }
```

Using parallel processing

We used the following methods of multiple imputation. The number of imputations was 10. In each case, the imputation model for x_3 contained x_1 , x_2 , the event indicator and the marginal Nelson-Aalen cumulative hazard:

missForest – from the missForest package, which completes a dataset in an iterative way using Random Forest prediction. It was run with maximum 10 iterations (default) and 100 trees per forest (default).

CART MICE – Classification and regression tree MICE method from the mice package (mice.impute.cart).

RF MICE (Doove) – Random Forest MICE method from Doove et al. [2], which is available as function mice.impute.rf in the mice package, with 10 or 100 trees.

RFcont MICE – Random Forest MICE method from the CALIBERrfimpute package with 5, 10, 20 or 100 trees.

Parametric MICE – normal-based linear regression with default settings, in which the imputation model for x_3 is of the form:

$$x_3 = \beta_0 + \beta_1.x_1 + \beta_2.x_2 + \beta_3.event + \beta_4.cumhaz + e$$

where e is the residual variance.

We analysed 1000 samples. We calculated the following for each method and each parameter:

- Bias of log hazard ratio
- Standard error of bias (Monte Carlo error)
- Mean square error
- Standard deviation of estimated log hazard ratio
- Mean length of 95% confidence intervals
- Coverage of 95% confidence intervals (proportion containing the true log hazard ratio)

3 Results

All the true log hazard ratios were set at 0.5.

3.1 Fully observed variables

Log hazard ratio for the continuous fully observed variable x_1 :

	Bias	Standard error of bias	Mean square error	SD of estimate	Mean 95% CI length	95% CI coverage
Full data	0.0167	0.00467	0.0221	0.148	0.558	0.945
missForest	-0.00689	0.00481	0.0232	0.152	0.561	0.941
CART MICE	0.00922	0.00475	0.0226	0.15	0.57	0.942
RF Doove MICE 10	0.0196	0.0046	0.0215	0.145	0.571	0.956
RF Doove MICE 100	0.0202	0.00459	0.0215	0.145	0.571	0.947
RFcont MICE 5	0.0178	0.00465	0.0219	0.147	0.576	0.949
RFcont MICE 10	0.0141	0.00464	0.0217	0.147	0.573	0.951
RFcont MICE 20	0.0111	0.00466	0.0218	0.147	0.572	0.957
RFcont MICE 100	0.00961	0.00465	0.0217	0.147	0.571	0.956
Parametric MICE	-0.0129	0.00478	0.023	0.151	0.583	0.951

Log hazard ratio for the continuous fully observed variable x_2 :

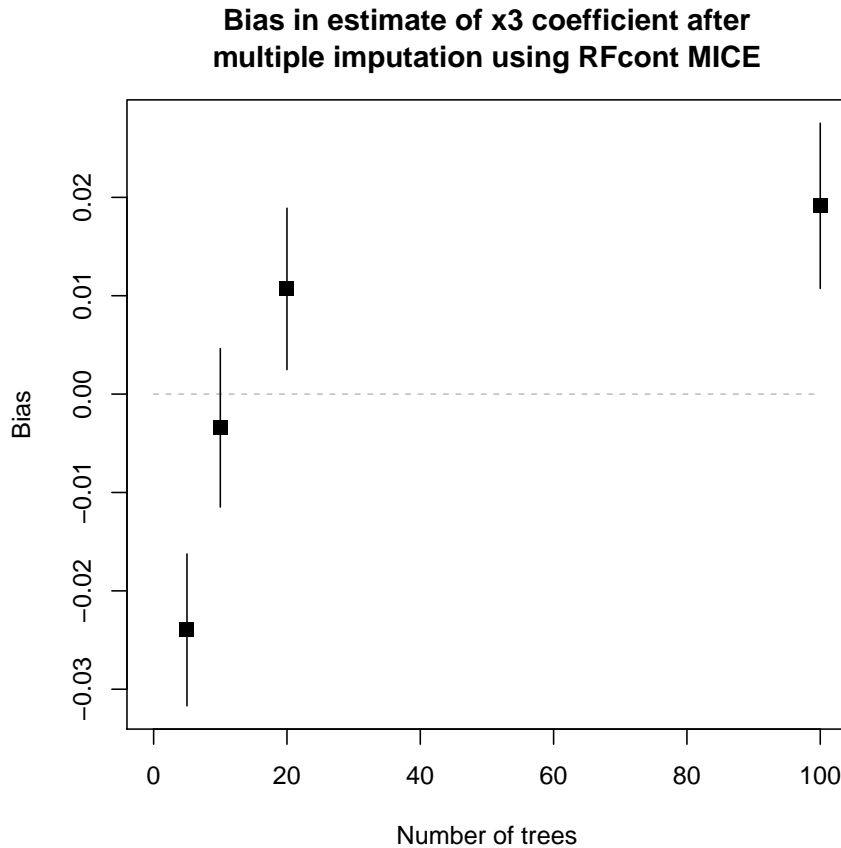
	Bias	Standard error of bias	Mean square error	SD of estimate	Mean 95% CI length	95% CI coverage
Full data	0.00996	0.00449	0.0202	0.142	0.56	0.957
missForest	-0.0122	0.0047	0.0222	0.148	0.563	0.952
CART MICE	0.00363	0.00464	0.0215	0.147	0.571	0.961
RF Doove MICE 10	0.0147	0.00446	0.0201	0.141	0.573	0.969
RF Doove MICE 100	0.0151	0.00446	0.0201	0.141	0.573	0.97
RFcont MICE 5	0.0124	0.00451	0.0205	0.143	0.577	0.97
RFcont MICE 10	0.00893	0.00449	0.0202	0.142	0.574	0.971
