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 1.2 of September 7, 2023

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. Code will be provided for all examples, so we load a few R packages here.

```
library(tidyverse)
library(broom)
library(mice)
```

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.

Important The MI steps described here work well to fit a single model or a small number of models, so that you can interpret the coefficients. 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. See FAQ 8.9 for more on how to combine MI with cross-validation or bootstrap sampling.

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 [n = 124, 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 bwt ftv smoke race
## 124   1   1   1     1    1   0
## 6     1   1   1     1    0   1
## 5     1   1   1     0    1   1
## 8     1   1   1     0    0   2
## 12    1   1   0     1    1   1
## 5     1   1   0     0    0   3
## 12    1   0   1     1    1   1
## 5     1   0   1     0    0   3
## 4     1   0   0     1    1   2
## 2     1   0   0     0    0   4
## 6     0   1   1     1    1   1
##       6  23  23    25   26 103
```

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)
tidy(m1.cc, conf.int = TRUE) %>%
  mutate(p.value = format.pval(p.value, eps = 0.0001))
```

```
## # A tibble: 3 x 7
##   term        estimate std.error statistic p.value conf.low conf.high
##   <chr>          <dbl>     <dbl>     <dbl> <chr>      <dbl>     <dbl>
## 1 (Intercept)  2.88       0.274     10.5   <0.0001   2.34      3.43
## 2 age          0.00549    0.0116     0.475 0.636    -0.0174    0.0284
## 3 smoke       -0.396      0.120     -3.31  0.001    -0.632    -0.160
```

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 8.4).

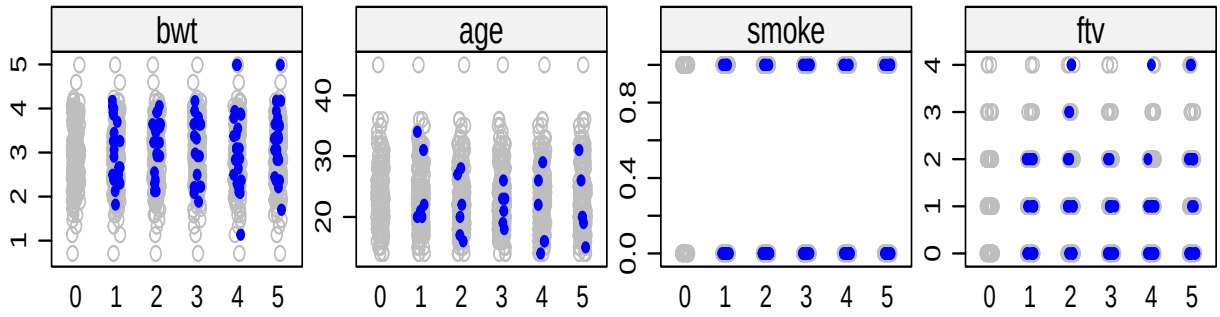
The model above considered 3 variables (`bwt, age, smoke`), and we think that `race` and `ftv` may be related to missingness, so we keep only those 5 variables. Then, we use the `mice()` command to impute the missing variables multiple times, using all other variables (including the outcome) as predictors in the imputation procedure (see Section 8.4). 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 8.6 and 8.3.

```
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` [van Buuren, 2018, Section 3.4] 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 `bwt`, 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. As we use the same imputed data to run several models, 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))
map(m1.mi$analyses, coef)

## [[1]]
## (Intercept)         age       smoke
##       2.747       0.016      -0.381
##
## [[2]]
## (Intercept)         age       smoke
##       2.841       0.012      -0.384
##
## [[3]]
## (Intercept)         age       smoke
##       2.745       0.016      -0.363
##
## [[4]]
## (Intercept)         age       smoke
##      2.9597      0.0079     -0.4629
##
## [[5]]
```

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

```
## (Intercept)         age       smoke
##       2.795       0.016      -0.453
```

