University of Zurich[UZH]

Epidemiology, Biostatistics
and Prevention Institute,
Epidemiology Department
+41 44 634 46 34
www.ebpi.uzh.ch

Multiple Imputation using R

Sarah R Haile (sarah.haile@uzh.ch)

Version 1.0 of July 30, 2019

Contents

Multiple imputation (MI) is a common statistical method used to analyze datasets where some values are missing. In this document we describe multiple imputation briefly, and show how to perform the analysis in R. The main and extended examples show a dataset where the outcome is binary, and logistic regression is used. After that, we show shorter examples for linear regression and Cox proportional hazards regression.

For more information about MI, you might find the following other references helpful. Sterne et al. [2009] and White et al. [2011] provide overviews on MI, while van Buuren and Groothuis-Oudshoorn [2011] describes the `mice` package in R and gives examples. White and Royston [2009] specifically discusses MI when survival data are present. Several papers give good overviews of MI in epidemiology: Perkins et al. [2017], Harel et al. [2017], Pedersen et al. [2017]. For further details on the underlying framework of the R package, `mice`, described here, see also the book *Flexible Imputation of Missing Data* (FIMD) [van Buuren, 2018] (full text available online), and the accompanying vignettes (Vink and van Buuren [2019], also linked from `?mice`).

1 Basic steps

The basic steps in R are as follows:

`mice()` Impute the data. That is, make `m` copies of the original dataset and fill in the missing values.

`with()` Analyze each of the completed datasets.

`pool()` Combine the parameter estimates using Rubin's rules.

2 Complete Cases analysis

The `birthwt` dataset (from the R package `MASS`) has been adapted for the analysis described here (see code in Appendix). The variables are:

`low` indicator of birth weight less than 2.5 kg.
`bwt` birth weight in grams.
`age` mother's age in years.
`lwt` mother's weight in pounds at last menstrual period.
`race` mother's race (1 = white, 2 = black, 3 = other).
`smoke` smoking status during pregnancy.
`ptl` number of previous premature labours.
`ht` history of hypertension.
`ui` presence of uterine irritability.
`ftv` number of physician visits during the first trimester.

Using `md.pattern()` we can examine the pattern of missingness in the data. Each combination of missing variables is given a row in the output, with 1 in the row indicating observed and 0 indicating missing. Here we see that while most of the 189 subjects have complete observations [top row], many observations have a single missing variable (only one 0 in the row), and some observations have multiple variables missing (several 0s in the row). Other functions to visualize patterns in missing data are also available from the `VIM` package.

```
md.pattern(dat[, c("bwt", "age", "smoke", "race", "ftv")], plot = FALSE)

##     age smoke race bwt ftv
## 130   1     1    1   1   1  0
## 16    1     1    1   1   0  1
## 6     1     1    1   0   1  1
## 8     1     1    1   0   0  2
## 7     1     1    0   1   1  1
## 4     1     0    1   1   1  1
## 7     1     0    0   0   1  3
## 11    0     1    1   1   1  1
##      11    11   14  21  24 81
```

First, we consider a regression model using only the complete cases: predictors for birthweight using linear regression.

```
m1.cc <- lm(bwt ~ age + smoke, data = dat)
summary(m1.cc)

##
## Call:
## lm(formula = bwt ~ age + smoke, data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2040.3  -446.9    41.3   552.6  1966.2
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2926.68     282.86   10.35   <2e-16
## age             2.16      11.90    0.18    0.856
## smoke        -237.86     122.59   -1.94    0.054
##
## Residual standard error: 738 on 150 degrees of freedom
##   (36 observations deleted due to missingness)
## Multiple R-squared:  0.0248,Adjusted R-squared:  0.0118
## F-statistic:  1.9 on 2 and 150 DF,  p-value: 0.153
```

We would like to see if our results are affected by the missing values.

3 A basic example

Now, we use MI to guess what the missing variables "could have been", and then use our imputed datasets to estimate the regression coefficients again.

3.1 Impute the missing observations using `mice()`.

There are various methods used to impute missing values for multiple variables at once. We recommend using `mice()` ("Multivariate Imputation using Chained Equations"). **Before running `mice()`, check that your dataset contains only variables you will use in your analysis, or variables you think are related to missingness (see Section 8.3).**