RFcont MICE 20	0.00616	0.00451	0.0204	0.143	0.574	0.967
RFcont MICE 100	0.00455	0.00454	0.0206	0.144	0.573	0.966
Parametric MICE	-0.0182	0.00465	0.0219	0.147	0.584	0.959

3.2 Partially observed variable

Log hazard ratio for the continuous partially observed variable x_3 :

	Bias	Standard error of bias	Mean square error	SD of estimate	Mean 95% CI length	95% CI coverage
Full data	0.0183	0.00403	0.0166	0.128	0.491	0.951
missForest	0.0927	0.00503	0.0338	0.159	0.532	0.868
CART MICE	-0.00586	0.00471	0.0222	0.149	0.54	0.932
RF Doove MICE 10	0.00959	0.00413	0.0171	0.131	0.549	0.969
RF Doove MICE 100	0.0104	0.00413	0.0172	0.131	0.549	0.969
RFcont MICE 5	-0.024	0.00394	0.0161	0.125	0.564	0.972
RFcont MICE 10	-0.00343	0.00411	0.0169	0.13	0.556	0.968
RFcont MICE 20	0.0107	0.00419	0.0176	0.132	0.556	0.969
RFcont MICE 100	0.0191	0.00429	0.0187	0.136	0.553	0.962
Parametric MICE	-0.037	0.00408	0.018	0.129	0.573	0.965

The following graph shows the bias for RFcont MICE methods by number of trees (bias estimated from 1000 simulations; the lines denote 95% confidence intervals):



3.3 Pairwise comparisons between methods

3.3.1 Comparison of bias

Difference between absolute bias (negative means that the first method is less biased). P values from paired sample t tests. Significance level: * $P < 0.05$, ** $P < 0.01$, *** $P < 0.001$.

Coefficient	RFcont MICE 10 vs parametric MICE	RFcont MICE 100 vs parametric MICE	RFcont MICE 10 vs RFcont MICE 100
x1	0.00114	-0.00333	0.00447 ***
x2	-0.00925	-0.0136	0.00438 ***
x3	-0.0336 ***	-0.0178 *	-0.0157

Coefficient	RF Doove MICE 10 vs RFcont MICE 10	RF Doove MICE 10 vs CART MICE	RF Doove MICE 10 vs RF Doove MICE 100
x1	0.00552 ***	0.0104 ***	-0.000557
x2	0.00577 ***	0.0111 ***	-0.000355
x3	0.00616	0.00373	-0.00085

3.3.2 Comparison of precision

Ratio of variance of estimates (less than 1 means that the first method is more precise). P values from F test. Significance level: * P <0.05, ** P <0.01, *** P <0.001.

Coefficient	RFcont MICE 10 vs parametric MICE	RFcont MICE 100 vs parametric MICE	RFcont MICE 10 vs RFcont MICE 100
x1	0.943	0.946	0.996
x2	0.932	0.954	0.976
x3	1.02	1.1	0.92

Coefficient	RF Doove MICE 10 vs RFcont MICE 10	RF Doove MICE 10 vs CART MICE	RF Doove MICE 10 vs RF Doove MICE 100
x1	0.984	0.939	1
x2	0.987	0.925	0.998
x3	1.01	0.768 ***	0.998

3.3.3 Comparison of confidence interval length

Ratio of mean length of 95% confidence intervals (less than 1 means that the first method produces smaller confidence intervals). P values from paired sample t test. Significance level: * P <0.05, ** P <0.01, *** P <0.001.

Coefficient	RFcont MICE 10 vs parametric MICE	RFcont MICE 100 vs parametric MICE	RFcont MICE 10 vs RFcont MICE 100
x1	0.9833 ***	0.9795 ***	1.004 ***
x2	0.9835 ***	0.9816 ***	1.002 **
x3	0.9715 ***	0.9649 ***	1.007 **