3.3 Combine the results using `pool()`.

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

##          term estimate std.error statistic df p.value   2.5 % 97.5 %
## 1 (Intercept)    2.818      0.25      11.1 86 3.2e-18  2.3119  3.323
## 2         age    0.013      0.01       1.3 94 2.0e-01 -0.0074  0.034
## 3       smoke   -0.409      0.12      -3.5 65 7.9e-04 -0.6406 -0.177
```

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` or `broom.mixed` package[2]. 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 0.27, indicating about 30 imputations. See Section 8.3.

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

`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 8.6.

[2]Check the list available at `vignette("available-methods")` or `https://cran.r-project.org/web/packages/broom/vignettes/available-methods.html` and confirm that `broom` is up to date.

`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 %>%
  select(bwt, age, smoke, race, ftv) %>%
  mutate(smoke = factor(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
   (b) number of physician 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 <- impc %>%
  mutate(age_cut = cut(age, c(15, 20, 25, 30, 50),
                       include.lowest = TRUE, right = FALSE),
```

```
        age_cut = relevel(age_cut, "[20,25)")) %>%
    mutate(ftv2 = ifelse(ftv > 2, 2, ftv),
        ftv2 = factor(ftv2, 2:0, c("2+ visits", "1 visit", "0 visits"))) %>%
  as.mids
```

Such an approach could be used, for example, if height and weight measurements are available, but BMI should be included in the models. It could also be used if we want to impute `bwt` and compute a binary variable `lwt` if `bwt` is at least 2.5kg.

We can then run any models with the derived variables as described above. The fraction of missing information, `fmi`, ranges from 0.12 to 0.27, indicating that approximately 30 imputations should be sufficient in this case.

```
m3.mi <- with(impc, lm(bwt ~ age_cut + smoke + ftv2))
pool(m3.mi)

## Class: mipo    m = 30
##             term  m estimate  ubar      b     t dfcom  df   riv lambda  fmi
## 1    (Intercept) 30    3.118 0.020 0.0040 0.025   179 129 0.203  0.169 0.18
## 2 age_cut[15,20) 30    0.040 0.018 0.0013 0.019   179 160 0.074  0.069 0.08
## 3 age_cut[25,30) 30   -0.060 0.020 0.0040 0.024   179 127 0.209  0.173 0.19
## 4 age_cut[30,50] 30    0.214 0.028 0.0044 0.033   179 138 0.162  0.139 0.15
## 5    smokesmoker 30   -0.371 0.012 0.0034 0.015   179 109 0.300  0.231 0.24
## 6    ftv21 visit 30    0.057 0.024 0.0052 0.029   179 123 0.228  0.186 0.20
## 7   ftv20 visits 30   -0.133 0.019 0.0042 0.024   179 124 0.224  0.183 0.20

summary(pool(m3.mi), conf.int = TRUE)

##             term estimate std.error statistic  df p.value 2.5 % 97.5 %
## 1    (Intercept)    3.118      0.16     19.92 129 4.2e-41  2.81   3.43
## 2 age_cut[15,20)    0.040      0.14      0.29 160 7.8e-01 -0.23   0.31
## 3 age_cut[25,30)   -0.060      0.15     -0.39 127 7.0e-01 -0.36   0.25
## 4 age_cut[30,50]    0.214      0.18      1.19 138 2.4e-01 -0.14   0.57
## 5    smokesmoker   -0.371      0.12     -3.02 109 3.1e-03 -0.61  -0.13
## 6    ftv21 visit    0.057      0.17      0.33 123 7.4e-01 -0.28   0.39
## 7   ftv20 visits   -0.133      0.15     -0.87 124 3.9e-01 -0.44   0.17
```

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 [van Buuren, 2018, Section 6.4].

5 Cox proportional hazards 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).

```
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)

##   term estimate std.error statistic  df p.value 2.5 % 97.5 %
## 1   t5      1.2     0.184      0.82  96  0.4168  0.81    1.7
## 2  age      1.0     0.011      2.72 109  0.0076  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 [van Buuren, 2018, Chapter 11]. For more examples, see [van Buuren, 2018, Chapters 7 and 11]. Below, we briefly show 2 approaches: 1) impute in wide format, reshape and model in long format; and 2) impute in long format and model in long format using `"2l"` methods in `mice` package.

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()`[3]. The outcome, distance, appears to increase in a statistically significant fashion with age.