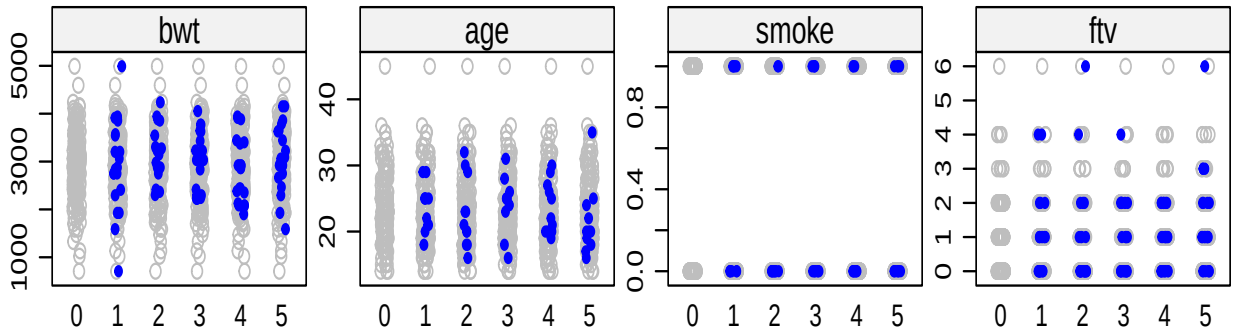
The model above considered 3 variables (`bwt, age, smoke`), and we think that `race` and `ftv` may be related to missingness, so we keep only those 5 variables. Then, we use the `mice()` command to impute the missing variables multiple times, using all other variables (including the outcome) as predictors in the imputation procedure (see Section 8.3). We also set a `seed` to ensure that we will get the same results any time we run the code. For more on which regression methods to use or how many datasets to impute see Sections 8.5 and 8.2.

```
library(mice)
small <- dat[, c("bwt", "age", "smoke", "race", "ftv")]
imp <- mice(small, m = 5, print = FALSE, seed = 12345)
imp

## Class: mids
## Number of multiple imputations:  5
## Imputation methods:
##       bwt       age     smoke      race       ftv
##     "pmm"     "pmm"     "pmm" "polyreg"     "pmm"
## PredictorMatrix:
##       bwt age smoke race ftv
## bwt     0   1     1    1   1
## age     1   0     1    1   1
## smoke   1   1     0    1   1
## race    1   1     1    0   1
## ftv     1   1     1    1   0
```

In this example, we see that `mice()` used `pmm` (predictive mean matching) to impute all the variables except `race` which used `polyreg`, a form of multinomial logistic regression. (Variables with no missing values will have no method ("") because no imputation method is needed.) Reading across the rows of the predictor matrix, we can also see that `mice()` used each of the other variables to impute both variables.

```
stripplot(imp, col=c("grey", "blue"), pch = c(1, 20))
```



Using `stripplot.mids()`[1] or `densityplot.mids()`, we can see that the imputed values are similar to the observed values for both `age` and `bmi`, indicating that imputation is probably appropriate for this analysis.

3.2 Run regression model on imputed data using `with()`.

To run a regression model with imputed data, we have to use `with()` (see `?with.mids`). Note that we use the same model formulation as above, but we leave out the `data` option. Note that we use the same imputed data to run several models, so there is no need to impute new data for every model of interest. In general we will not look at the results of `with()` directly, but instead `pool()` them first. We can however take a look at the `analyses` and get the results of each of the `m` fitted regression models.

```
m1.mi <- with(imp, lm(bwt ~ age + smoke))
t(sapply(m1.mi$analyses, coef))
```

```
##      (Intercept)  age smoke
## [1,]        2933  5.3  -324
## [2,]        2943  4.1  -254
## [3,]        3108 -4.4  -209
## [4,]        3042 -1.6  -247
## [5,]        2927  5.5  -291
```

3.3 Combine the results using `pool()`.

```
summary(pool(m1.mi), conf.int = TRUE)
```

```
##             estimate std.error statistic  df p.value 2.5 % 97.5 %
## (Intercept)   2990.5       265     11.29 109   0.000  2466   3515
## age              1.8        11      0.16  66   0.877   -21     25
## smoke         -265.3       118     -2.25  74   0.027  -500    -30
```

[1] Changing the default colors using `col` and the shapes using `pch` in `stripplot()` makes these easier to read.

The `pool()` function should work in most cases, for most types of models, as long as there is a `tidy` method for it in the `broom` package. (For this reason, multinomial logistic regression should use `multinom()` and not `mlogit()` when using `mice()`.) Note that, unlike with usual model results, there are not many additional options after MI has been performed and the results have been pooled. Typical functions like `coef()` and `predict()` are not available.

4 Extended example

Now let's suppose that we want to a) give each level of `smoke` a label instead of using 0/1 coding, b) consider `age` as categorical variable, and c) recode `ftv` by grouping together 2-6 visits, and include it in the new model.

Here, we could impute the missing data as above, and then calculate the variables we need for our regression models. The approach in this case would be as follows:

1. Recode `smoke` as a factor.
2. Impute `age`, `bwt` and `ftv` as before. The computed variables `age_cut` and `ftv2` are not included in the dataset at this point.
3. Calculate `age_cut` and `ftv2`.