Coefficient	RF Doove MICE 10 vs RFcont MICE 10	RF Doove MICE 10 vs CART MICE	RF Doove MICE 10 vs RF Doove MICE 100
x1	0.9967 ***	1.003 ***	0.9995
x2	0.9972 ***	1.002 **	0.9993
x3	0.9874 ***	1.018 ***	1

3.3.4 Comparison of confidence interval coverage

Difference between percentage coverage of 95% confidence intervals (positive means that the first method has greater coverage). P values for pairwise comparisons by McNemar's test. Significance level: * P <0.05, ** P <0.01, *** P <0.001.

Coefficient	RFcont MICE 10 vs parametric MICE	RFcont MICE 100 vs parametric MICE	RFcont MICE 10 vs RFcont MICE 100
x1	0.0	0.5	-0.5
x2	1.2 *	0.7	0.5
x3	0.3	-0.3	0.6

Coefficient	RF Doove MICE 10 vs RFcont MICE 10	RF Doove MICE 10 vs CART MICE	RF Doove MICE 10 vs RF Doove MICE 100
x1	0.5	1.4 **	0.9
x2	-0.2	0.8 *	-0.1
x3	0.1	3.7 ***	0.0

4 Discussion

In this simulation, parametric MICE using the default settings yielded a biased estimate for the coefficient for the partially observed variable x_3 . This is because the interaction between x_1 and x_2 was not included in the imputation models. The estimate using the CART or Random Forest MICE methods were less biased, more precise and had shorter confidence intervals with greater coverage. Omissions of interactions between predictors can potentially result in bias using parametric MICE even if, as in this case, the interaction is not present in the substantive model.

4.1 CART versus Random Forest MICE

CART MICE produced estimates for the x_3 coefficient that were less precise than the Random Forest MICE methods, and coverage of 95% confidence intervals was only 93%.

4.2 Comparison of Random Forest MICE methods

Coefficients estimated after imputation using CART or Random Forest MICE methods were slightly biased. The bias was statistically significant but small in magnitude. Using RFcont MICE, the x_3 coefficient was biased towards the null with 5 or 10 trees and biased away from the null with 20 or more trees; bias was minimised using 10 or 20 trees.

4.3 missForest

Parameters estimated after imputation using missForest were biased and the coverage of 95% confidence intervals was less than 95%. Failure to draw from the correct conditional distribution leads to bias and underestimation of the uncertainty when statistical models are fitted to imputed data.

4.4 Implications for further research

This simulation demonstrates a situation in which Random Forest MICE methods have an advantage over parametric MICE. Both Doove's method (RF) and our method (RFcont) performed well, and on some performance measures Doove's method was superior.

It would be useful to compare these methods in simulations based on real datasets.

5 Appendix: R code

5.1 R functions

This R code needs to be run in order to load the necessary functions before running the script (Section 5.2).

5.1.1 Data generating functions