```
library(lme4)
library(broom.mixed)
phr <- potthoffroy
idmis <- c(3,6,9,10,13,16,23,24,27)
```

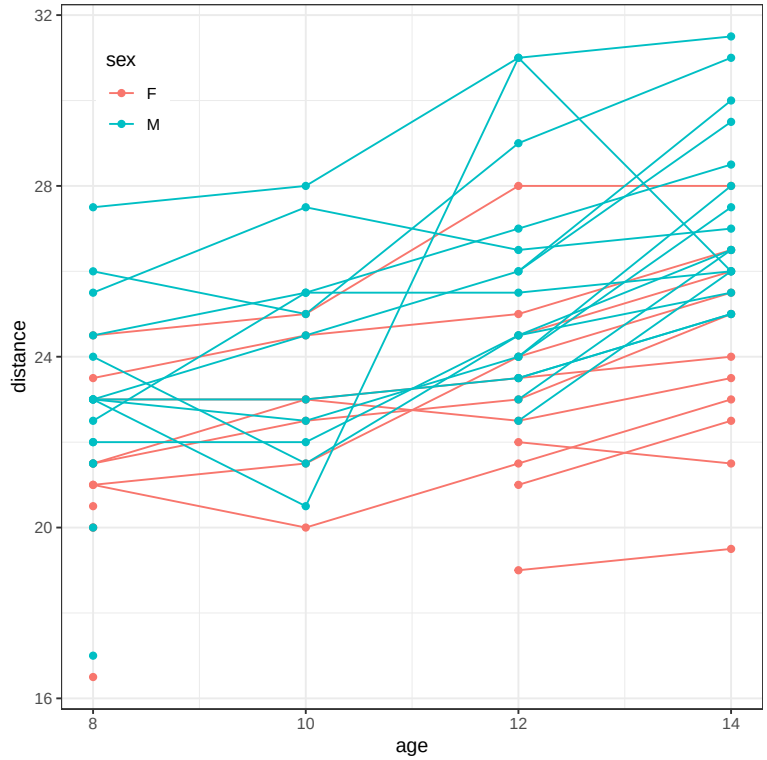
[3]Using the R package `lmerTest` adds p-values to the usual `lme4` results, but causes problem with `pool()`.

```
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 <- phr %>%
  pivot_longer(d8:d14,
               values_to = "distance",
               names_to = "age",
               names_pattern = "d(.*)",
               names_transform = as.numeric)

ggplot(aes(age, distance,
           group = id, color = sex), data = phr_long) +
  geom_line()  +
  geom_point() +
  theme_bw() +
  theme(legend.position = c(0.05, 0.95),
        legend.justification = c(0, 1))
```




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

## # A tibble: 5 x 8
```

```
##   effect   group    term          estimate std.error statistic conf.low conf.high
##   <chr>    <chr>    <chr>            <dbl>     <dbl>     <dbl>    <dbl>     <dbl>
## 1 fixed    <NA>     (Intercept)    15.3      0.936      16.3    13.4      17.1
## 2 fixed    <NA>     age             0.666    0.0646     10.3     0.539     0.792
## 3 fixed    <NA>     sexM            2.39     0.779       3.07    0.863     3.92
## 4 ran_pars id       sd__(Interc~    1.83    NA          NA      NA        NA
## 5 ran_pars Residual sd__Observa~    1.48    NA          NA      NA        NA
```

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)

imp_comp <- mice::complete(imp, "all", include = TRUE)
# by default, complete uses: include = FALSE (only imputed data)
imp_long <- map(imp_comp, ~ pivot_longer(., d8:d14,
                values_to = "distance",
                names_to = "age",
                names_pattern = "d(.*)",
                names_transform = as.numeric))

m6.mi <- map(imp_long, lmer, formula =  distance ~ age + sex + (1 | id))
summary(pool(m6.mi), conf.int = TRUE)

##          term estimate std.error statistic df p.value 2.5 % 97.5 %
## 1 (Intercept)    15.33     0.938        16 90 1.0e-28 13.47   17.2
## 2         age     0.67     0.066        10 91 2.4e-16  0.53    0.8
## 3        sexM     2.25     0.756         3 91 3.7e-03  0.75    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)

##          term estimate std.error statistic df p.value 2.5 % 97.5 %
## 1 (Intercept)    15.32     0.938        16 95 2.4e-29 13.46  17.19
## 2         age     0.66     0.065        10 97 4.8e-17  0.54   0.79
## 3        sexM     2.30     0.773         3 99 3.6e-03  0.77   3.84
```

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 [van Buuren, 2018, 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()`[4]. There are differing approaches to how to compute the fitted values. Do we 1) pool 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[5]. Here, we outline code for the second approach 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.

0. some general code

```
# Some code for both approaches...
options(digits = 3)
za <- qnorm(1 - 0.05 / 2) # for 95% CIs
fn <- function(x){x} # for logistic regression --> qlogis

imp <- mice(dat, m = 10, print = FALSE)
# 1. get model results
```

1. pool model coefficients and then get fitted values

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

[5] However van Buuren appears to disagree with this approach, which is perhaps why it's not part of `mice`....