Therefore the following three options should be examined more critically before running `mice()` again:

m The number of imputed datasets (default `m = 5`) . One good approach to determining `m` is to check the fraction of missing information (`fmi`) in model results, and use 100 times the highest value. For this data, the highest `fmi` is 0.29, indicating about 30 imputations. See Section 8.2.

```
check1 <- with(imp, lm(bwt ~ age + smoke + race + ftv))
pool(check1)$pooled

##             estimate  ubar    b     t dfcom  df    riv lambda   fmi
## (Intercept)     3527 74678 6735 82760   183 118 0.1082 0.0977 0.113
## age              -10   113   29   148   183  47 0.3121 0.2378 0.268
## smoke           -407 12226 3112 15961   183  48 0.3055 0.2340 0.264
## raceblack       -413 23459  193 23690   183 178 0.0099 0.0098 0.021
## raceother       -458 14718  474 15287   183 164 0.0387 0.0372 0.049
## ftv               31  2383  696  3217   183  41 0.3504 0.2595 0.293
```

`method` Using the default settings of `mice()`, the imputation method for each of the variables except `race` will be predictive mean matching `pmm`, since all variables in the dataset are numeric. If we recode `smokes` as a factor, the default method would then be `logreg`, logistic regression. See also Section 8.5.

`predictorMatrix` By default, the imputation model for each variable includes all other variables. Reading across the rows below, 1 means the predictor in that column is included in the imputation model for that row, else 0 means it is not included. The default seems to be acceptable here, since there are no variables which are in the dataset twice (e.g. `age` and `age_cut`) and no variables which are calculated from other variables in the dataset (e.g. if `height`, `weight` and `bmi` were all present).

First we fix the smoking variable, and then get the default settings from `mice()` using the option `maxit = 0`.

```
dat2 <- dat[, c("bwt", "age", "smoke", "race", "ftv")]
dat2$smoke <- factor(dat2$smoke, 0:1, c("non-smoker", "smoker"))

init <- mice(dat2, maxit = 0)
init$method

##       bwt       age     smoke      race       ftv
##     "pmm"     "pmm"  "logreg" "polyreg"     "pmm"

init$predictorMatrix

##       bwt age smoke race ftv
## bwt     0   1     1    1   1
## age     1   0     1    1   1
## smoke   1   1     0    1   1
## race    1   1     1    0   1
## ftv     1   1     1    1   0
```

After fixing the smoking variable, the default methods look correct. We then impute the missing observations, and then calculate the derived variables we want from age, birthweight, and `ftv`. This happens in 3 steps:

1. To get imputed data that look like a regular dataset, generate `complete()` imputed data,

```
imp2 <- mice(dat2, m = 30, print = FALSE, seed = 123456)
impc <- complete(imp2, "long", include = TRUE)
```

2. derive any additional variables,

   (a) age

```
impc$age_cut <- with(impc, cut(age, c(15, 20, 25, 30, 50),
                               include.lowest = TRUE, right = FALSE))
levels(impc$age_cut)

## [1] "[15,20)" "[20,25)" "[25,30)" "[30,50]"

impc$age_cut <- relevel(impc$age_cut, "[20,25)")
```

   (b) number of physician visits

```
impc$ftv2 <- impc$ftv
impc$ftv2[impc$ftv > 2] <- 2
impc$ftv2 <- factor(impc$ftv2, 2:0, c("2+ visits", "1 visit", "0 visits"))
levels(impc$ftv2)

## [1] "2+ visits" "1 visit"   "0 visits"
```

3. and finally, declare the imputed data to be `mids` again using `as.mids()`. This is the format `mice` is expecting to use for any regression analyses.

```
impc <- as.mids(impc)
```

Such an approach could be used, for example, if height and weight measurements are available, but BMI should be included in the models.

We can then run any models with the derived variables as described above.

```
m3.mi <- with(impc, lm(bwt ~ age_cut + smoke + ftv2))
summary(pool(m3.mi), conf.int = TRUE)

##                estimate std.error statistic  df p.value 2.5 % 97.5 %
## (Intercept)        3012       162     18.56 128   0.000  2691 3332.9
## age_cut[15,20)      137       146      0.94 147   0.350  -152  425.7
## age_cut[25,30)     -143       150     -0.95 152   0.344  -440  154.3
## age_cut[30,50]      179       203      0.88 108   0.381  -224  581.3
## smokesmoker        -245       126     -1.95 113   0.054  -494    4.1
## ftv21 visit          70       180      0.39 106   0.698  -287  427.7
## ftv20 visits       -100       155     -0.64 132   0.522  -406  207.2
```

There are other approaches to handling derived variables, for example, using what is often referred to as "passive imputation". That approach may be useful in more complicated situations. See FIMD Section 6.4.