```
makeSurv <- function(n = 2000, loghr = kLogHR){
  # Creates a survival cohort of n patients. Assumes that censoring is
  # independent of all other variables

  # x1 and x2 are random normal variables
  data <- data.frame(x1 = rnorm(n), x2 = rnorm(n))

  # Create the x3 variable
  data$x3 <- 0.5 * (data$x1 + data$x2 - data$x1 * data$x2) + rnorm(n)

  # Underlying log hazard ratio for all variables is the same
  data$y <- with(data, loghr * (x1 + x2 + x3))
  data$survtime <- rexp(n, exp(data$y))

  # Censoring - assume uniform distribution of observation times
  # up to a maximum
  obstime <- runif(nrow(data), min = 0,
    max = quantile(data$survtime, 0.5))
  data$event <- as.integer(data$survtime <= obstime)
  # Generate integer survival times
  data$time <- ceiling(100 * pmin(data$survtime, obstime))

  # Observed marginal cumulative hazard for imputation models
  data$cumhaz <- nelsonaalen(data, time, event)

  # True log hazard and survival time are not seen in the data
  # so remove them
  data$y <- NULL
  data$survtime <- NULL

  return(data)
}
<bytecode: 0x5596481f4058>

makeMarSurv <- function(data, pmissing = kPmiss){
  # Introduces missing data dependent on event indicator
  # and cumulative hazard and x1 and x2

  logistic <- function(x){
    exp(x) / (1 + exp(x))
  }

  predictions <- function(lp, n){
    # uses the vector of linear predictions (lp) from a logistic model
    # and the expected number of positive responses (n) to generate
    # a set of predictions by modifying the baseline

    trialn <- function(lptrial){
      sum(logistic(lptrial))
    }
    stepsize <- 32
```

```

lptrial <- lp
# To avoid errors due to missing linear predictors (ideally
# there should not be any missing), replace with the mean
if (any(is.na(lptrial))){
  lp[is.na(lptrial)] <- mean(lptrial, na.rm = TRUE)
}
while(abs(trialn(lptrial) - n) > 1){
  if (trialn(lptrial) > n){
    # trialn bigger than required
    lptrial <- lptrial - stepsize
  } else {
    lptrial <- lptrial + stepsize
  }
  stepsize <- stepsize / 2
}
# Generate predictions from binomial distribution
as.logical(rbinom(logical(length(lp)), 1, logistic(lptrial)))
}
data$x3[predictions(0.1 * data$x1 + 0.1 * data$x2 +
  0.1 * data$cumhaz + 0.1 * data$event, nrow(data) * pmissing)] <- NA
return(data)
}
<bytecode: 0x559645729be8>

```

5.1.2 Functions to analyse data

```

coxfull <- function(data){
  # Full data analysis
  coefs <- as.data.frame(summary(coxph(myformula, data = data))$coef)
  # return a data.frame of coefficients (est), upper and lower 95% limits
  out <- data.frame(est = coefs[, 'coef'],
    lo95 = coefs[, 'coef'] + qnorm(0.025) * coefs[, 'se(coef)'],
    hi95 = coefs[, 'coef'] + qnorm(0.975) * coefs[, 'se(coef)'],
    row.names = row.names(coefs))
  out$cover <- kLogHR >= out$lo95 & kLogHR <= out$hi95
  out
}

coximpute <- function(imputed_datasets){
  # Analyses a list of imputed datasets
  docoxmodel <- function(data){
    coxph(myformula, data = data)
  }
  mirafits <- as.mira(lapply(imputed_datasets, docoxmodel))
  coefs <- as.data.frame(summary(pool(mirafits)))
  if ('term' %in% colnames(coefs)){
    row.names(coefs) <- as.character(coefs$term)
  }
  if (!('lo 95' %in% colnames(coefs))){
    # newer version of mice
    # use normal approximation for now, as assume large sample
    # and large degrees of freedom for t distribution
    out <- data.frame(est = coefs$estimate,

```

```

        lo95 = coefs$estimate + qnorm(0.025) * coefs$std.error,
        hi95 = coefs$estimate + qnorm(0.975) * coefs$std.error,
        row.names = row.names(coefs))
  } else if ('lo 95' %in% colnames(coefs)){
    # older version of mice
    out <- data.frame(est = coefs$est,
                      lo95 = coefs[, 'lo 95'], hi95 = coefs[, 'hi 95'],
                      row.names = row.names(coefs))
  } else {
    stop('Unable to handle format of summary.mipo object')
  }
  # Whether this confidence interval contains the true hazard ratio
  out$cover <- kLogHR >= out$lo95 & kLogHR <= out$hi95
  out
}

domissf <- function(missdata, reps = NIMPS){
  # Imputation by missForest
  out <- list()
  for (i in 1:reps){
    invisible(capture.output(
      out[[i]] <- missForest(missdata)$ximp))
  }
  out
}