```
mod <- with(imp, lm(bwt ~ (age + smoke)^2))
(m <- length(mod$analyses))

## [1] 10

# 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),
                  ftv = 0:3,
                  smoke = 0:1)
# save design matrix
mm <- model.matrix(formula(mod$analyses[[1]]), data = nd)
# get linear predictors and their SEs
mi_coef <- getqbar(pool(mod)) # pooled coefficients
ubar <- Reduce(`+`, lapply(mod$analyses, vcov)) / m
bvar <- var(t(sapply(mod$analyses, coef)))
mi_variance <- ubar + (1 + 1 / m) * bvar
nd$lp <- mm %*% mi_coef
nd$se <- sqrt(diag(mm %*% mi_variance %*% t(mm)))
nd$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"))

nd1 <- nd
```

2. get fitted values first and then pool them

```
# get fitted values
mod <- with(imp, lm(bwt ~ (age + smoke)^2))
(m <- length(mod$analyses))

## [1] 10

nd <- expand.grid(bwt = 3000, # the outcome can be any non-missing value
                  age = seq(15, 45, 1),
                  ftv = 0:3,
                  smoke = 0:1)
mm <- model.matrix(formula(mod$analyses[[1]]), data = nd)

modpred <- map(mod$analyses, predict, newdata = nd, se.fit = TRUE)
predmean <- map(modpred, ~ .$fit)
predvar <- map(modpred, ~ .$se.fit ^ 2)
nd <- nd %>%
  as_tibble() %>%
  mutate(m = m,
```

```
          .id = 1:n()) %>%
  uncount(m) %>%
  group_by(.id) %>%
  mutate(.imp = 1:n()) %>%
  ungroup() %>%
  arrange(.imp, .id) %>%
  mutate(q = unlist(predmean),
         u = unlist(predvar))

nd <- nd %>%
  arrange(.id, .imp) %>%
  as_tibble()

# B: pool them
nd <- nd %>%
  group_by(.id) %>%
  nest() %>%
  summarize(smoke = map_int(data, ~ .$smoke[1]),
            age = map_int(data, ~ .$age[1]),
            pool = map(data, ~ pool.scalar(.$q, .$u)),
            qbar = map_dbl(pool, ~ .$qbar),
            totalvar = map_dbl(pool, ~ .$t)) %>%
  mutate(lb = qbar - za * sqrt(totalvar),
         ub = qbar + za * sqrt(totalvar)) %>%
  mutate(smoke = factor(smoke, 0:1, c("non-smoker", "smoker")))

nd2 <- nd
```

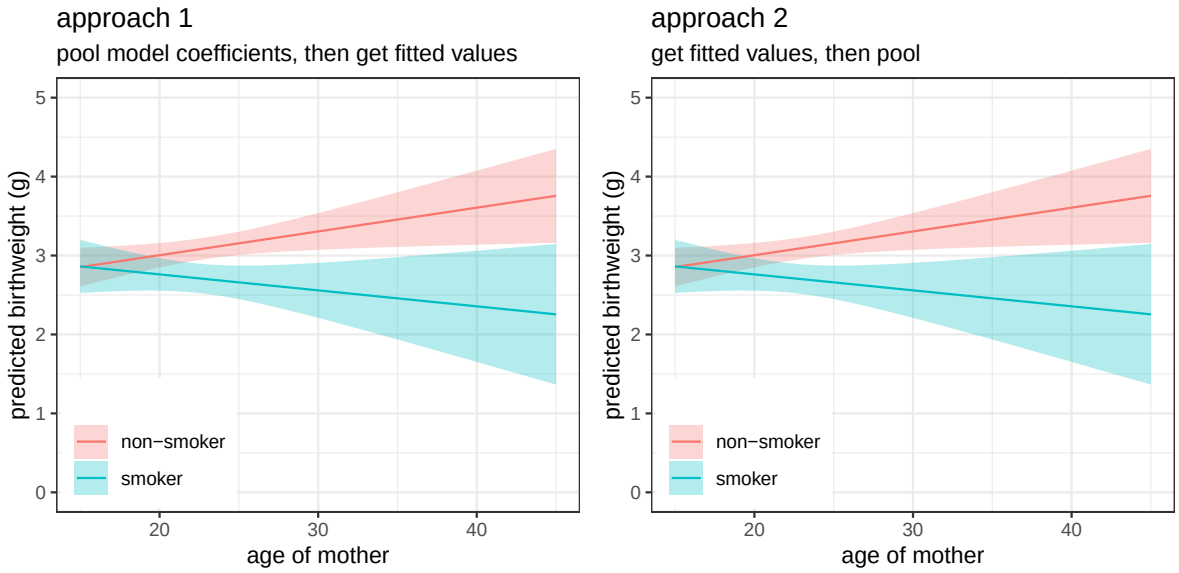
3. Plot the results.