5 Cox proportional hazards regression

It is recommended to include two variables related to the survival endpoint in the imputation models, the Nelson-Aalen estimate of the cumulative hazard (`nelsonaalen()`) and the event indicator, in the imputation process. For more details, see Section 8.4.

```
library(survival)
md.pattern(stanford2, plot = FALSE)

##     id time status age t5
## 157  1    1      1   1  1  0
## 27   1    1      1   1  0  1
##      0    0      0   0 27 27

stanford2$nelsonaalen <- nelsonaalen(stanford2, time, status)

imp.surv <- mice(stanford2, m = 20, print = FALSE)
m4.mi <- with(imp.surv, coxph(Surv(time, status) ~ t5 + age))
summary(pool(m4.mi), conf.int = TRUE, exponentiate = TRUE)

##     estimate std.error statistic     df p.value 2.5 % 97.5 %
## t5       1.2     0.179      0.86   5882  0.3920  0.82    1.7
## age      1.0     0.011      2.71 997026  0.0066  1.01    1.1
```

Since we used the `exponentiate = TRUE` option, the colum labelled `estimate` shows the hazard ratios.

6 Multiple imputation with clustered or longitudinal data

Multiple imputation on longitudinal or clustered data is a difficult problem. In the case of longitudinal data, the easiest approach is often to impute the data in wide format (one row per patient, with each timepoint in a different column), convert the data to long (one row per patient and timepoint), and then perform the regression analysis. An example of this type of analysis is shown below. For cluster randomized data where the random effects (for example, schools or child care centers) should be included in the imputation process, another approach should be used, see Chapter 11 in FIMD. For more examples, see Chapters 7 and 11. Below, we briefly show 3 approaches: 1) impute wide format, reshape and model in long format; 2) impute long format and model in long format using "`2l`" methods in `mice` package; and 3) impute long format and model in long format using R package `hmi`.

Potthof-Roy data: approach 1 In the data from Potthoff and Roy (1964), we have repeated measures from 4 timepoints, in wide format, to which we have added some missing values. In the usual complete cases analysis, we first reshape the data to long, plot the observations, and run a linear mixed model using `lmer()`[2]. The outcome, distance, appears to increase in a statistically significant fashion over time.

```
library(lme4)
phr <- potthoffroy
idmis <- c(3,6,9,10,13,16,23,24,27)
phr[idmis, 4] <- NA
head(phr, 3)

##   id sex d8 d10 d12 d14
## 1  1   F 21  20  22  23
## 2  2   F 21  22  24  26
## 3  3   F 20  NA  24  26

phr_long <- reshape(phr,
        idvar = "id",
        varying = c("d8", "d10", "d12", "d14"),
        v.names = "distance",
        times = c(8, 10, 12, 14),
        timevar = "age",
        direction = "long")

m5.cc <- lmer(distance ~ age + sex + (1 | id), data = phr_long)
broom::tidy(m5.cc, conf.int = TRUE)

## # A tibble: 5 x 7
##   term              estimate std.error statistic conf.low conf.high group
```

[2]Using the R package `lmerTest` adds p-values to the usual `lme4` results, but causes problem with `pool()`.

```
##   <chr>                <dbl>     <dbl>     <dbl>    <dbl>     <dbl> <chr>
## 1 (Intercept)         15.3      0.936      16.3    13.4      17.1   fixed
## 2 age                  0.666    0.0646     10.3     0.539     0.792 fixed
## 3 sexM                 2.39     0.779       3.07    0.863     3.92  fixed
## 4 sd_(Intercept).id    1.83    NA          NA      NA        NA     id
## 5 sd_Observation.R~    1.48    NA          NA      NA        NA     Residu~
```

We follow the same basic approach with multiple imputation. This is the same approach used in previous examples, but we add an extra step to reshape the data between the imputation step and the analysis step.

```
imp <- mice(phr, m = 10, print = FALSE)

## base R approach using lapply
imp_comp <- mice::complete(imp, "all")
# by default, complete uses: include = FALSE (only imputed data)
imp_long <- lapply(imp_comp, reshape,
       idvar = "id",
       varying = c("d8", "d10", "d12", "d14"),
       v.names = "distance",
       times = c(8, 10, 12, 14),
       timevar = "age",
       direction = "long")
m6.mi <- lapply(imp_long, lmer, formula =  distance ~ age + sex + (1 | id))
summary(pool(m6.mi), conf.int = TRUE)

##             estimate std.error statistic  df p.value 2.5 % 97.5 %
## (Intercept)    15.33     0.923      16.6 100   0.000 13.50   17.2
## age             0.67     0.065      10.3 100   0.000  0.54    0.8
## sexM            2.24     0.761       2.9  99   0.004  0.73    3.8
```