mice.impute.cart <- function (y, ry, x, wy = NULL, minbucket = 5, cp = 1e-04, ...)
{
  install.on.demand("rpart", ...)
  if (is.null(wy)) {
    wy <- !ry
  }
  minbucket <- max(1, minbucket)
  if (dim(x)[2] == 0) {
    x <- cbind(x, 1)
    dimnames(x) <- list(NULL, "int")
  }
  xobs <- data.frame(x[ry, , drop = FALSE])
  xmis <- data.frame(x[wy, , drop = FALSE])
  yobs <- y[ry]
  if (!is.factor(yobs)) {
    fit <- rpart::rpart(yobs ~ ., data = cbind(yobs, xobs),
                        method = "anova", control = rpart::rpart.control(minbucket = minbucket,
                                                                              cp = cp, ...))
    leafnr <- floor(as.numeric(row.names(fit$frame[fit$where,
      ])))
    fit$frame$yval <- as.numeric(row.names(fit$frame))
    nodes <- predict(object = fit, newdata = xmis)
    donor <- lapply(nodes, function(s) yobs[leafnr == s])
    impute <- vapply(seq_along(donor), function(s) sample(donor[[s]],
      1), numeric(1))
  }
  else {

```

```

cat.has.all.obs <- table(yobs) == sum(ry)
if (any(cat.has.all.obs)) {
  return(rep(levels(yobs)[cat.has.all.obs], sum(wy)))
}
xy <- cbind(yobs, xobs)
xy <- droplevels(xy)
fit <- rpart::rpart(yobs ~ ., data = xy, method = "class",
  control = rpart::rpart.control(minbucket = minbucket,
    cp = cp, ...))
nodes <- predict(object = fit, newdata = xmis)
impute <- apply(nodes, MARGIN = 1, FUN = function(s) {
  sample(colnames(nodes), size = 1, prob = s)
})
}
impute
}
<bytecode: 0x55964bdb2118>
<environment: namespace:mice>

mice.impute.rfdoove10 <- function(y, ry, x, ...){
  mice::mice.impute.rf(y = y, ry = ry, x = x, ntrees = 10)
}

mice.impute.rfdoove100 <- function(y, ry, x, ...){
  mice::mice.impute.rf(y = y, ry = ry, x = x, ntrees = 100)
}

mice.impute.rfcont5 <- function(y, ry, x, ...){
  CALIBERrfimpute::mice.impute.rfcont(
    y = y, ry = ry, x = x, ntree_cont = 5)
}

mice.impute.rfcont10 <- function(y, ry, x, ...){
  CALIBERrfimpute::mice.impute.rfcont(
    y = y, ry = ry, x = x, ntree_cont = 10)
}

mice.impute.rfcont20 <- function(y, ry, x, ...){
  CALIBERrfimpute::mice.impute.rfcont(
    y = y, ry = ry, x = x, ntree_cont = 20)
}

mice.impute.rfcont100 <- function(y, ry, x, ...){
  CALIBERrfimpute::mice.impute.rfcont(
    y = y, ry = ry, x = x, ntree_cont = 100)
}

domice <- function(misssdata, functions, reps = NIMPS){
  mids <- mice(misssdata, defaultMethod = functions,
    m = reps, visitSequence = 'monotone',
    printFlag = FALSE, maxit = 10)
  lapply(1:reps, function(x) complete(mids, x))
}

```

```
doanalysis <- function(x){
  # Creates a dataset, analyses it using different methods, and outputs
  # the result as a matrix of coefficients / SE and coverage
  data <- makeSurv(kSampleSize)
  missdata <- makeMarSurv(data)
  out <- list()
  out$full <- coxfull(data)
  out$missf <- coximpute(domissf(missdata))
  out$rf5 <- coximpute(domice(missdata, 'rfcont5'))
  out$rf10 <- coximpute(domice(missdata, 'rfcont10'))
  out$rf20 <- coximpute(domice(missdata, 'rfcont20'))
  out$rf100 <- coximpute(domice(missdata, 'rfcont100'))
  out$rfdoove10 <- coximpute(domice(missdata, 'rfdoove10'))
  out$rfdoove100 <- coximpute(domice(missdata, 'rfdoove100'))
  out$cart <- coximpute(domice(missdata, 'cart'))
  out$mice <- coximpute(domice(missdata, 'norm'))
  out
}
```

5.1.3 Functions to compare methods