```
library(patchwork)
# 5. plot the predictions
theme_set(theme_bw())
p1 <- ggplot(aes(age, pred,
      ymin = lb, ymax = ub,
      color = smoke, fill = smoke),
      data = nd1) +
  geom_ribbon(alpha = 0.3, color = NA) +
  geom_line() +
  xlab("age of mother") + ylab("predicted birthweight (g)") +
  ggtitle("approach 1", subtitle = "pool model coefficients, then get fitted values") +
  ylim(0, 5) +
  guides(color = guide_legend(""), fill = guide_legend("")) +
  theme(legend.position = c(0.01, 0.01), legend.justification = c(0, 0))

p2 <- ggplot(aes(age, qbar,
```

```
      ymin = lb, ymax = ub,
      color = smoke, fill = smoke),
      data = nd2) +
  geom_ribbon(alpha = 0.3, color = NA) +
  geom_line() +
  xlab("age of mother") + ylab("predicted birthweight (g)") +
  ggtitle("approach 2", subtitle = "get fitted values, then pool") +
  ylim(0, 5) +
  guides(color = guide_legend(""), fill = guide_legend("")) +
  theme(legend.position = c(0.01, 0.01), legend.justification = c(0, 0))

p1 + p2
```



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).

```
library(gtsummary)
small %>%
  tbl_summary(missing = "ifany",
              missing_text = "(missing)",
              type = list("ftv" ~ "continuous"))
```

8.2 How do I report my MI analysis?

Beyond what can be found in the relevant CONSORT (randomized trials) or STROBE (observational studies) guidelines, Box 2 in Sterne et al. [2009] has a good description of how results should be reported if MI was used in the analysis.

8.3 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 results, 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.

A two-stage approach for determining the number of imputations was proposed by von Hippel [2018] imputation, based on `fmi` and between imputation variability. There is an R function which implements that method at `https://github.com/josherrickson/howManyImputations`. (You can either install the package as done here, or just copy the one relevant function to your code.) For the previous example, the function can be used as follows:

```
#devtools::install_github("josherrickson/howManyImputations")
library(howManyImputations)
how_many_imputations(mod, cv = 0.05)

## [1] 64
```

8.4 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 [van Buuren, 2018, Section 6.4.2]).

8.5 Can I show odds ratios after logistic regression?

Yes, try the option `exponentiate = TRUE`.

```
mod <- with(imp, glm(I(bwt < 2.5) ~ age + race + smoke,
                     family = binomial))
summary(pool(mod), exponentiate = TRUE, conf.int = TRUE)

##          term estimate std.error statistic    df p.value  2.5 % 97.5 %
## 1 (Intercept)    0.143    0.9933    -1.960  85.2 0.05329 0.0198   1.03
## 2         age    0.992    0.0397    -0.198  60.1 0.84347 0.9164   1.07
## 3   raceblack    3.573    0.6290     2.025  50.4 0.04822 1.0104  12.64
## 4   raceother    4.089    0.4549     3.096 116.4 0.00246 1.6610  10.07
## 5       smoke    4.210    0.4553     3.157  61.8 0.00246 1.6945  10.46
```

8.6 Which method do I use to impute the variables?

By default, `mice()` uses the following default methods (option `defaultMethod`): predictive mean matching ('pmm') [van Buuren, 2018, Section 3.4]) for numeric data, logistic regression for factors with 2 levels, and multinomial logistic regression for factors with 3 or more 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 also the `miceadds` package. van Buuren [2018, Chapter 3] has a good discussion of the various methods.