Pothoff-Roy data: approach 2 In an alternate approach, we try to impute and analyze the data in long format, using random effects. The method is then `2l.pmm` (PMM for 2-level data, from the `miceadds` package [Robitzsch et al., 2019]), and we mark `id` as the cluster variable with `-2` in the predictor matrix. This would also be an appropriate way to include centers in the imputation procedure for multicenter RCTs.

```
library(miceadds)
imp0 <- mice(phr_long, maxit = 0)

meth <- imp0$meth
meth["distance"] <- "2l.pmm"
meth

##       id      sex      age distance
##       ""       ""       "" "2l.pmm"
```

```
pred <- imp0$pred
pred[, "id"] <- -2
pred

##          id sex age distance
## id       -2   1   1        1
## sex      -2   0   1        1
## age      -2   1   0        1
## distance -2   1   1        0

imp <- mice(phr_long, m = 10, predictorMatrix = pred, method = meth, print = FALSE)
m7.cc <- with(imp, lmer(distance ~ age + sex + (1 | id)))
summary(pool(m7.cc), conf.int = TRUE)

##             estimate std.error statistic df p.value 2.5 % 97.5 %
## (Intercept)    15.35     0.947        16 94  0.0000 13.47  17.23
## age             0.66     0.066        10 96  0.0000  0.53   0.79
## sexM            2.33     0.769         3 99  0.0031  0.81   3.86
```

Pothoff-Roy data: approach 3 We could also use the `hmi` package (Hierarchical Multiple Imputation). If you are interested in combining more than just the fixed effects, have a look at the `hmi_pool()` function. Otherwise, `with()` and `pool()` can be used as usual.

```
library(hmi)
fm0 <- distance ~ age + sex + (1 | id)
phr_imp <- hmi(phr_long, model_formula = fm0, m = 10, maxit = 3)

## Imputation progress:
## 0%   20%  40%  60%  80%  100%
## |----|----|----|----|----|

phr_mod <- with(phr_imp, lmer(distance ~ age + sex + (1 | id)))
summary(pool(phr_mod), conf.int = TRUE)

##             estimate std.error statistic df p.value 2.5 % 97.5 %
## (Intercept)    15.33     0.941      16.3 98  0.0000 13.46  17.20
## age             0.66     0.067       9.9 99  0.0000  0.53   0.79
## sexM            2.39     0.761       3.1 99  0.0022  0.88   3.91
```

Finally, in some cluster-randomized trials, it may be that the clusters are relatively unimportant, and may be reasonably ignored in the imputation procedure (see for example, FIMD Section 7.3.2).

7 Predictions / fitted values after MI

Can I get fitted values from my pooled results? Good question! This feature is not built in to `mice()` yet[3]. There are differing approaches to how to compute the fitted values. Do we 1) pool