```
pstar <- function(x){
  if (!is.na(x)){
    if (x < 0.001){
      '***'
    } else if (x < 0.01){
      '**'
    } else if (x < 0.05){
      '*'
    } else {
      ''
    }
  } else {
    ''
  }
}

<bytecode: 0x559645d5db00>

compareBias <- function(method1, method2){
  # Generates a table comparing bias
  # Comparison statistic is the difference in absolute bias
  # (negative means first method is better)

  compareBiasVar <- function(varname){
    # All coefficients should be kLogHR
    bias1 <- sapply(results, function(x){
      x[[method1]][varname, 'est']
    }) - kLogHR
    bias2 <- sapply(results, function(x){
      x[[method2]][varname, 'est']
    }) - kLogHR

    if (sign(mean(bias1)) == -1){
```

```

        bias1 <- -bias1
    }
    if (sign(mean(bias2)) == -1){
        bias2 <- -bias2
    }

    paste(formatC(mean(bias1) - mean(bias2), format = 'fg', digits = 3),
          pstar(t.test(bias1 - bias2)$p.value))
}

    sapply(variables, compareBiasVar)
}
<bytecode: 0x55964c6ea8d0>

compareVariance <- function(method1, method2){
    # Generates a table comparing precision between two methods
    # Comparison statistic is ratio of variance
    # (smaller means first method is better)

    compareVarianceVar <- function(varname){
        e1 <- sapply(results, function(x){
            x[[method1]][varname, 'est']
        })
        e2 <- sapply(results, function(x){
            x[[method2]][varname, 'est']
        })
        paste(formatC(var(e1) / var(e2), format = 'fg', digits = 3),
              pstar(var.test(e1, e2)$p.value))
    }

    sapply(variables, compareVarianceVar)
}
<bytecode: 0x55964c294e70>

compareCILength <- function(method1, method2){
    # Generates a table comparing coverage percentage between two methods
    # Comparison statistic is the ratio of confidence interval lengths
    # (less than 1 = first better)

    compareCILengthVar <- function(varname){
        # Paired t test for bias (difference in estimate)
        len1 <- sapply(results, function(x){
            x[[method1]][varname, 'hi95'] -
            x[[method1]][varname, 'lo95']
        })
        len2 <- sapply(results, function(x){
            x[[method2]][varname, 'hi95'] -
            x[[method2]][varname, 'lo95']
        })

        paste(formatC(mean(len1) / mean(len2),
                      format = 'fg', digits = 4),
              pstar(t.test(len1 - len2)$p.value))
    }
}

```



```

    }

    sapply(variables, compareCILengthVar)
}
<bytecode: 0x559649db1b88>

compareCoverage <- function(method1, method2){
  # Generates a table comparing coverage percentage between two methods
  # Comparison statistic is the difference in coverage
  # (positive = first better)

  compareCoverageVar <- function(varname){
    # Paired t test for bias (difference in estimate)

    cov1 <- sapply(results, function(x){
      x[[method1]][varname, 'cover']
    })
    cov2 <- sapply(results, function(x){
      x[[method2]][varname, 'cover']
    })

    paste(formatC(100 * (mean(cov1) - mean(cov2)), format = 'f',
      digits = 1),
      pstar(binom.test(c(sum(cov1 == TRUE & cov2 == FALSE),
        sum(cov1 == FALSE & cov2 == TRUE)))$p.value))
  }

  sapply(variables, compareCoverageVar)
}
<bytecode: 0x5596452993b0>

```

5.1.4 Functions to compile and display results

```

getParams <- function(coef, method){
  estimates <- sapply(results, function(x){
    x[[method]][coef, 'est']
  })
  bias <- mean(estimates) - kLogHR
  se_bias <- sd(estimates) / sqrt(length(estimates))
  mse <- mean((estimates - kLogHR) ^ 2)
  ci_len <- mean(sapply(results, function(x){
    x[[method]][coef, 'hi95'] - x[[method]][coef, 'lo95']
  })))
  ci_cov <- mean(sapply(results, function(x){
    x[[method]][coef, 'cover']
  })))
  out <- c(bias, se_bias, mse, sd(estimates), ci_len, ci_cov)
  names(out) <- c('bias', 'se_bias', 'mse', 'sd', 'ci_len', 'ci_cov')
  out
}
<bytecode: 0x559647464eb0>

showTable <- function(coef){
  methods <- c('full', 'missf', 'cart', 'rfdooove10',

```