```
apropos("mice.impute")

##  [1] "mice.impute.2l.bin"              "mice.impute.2l.lmer"
##  [3] "mice.impute.2l.norm"             "mice.impute.2l.pan"
##  [5] "mice.impute.2lonly.mean"         "mice.impute.2lonly.norm"
##  [7] "mice.impute.2lonly.pmm"          "mice.impute.cart"
##  [9] "mice.impute.jomoImpute"          "mice.impute.lasso.logreg"
## [11] "mice.impute.lasso.norm"          "mice.impute.lasso.select.logreg"
## [13] "mice.impute.lasso.select.norm"   "mice.impute.lda"
## [15] "mice.impute.logreg"              "mice.impute.logreg.boot"
## [17] "mice.impute.mean"                "mice.impute.midastouch"
## [19] "mice.impute.mnar.logreg"         "mice.impute.mnar.norm"
## [21] "mice.impute.mpmm"                "mice.impute.norm"
## [23] "mice.impute.norm.boot"           "mice.impute.norm.nob"
## [25] "mice.impute.norm.predict"        "mice.impute.panImpute"
```

```
## [27] "mice.impute.passive"            "mice.impute.pmm"
## [29] "mice.impute.polr"               "mice.impute.polyreg"
## [31] "mice.impute.quadratic"          "mice.impute.rf"
## [33] "mice.impute.ri"                 "mice.impute.sample"
```

8.7 Can I pool quantities that are not regression coefficients?

Maybe! See Marshall et al. [2009] and the function `pool.scalar()`. An alternative approach is to report median and range of the coefficients.

```
mod$analyses %>%
 map(tidy) %>%
 bind_rows() %>%
 group_by(term) %>%
 summarize(median = median(estimate),
           min = min(estimate),
           max = max(estimate))
```

```
## # A tibble: 5 x 4
##   term          median     min     max
##   <chr>          <dbl>   <dbl>   <dbl>
## 1 (Intercept) -1.97    -2.69   -1.17
## 2 age         -0.00448 -0.0392  0.0284
## 3 raceblack    1.36     0.802   1.69
## 4 raceother    1.37     1.25    1.66
## 5 smoke        1.54     1.10    1.68
```

8.8 Can I combine MI with model selection?

MI is not appropriate for use in model choice, in most cases at least, as measures of goodness-of-fit like AIC cannot be pooled. Wood et al. describe a number of possible approaches to combining MI with model selection, recommending the use of the Wald tests from pooled analysis after MI. Some approaches to stepwise model selection along with MI are described in e.g. van Buuren [2018, Section 5.4.2] (though I would recommend using `MASS::stepAIC` instead).

8.9 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 training and testing samples), and then to perform MI. Feel free to set up an appointment with me if you need to do this.

8.10 Can I use parallel processing to run `mice` on a big dataset or to create many imputations?

Starting with version 3.15.0 of the `mice` package, a new function `futuremice` takes advantage of parallel process features provided by the `future` and `furrr` packages [Bengtsson, 2021, Vaughan and Dancho, 2022]. See also the vignette for `futuremice` Vink and van Buuren [2019].

For reproducible results, it is recommended to set the option `parallelseed`. It should also be noted that `future.plan = "multisession"` is the only option available in Windows, but `future.plan = "multicore"` could be used with other operating systems (see `?future::plan`). These options are not available while running R interactively in RStudio, so the times given below are from a run of the same code using `Rscript` in batch mode (`?Rscript`).

```
library(future) # future and furrr will be loaded automatically with the mice
library(furrr)  # package, but to make it clear which packages are being used
                # I have added these here
library(tictoc) # tictoc reports processing time

tic()
imp_seq <- mice(dat, m = 200, seed = 12345, printFlag = FALSE)
toc()
# 30.525 sec elapsed
tic()
imp_parallel <- futuremice(dat, m = 200, n.core = 6,
                           parallelseed = 12345,
                           future.plan = "multisession")
toc()
# 21.269 sec elapsed
parallel::detectCores()
# [1] 8
tic()
imp_parallel <- futuremice(dat, m = 200, n.core = 6,
                           parallelseed = 12345,
                           future.plan = "multicore")
toc()
# 11.413 sec elapsed