[3] https://github.com/stefvanbuuren/mice/issues/82

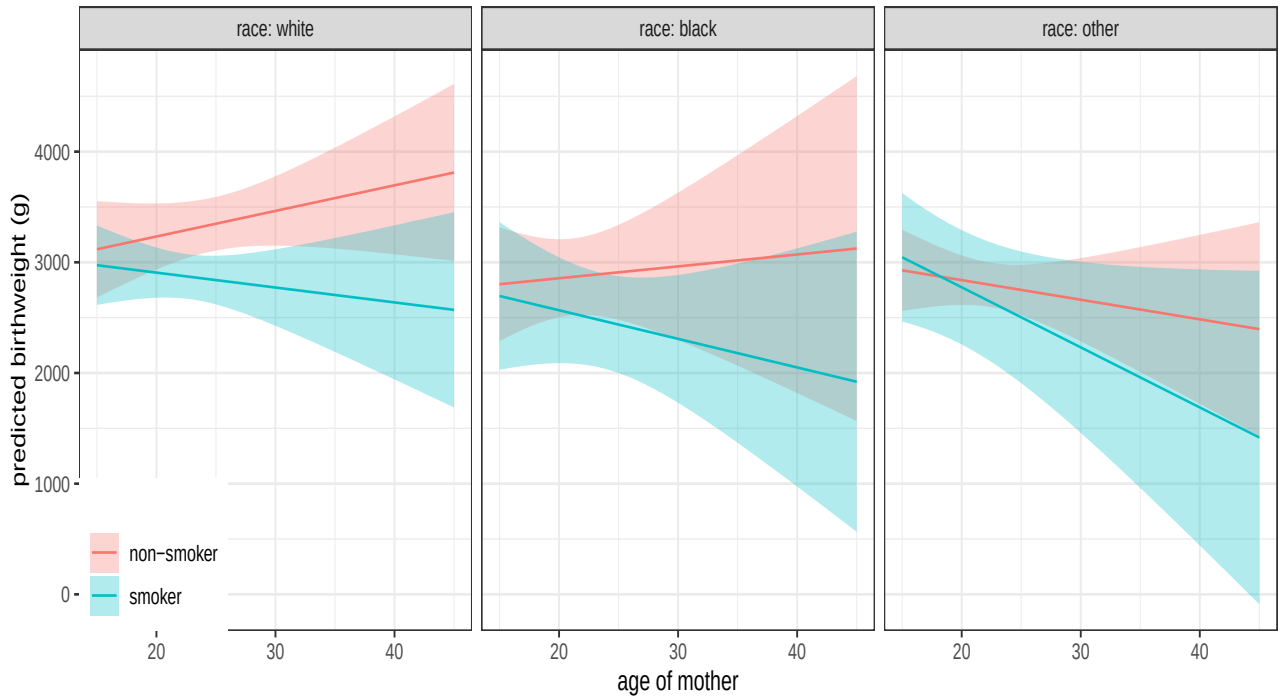
model coefficients and then get fitted values, or 2) get fitted values first and then pool them? Miles [2016] suggests that the order does not matter, and that the second approach is easier[4]. Here, we outline code for the second approach[5] which requires 2 pieces of information: a) `mod`, the set of model results, and b) `nd`, a set of new data for which you want predictions.

```
options(digits = 3)
# make some functions to easily access the pooled coefficients and [total] variance
mi_coef <- function(x){
  if(!any(class(x) == "mira")) message("x should be (unpooled) analysis results after MI")
  getqbar(pool(x))
}
mi_variance <- function(x){
  if(!any(class(x) == "mira")) message("x should be (unpooled) analysis results after MI")
  # pool() doesn't give the whole variance-covariance matrix,
  # so we compute it using Rubin's rules
  m <- length(x$analyses)
  ubar <- Reduce(`+`, lapply(x$analyses, vcov)) / m
  bvar <- var(t(sapply(x$analyses, coef)))
  tvar <- ubar + (1 + 1 / m) * bvar
  tvar
}

## probably you only do this if you want to plot them,
## so here's a somewhat realistic example...
# 0. From your imputed data...
imp <- mice(dat, m = 10, print = FALSE)
# 1. get model results
mod <- with(imp, lm(bwt ~ (age + race + smoke)^2))
# 2. make some new data for which we want fitted values
nd <- expand.grid(bwt = 3000, # the outcome can be any non-missing value
                  age = seq(15, 45, 1),
                  race = factor(c("white", "black", "other"), levels(dat$race)),
                  smoke = 0:1)
# 3. make the design matrix for the new dataset based on the model you used
mm <- model.matrix(formula(mod$analyses[[1]]), data = nd)
za <- qnorm(1 - 0.05 / 2) # for 95% CIs
fn <- function(x){x} # for logistic regression --> qlogis
# 4. get linear predictors and their SEs
nd$lp <- mm %*% mi_coef(mod)
nd$se <- sqrt(diag(mm %*% mi_variance(mod) %*% t(mm)))
nd$pred <- with(nd, fn(lp))
nd$lb <- with(nd, fn(lp - za * se))
nd$ub <- with(nd, fn(lp + za * se))
nd$smoke <- factor(nd$smoke, 0:1, c("non-smoker", "smoker"))
```

[4]However van Buuren appears to disagree with this approach, which is perhaps why it's not part of `mice` yet....
[5]Eventually, it would be good to make a function out of these steps....

```
# 5. plot the predictions
library(ggplot2)
theme_set(theme_bw())
qplot(age, pred, data = nd,
      ymin = lb, max = ub,
      color = smoke, fill = smoke, geom = "blank") +
  geom_ribbon(alpha = 0.3, color = NA) +
  geom_line() +
  facet_grid(cols = vars(race), rows = NULL, labeller = "label_both") +
  xlab("age of mother") + ylab("predicted birthweight (g)") +
  guides(color = guide_legend(""), fill = guide_legend("")) +
  theme(legend.position = c(0, 0), legend.justification = c(0, 0))
```



8 Frequently Asked Questions

8.1 Do I need to include imputed values in Table 1?

Multiple imputation is only for model fitting, and should not be used for tables of patient characteristics (which ideally should show how many missing values there are, see example below). MI is also not appropriate for use in model choice, in most cases at least, as measures of goodness-of-fit like AIC cannot be pooled. Use it only when you have a final model or a small group of final models, if you would like a "sensitivity analysis" to see how robust the coefficients in the complete cases analysis are to the missing measurements or observations.

```
library(tableone)
tab <- CreateTableOne(data = small, includeNA = TRUE)
print(tab, missing = TRUE)
```

8.2 How many imputations?

Madley-Dowd et al. [2019] have suggested that considering fraction of missing information (`fmi`) is more appropriate than proportion of incomplete cases (as suggested by White et al. [2011, Section 7.3] among others) when determining the number of imputations. However, `fmi` cannot be checked until at least some imputations have been generated and a model has been analyzed. Given that, the following procedure could be employed:

1. Set up your imputation procedure, leaving `m = 5`.
2. Check that the imputation runs through without errors.
3. Run a model which includes all the variables you want to include, and pool the results.
4. Print `summary()` of the pooled model resultes, and inspect the `fmi` column.
5. Round the highest `fmi` up to the nearest 0.05 (e.g., round 0.236 to 0.25) and multiply that value by 100 to get a new value of `m` (e.g., `m = 25`).
6. Run `mice()` again and continue with your analysis.