```

      'rfdoove100', 'rf5', 'rf10', 'rf20', 'rf100', 'mice')
methodnames <- c('Full data', 'missForest', 'CART MICE',
      'RF Doove MICE 10', 'RF Doove MICE 100',
      paste('RFcont MICE', c(5, 10, 20, 100)),
      'Parametric MICE')
out <- t(sapply(methods, function(x){
  getParams(coef, x)
}))
out <- formatC(out, digits = 3, format = 'fg')
out <- rbind(c('', 'Standard', 'Mean', 'SD of', 'Mean 95%',
  '95% CI'), c('Bias', 'error of bias', 'square error', 'estimate',
  'CI length', 'coverage'), out)
out <- cbind(c('', '', methodnames), out)
rownames(out) <- NULL
print(xtable(out), floating = FALSE, include.rownames = FALSE,
  include.colnames = FALSE, hline.after = c(0, 2, nrow(out)))
}
<bytecode: 0x5596456f8cc8>

maketable <- function(comparison){
  # comparison is a function such as compareCoverage, compareBias
  compare <- cbind(comparison('rf10', 'mice'),
    comparison('rf100', 'mice'),
    comparison('rf10', 'rf100'))
  compare <- cbind(rownames(compare), compare)
  compare <- rbind(
    c('', 'RFcont MICE 10 vs', 'RFcont MICE 100 vs',
      'RFcont MICE 10 vs'),
    c('Coefficient', 'parametric MICE',
      'parametric MICE', 'RFcont MICE 100'),
    compare)
  rownames(compare) <- NULL
  print(xtable(compare), include.rownames = FALSE,
    include.colnames = FALSE, floating = FALSE,
    hline.after = c(0, 2, nrow(compare)))

  cat('\n\\vspace{1em}\\n')

  compare <- cbind(comparison('rfdoove10', 'rf10'),
    comparison('rfdoove10', 'cart'),
    comparison('rfdoove10', 'rfdoove100'))
  compare <- cbind(rownames(compare), compare)
  compare <- rbind(
    c('', 'RF Doove MICE 10 vs', 'RF Doove MICE 10 vs',
      'RF Doove MICE 10 vs'),
    c('Coefficient', 'RFcont MICE 10',
      'CART MICE', 'RF Doove MICE 100'),
    compare)
  rownames(compare) <- NULL
  print(xtable(compare), include.rownames = FALSE,
    include.colnames = FALSE, floating = FALSE,
    hline.after = c(0, 2, nrow(compare)))
}

```

<bytecode: 0x55964be110f0>

5.2 R script

Run this script after loading the functions above.

```
# Install CALIBERrfimpute if necessary:
# install.packages("CALIBERrfimpute", repos="http://R-Forge.R-project.org")
library(CALIBERrfimpute)
library(missForest)
library(survival)
library(xtable)
library(parallel) # Use parallel processing on Unix

# Initialise constants
kPmiss <- 0.2 # probability of missingness
kLogHR <- 0.5 # true log hazard ratio

# Set number of patients in simulated datasets
NPATS <- 2000

# Set number of samples
N <- 1000

# Set number of imputations
NIMPS <- 10

# Perform the simulation
results <- mclapply(1:N, doanalysis)

# Show results
showTable('x1'); showTable('x2'); showTable('x3')

# Names of the variables in the comparison
variables <- c('x1', 'x2', 'x3')

# Show comparisons between methods
maketable(compareBias)
maketable(compareVariance)
maketable(compareCILength)
maketable(compareCoverage)
```

References

- [1] Shah AD, Bartlett JW, Carpenter J, Nicholas O, Hemingway H. Comparison of Random Forest and Parametric Imputation Models for Imputing Missing Data Using MICE: A CALIBER Study. *American Journal of Epidemiology* 2014. doi: 10.1093/aje/kwt312
- [2] Doove LL, van Buuren S, Dusseldorp E. Recursive partitioning for missing data imputation in the presence of interaction effects. *Computational Statistics and Data Analysis* 2014;72:92–104. doi: 10.1016/j.csda.2013.10.025