fit <- with(imp_parallel, lm(bwt ~ age * smoke + ftv))
summary(pool(fit), conf.int = TRUE) %>%
  mutate(p.value = format.pval(p.value, eps = 0.0001))
#          term estimate std.error statistic  df p.value    2.5 % 97.5 %
# 1 (Intercept)    2.448     0.311      7.88 160  <1e-04  1.83494 3.0619
# 2         age    0.026     0.013      1.99 157    0.05  0.00025 0.0527
# 3       smoke    0.588     0.551      1.07 130    0.29 -0.50216 1.6776
# 4         ftv    0.041     0.060      0.68 137    0.50 -0.07830 0.1599
# 5   age:smoke   -0.042     0.024     -1.77 123    0.08 -0.08937 0.0051
```

References

Henrik Bengtsson. A unifying framework for parallel and distributed processing in R using futures. *The R Journal*, 13(2):208–227, 2021. doi:10.32614/RJ-2021-048. URL https://doi.org/10.32614/RJ-2021-048.

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.

Davis Vaughan and Matt Dancho. *furrr: Apply Mapping Functions in Parallel using Futures*, 2022. URL https://CRAN.R-project.org/package=furrr. R package version 0.3.1.

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

Paul von Hippel. How many imputations do you need? a two-stage calculation using a quadratic rule. *Sociological Methods & Research*, 2018. doi:10.1177/0049124117747303. URL https://statisticalhorizons.com/how-many-imputations.

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.

AM Wood, IR White, and P Royston. How should variable selection be performed with multiply imputed data? 27(17):3227–46. doi:10.1002/sim.3177.

A Recent Changes

Version 1.0

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

Version 1.1

Added the von Hippel [2018] method of determining an appropriate number of imputations. Added some comments on reporting analysis that uses MI.

Version 1.2

Removed the hierarchical data example using `hmi`, which is no longer on CRAN. Added a FAQ about parallel processing. Changed the code slightly to reflect the use of tidyverse, for example: `pivot_longer` rather than `reshape`.

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)$pattern
patt <- rbind(patt,
              c(1, 1, 1, 1, 1, 1, 0, 1, 0),
              c(0, 1, 0, 1, 1, 1, 0, 1, 0),
              c(1, 1, 1, 1, 1, 0, 1, 0, 1),
              c(0, 1, 0, 1, 1, 0, 1, 0, 1),
              c(0, 1, 0, 0, 0, 1, 1, 0, 0),
              c(1, 1, 1, 0, 0, 1, 1, 0, 0),
              c(0, 1, 0, 0, 0, 0, 0, 1, 1),
              c(1, 1, 1, 0, 0, 0, 0, 1, 1))
dat <- ampute(dat0, pattern = patt)$amp
dat$race <- factor(dat$race, 1:3, c("white", "black", "other"))
dat$bwt <- dat$bwt / 1000 # convert g to kg
```

C R packages used

```
R.version.string

## [1] "R version 4.3.1 (2023-06-16)"

base_pkgs <- sessionInfo()$basePkgs
base_pkgs

## [1] "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"
## [7] "base"

pkgname <- map_chr(sessionInfo()$otherPkgs, ~ .$Package)
pkgver <- map_chr(sessionInfo()$otherPkgs, ~ .$Version)
other_pkgs <- paste(pkgname, " (", pkgver, ")", sep = "")
other_pkgs

##  [1] "howManyImputations (0.2.4.9001)" "patchwork (1.1.3)"
##  [3] "broom.mixed (0.2.9.4)"           "lme4 (1.1-34)"
##  [5] "Matrix (1.6-0)"                  "survival (3.5-5)"
##  [7] "mice (3.16.0)"                   "broom (1.0.5)"
##  [9] "lubridate (1.9.2)"               "forcats (1.0.0)"
## [11] "stringr (1.5.0)"                 "dplyr (1.1.3)"
## [13] "purrr (1.0.2)"                   "readr (2.1.4)"
## [15] "tidyr (1.3.0)"                   "tibble (3.2.1)"
## [17] "ggplot2 (3.4.3)"                 "tidyverse (2.0.0)"
## [19] "knitr (1.43)"
```

This document was generated on 2023-09-07 at 13:38.