8.3 Which variables should be included in the imputation step?

White et al. [2011, Section 5] recommend using covariates and the outcome from the analysis models you want to run, as well as predictors of the incomplete variable(s). In general, it's good practice to create a small analysis dataset containing only the variables you need for the analysis, and use that in `mice()`, rather than the full dataset. If you plan on using any interactions in your analysis model(s), these should also be included in the imputation model (see FIMD Section 6.4.2).

8.4 How can I `mice()` when performing Cox PH regression?

White et al. [2011, Section 5] recommend using covariates and the outcome from the analysis models, as well as predictors of the incomplete variable. Further, White and Royston [2009] recommend using the

- the Nelson-Aalen estimate of the cumulative hazard (computed using `nelsonaalen()`), and
- the event indicator (for example, `died` as a 0/1 variable).

See Section 5.

8.5 Which method do I use to impute the variables?

By default, `mice()` uses the following default methods (option `defaultMethod`): predictive mean matching (`pmm`) for numeric data, logistic regression for factors with 2 levels, and multinomial logistic regression for factors with 3 levels. Note that binary variables that are still coded numerically (0/1, 1/2, etc) will have the default method "pmm", unless you recode it as a `factor`. There are many more available methods (see details in `methods(mice)`).

```
methods(mice)

##  [1] mice.impute.2l.bin       mice.impute.2l.lmer
##  [3] mice.impute.2l.norm      mice.impute.2lonly.mean
##  [5] mice.impute.2lonly.norm  mice.impute.2lonly.pmm
```

```
##  [7] mice.impute.2l.pan       mice.impute.cart
##  [9] mice.impute.jomoImpute   mice.impute.lda
## [11] mice.impute.logreg       mice.impute.logreg.boot
## [13] mice.impute.mean         mice.impute.midastouch
## [15] mice.impute.norm         mice.impute.norm.boot
## [17] mice.impute.norm.nob     mice.impute.norm.predict
## [19] mice.impute.panImpute    mice.impute.passive
## [21] mice.impute.pmm          mice.impute.polr
## [23] mice.impute.polyreg      mice.impute.quadratic
## [25] mice.impute.rf           mice.impute.ri
## [27] mice.impute.sample       mice.mids
## [29] mice.theme
## see '?methods' for accessing help and source code
```

See also Chapter 3 in FIMD for discussion of the various methods.

8.6 Can I combine MI with cross-validation or bootstrap sampling?

Generally, it is easiest if MI and resampling methods do not need to be combined, but sometimes it is not possible to avoid it. Schomaker and Heumann [2018] and Wahl et al. [2016] both explored different combinations of bootstrap sampling / cross-validation and MI. They generally suggest that a reasonable approach is to first get bootstrap samples (or for *k*-fold cross-validation, to split into testing and testing samples), and then to perform MI. Feel free to set up an appointment with me if you need to do this.

8.7 Can I pool quantities that are not regression coefficients?

Maybe! See Marshall et al. [2009] and the function `pool.scalar()`.

8.8 Can I show odds ratios after logistic regression?

Yes, try the option `exponentiate = TRUE`.

```
mod <- with(imp, glm(I(bwt < 2500) ~ age + race + smoke))
summary(pool(mod), exponentiate = TRUE, conf.int = TRUE)
```

```
##             estimate std.error statistic    df p.value 2.5 % 97.5 %
## (Intercept)    1.151   0.19979    0.7052  80.3  0.4827 0.774   1.71
## age            0.999   0.00764   -0.0751  71.4  0.9404 0.984   1.01
## raceblack      1.244   0.10219    2.1375 178.9  0.0339 1.017   1.52
## raceother      1.249   0.09242    2.4096  61.8  0.0190 1.039   1.50
## smoke          1.231   0.08568    2.4236  58.8  0.0185 1.037   1.46
```

8.9 Can I use the tidyverse (`dplyr`, etc) with `mice`?

Using a `tidyverse` approach to coding, the code in Section 4 would be the following:

```
impc <- complete(imp2, "long", include = TRUE) %>%
  mutate(age_cut = cut(age, c(15, 20, 25, 30, 50),
                               include.lowest = TRUE, right = FALSE),
         age_cut = relevel(age_cut, "[20,25)"),
         ftv2 = ifelse(ftv > 2, 2, ftv),
         ftv2 = factor(ftv2, 2:0, c("2+ visits", "1 visit", "0 visits"))) %>%
  as.mids()
```

For the Pothoff-Roy data (Section 6), we could also use this approach[6]:

```
## tidyverse approach using purrr::map
mice::complete(imp, "all") %>%
  # same reshape options as before
  map(reshape, idvar = "id",
      varying = c("d8", "d10", "d12", "d14"),
      v.names = "distance",
      times = c(8, 10, 12, 14),
      timevar = "age",
      direction = "long") %>%
  # use purrr::map() instead of mice::with()
  map(lmer, formula = distance ~ age + sex + (1 | id)) %>%
  pool() %>%
  summary(conf.int = TRUE)
```

A Recent Changes

Version 1.0

reworked the original "Multiple Imputation using Stata" document for R.

B Code to make birthweight dataset used here

```
library(mice)
set.seed(987654)
dat0 <- MASS::birthwt[, c("bwt", "age", "lwt", "race",
                          "smoke", "ptl", "ht", "ui", "ftv")]
patt <- ampute(dat0)$pattern
patt <- rbind(patt,
              c(1, 1, 0, 1, 1, 0, 0, 0, 0),
              c(0, 1, 0, 1, 1, 0, 0, 0, 0),
              c(0, 1, 0, 0, 0, 1, 1, 1, 1))
dat <- ampute(dat0, pattern = patt)$amp
dat$race <- factor(dat$race, 1:3, c("white", "black", "other"))
```

[6] It appears that `gather()` and `spread()` will soon be replaced in `tidyr` (https://tidyr.tidyverse.org/dev/articles/pivot.html), so I am using `base::reshape()`. Any similar functions should work here.

References

O Harel, EM Mitchell, NJ Perkins, SR Cole, EJ Tchetgen Tchetgen, BL Sun, and EF Schisterman. Multiple Imputation for Incomplete Data in Epidemiologic Studies. *American Journal of Epidemiology*, 187(3):576–584, 11 2017. ISSN 0002-9262. doi:10.1093/aje/kwx349.

P Madley-Dowd, R Hughes, K Tilling, and J Heron. The proportion of missing data should not be used to guide decisions on multiple imputation. *Journal of Clinical Epidemiology*, 110:63 – 73, 2019. ISSN 0895-4356. doi:10.1016/j.jclinepi.2019.02.016.

A Marshall, DG Altman, RL Holder, and Pk Royston. Combining estimates of interest in prognostic modelling studies after multiple imputation: current practice and guidelines. *BMC Medical Research Methodology*, 9(1): 57, Jul 2009. ISSN 1471-2288. doi:10.1186/1471-2288-9-57.

A Miles. Obtaining predictions from models fit to multiply imputed data. *Sociological Methods & Research*, 45 (1):175–185, 2016. doi:10.1177/0049124115610345.

AB Pedersen, EM Mikkelsen, D Cronin-Fenton, NR Kristensen, TM Pham, L Pedersen, and I Petersen. Missing data and multiple imputation in clinical epidemiological research. *Clinical Epidemiology*, 2017(9):157–166, 2017. doi:10.2147/CLEP.S129785.

NJ Perkins, SR Cole, O Harel, EJ Tchetgen Tchetgen, BL Sun, EM Mitchell, and EF Schisterman. Principled Approaches to Missing Data in Epidemiologic Studies. *American Journal of Epidemiology*, 187(3):568–575, 11 2017. ISSN 0002-9262. doi:10.1093/aje/kwx348.

A Robitzsch, S Grund, and T Henke. *miceadds: Some additional multiple imputation functions, especially for mice*, 2019. URL https://CRAN.R-project.org/package=miceadds. R package version 3.3-33.

M Schomaker and C Heumann. Bootstrap inference when using multiple imputation. *Statistics in Medicine*, 37(14):2252–2266, 2018. doi:10.1002/sim.7654.

JAC Sterne, IR White, JB Carlin, M Spratt, P Royston, MG Kenward, AM Wood, and JR Carpenter. Multiple imputation for missing data in epidemiological and clinical research: potential and pitfalls. *BMJ*, 338, 2009. ISSN 0959-8138. doi:10.1136/bmj.b2393.

S van Buuren. *Flexible Imputation of Missing Data*. 2nd edition, 2018. URL https://stefvanbuuren.name/fimd.

S van Buuren and K Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in R. *Journal of Statistical Software*, 45, 2011. doi:10.18637/jss.v045.i03.

G Vink and S van Buuren. micevignettes, 2019. URL https://www.gerkovink.com/miceVignettes/.

S Wahl, A-L Boulesteix, A Zierer, B Thorand, and MA van de Wiel. Assessment of predictive performance in incomplete data by combining internal validation and multiple imputation. *BMC Medical Research Methodology*, 16(1):144, Oct 2016. ISSN 1471-2288. doi:10.1186/s12874-016-0239-7.

IR White and P Royston. Imputing missing covariate values for the Cox model. *Statistics in Medicine*, 28(15): 1982–1998, 2009. ISSN 1097-0258. doi:10.1002/sim.3618.

IR White, P Royston, and AM Wood. Multiple imputation using chained equations: Issues and guidance for practice. *Statistics in Medicine*, 30(4):377–399, 2011. ISSN 1097-0258. doi:10.1002/sim.4067.