Missing Data: Some Foundational Concepts and Applied Strategies

Ashley I Naimi

Fall 2023

Contents

Learning Objectives

- Articulate the distinctions between MCAR, MAR, and NMAR missing data.
- Describe the relationships between MCAR, MAR, and NMAR, and marginal versus conditional exchangeability.
- Describe the difference between monotone and nonmonotone missing data.
- Articulate the distinction between complete case analysis, single imputation, and multiple imputation.
- Be able to deploy MICE in the R package.

1 Introduction

Missing data are everywhere, and range in their complexity.

In this lecture, we'll cover what we mean when we say "missing data," how it relates to concepts such as confounding (through exchangeability), what the missing data types are and what they mean, and solutions that have been proposed to address them.

2 Missing Data

Simply stated, missing data arise when some values of a particular variable in a dataset are unknown. The primary concern is that, depending upon the missingness pattern, these missing data may lead to bias in our effect estimates, summary statistics, statistical tests, or other summary measures of interest (Figure 1).

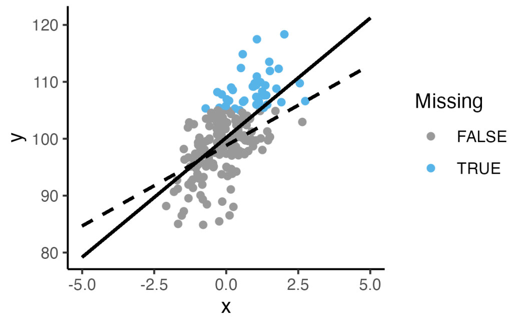


Figure 1: Illustration of the impact of missing data in a simple setting. Solid black line represents the association between X and Y when there are no missing data. Dashed black line represents the association between X and Y in teh absence of the light blue points in the figure.

3 Types of Missingess: Independence Patterns

Generally, whether or not missing data has an impact on the results of a given analysis depends entirely on its type. Missing data can be classified into three types:

- Missing Completely at Random
- Missing at Random

- Not Missing at Random

Data are considered missing completely at random (MCAR) if the probability of missingness does not depend on any other variables, either measured or unmeasured.

Data are considered missing at random (MAR) if the probability of missingness depends on other variables, AND these other variables are measured and available in the data.

Data are considered not missing at random if the probabilty of missingness depends on other variables, AND these other variables are NOT measured, and thus not available in the data.

These missing data patterns are comparable to the same concepts we encountered when introduced to exchangeability. Briefly, exchangeabilty is met when the expsoure X is independent of the potential outcomes Y^x.

$$E(Y^x \mid X) = E(Y^x)$$

When true, this equation is defined as marginal exchangeability, because we need not condition on any other variables to make the statement hold. Marginal exchangeability holds (in expectation) in an ideal randomized trial, because randomization creates an independence between the exposure assignment mechanism and all other variables (measured or unmeasured) that may be associated with the outcome.

The counterpart to marginal exchangeabilty in the missing data context is **missing completely at random.** For example, consider that we have a missingness indicator R_x, where $R_x = 1$ if an observation is missing information on the exposure status X, and $R_x = 0$ if an observation is NOT missing expsoure information. In this case, we can assume missing at random if:

$$E(Y^x \mid X, R = 0) = E(Y^x \mid X)$$

Alternatively, if we need to adjust for a set of variables C in order for the exchangeability assumption to hold, we have conditional exchangeability. Recall that, in the context of a randomized trial, conditional exchangeability arises when we randomize the exposure with a different probability conditional on some factor C, and that this factor C is also associated with the outcome. In this case, we need to condition on C to regain an independence between the expsoure and potential outcomes:

$$E(Y^x \mid X, C) = E(Y^x \mid C)$$

The counterpart to conditional exchangeability in the missing data context is **missing at random.** In our previous example, the probability of expsoure missingness may depend on certain variables Z. In this case, we would have to condition on Z to acheive independence between the missingness indicator R_x and the potential outcomes Y^x:

$$E(Y^x \mid X, Z, R = 0) = E(Y^x \mid X, Z)$$

Finally, we the case where exchangeability does not hold. This situation might be encountered, for example, in the presence of unmeasured confounding. Meaning, there are variables C that we require to make the independence between the potential outcome and the observed exposure hold, but we have not measured them. In this context of missing data, if we have variables Z that predict both the probability of missingness and the potential outcomes, and we have not measured them, we have **not missing at random.**

4 Types of Missingness: Data Patterns

A second set of patterns that are important to understand in the context of missing data is the notion of monotone versus nonmonotone missingness. These patterns arise when there is a dataset with many variables, and more than one variable is missing information. In this setting, **how** the missing values are related to each other across all observations in the data matters.

These patterns matter when it comes to dealing with missing data. For example, in the monotone case, one can employ a computationally simpler approach compared to when the missing data are distributed nonmonotone. However, when missing data are present, methods to impute nonmonotone data will work just as well when data are actually monotone (though the converse is not true). For this reason, we focus here on methods for nonmonotone data, specifically, multiple imputation via chained equations (MICE).

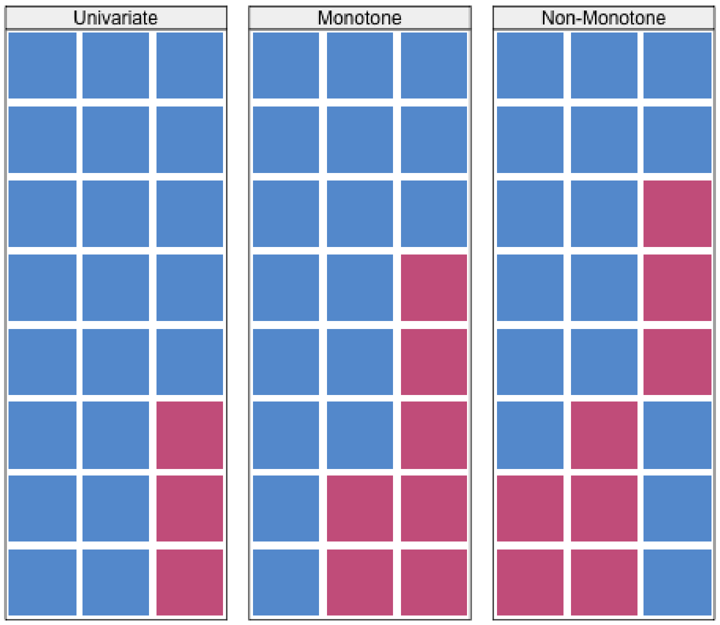


Figure 2: Example missing data patterns: univariate, monotone, and nonmonotone missingness. Figure modified from van Buuren (2018) chapter 4.

5 Methods for Dealing with Missing Data

Before proceeding with an explanation and illustration of MICE, let's briefly discuss complete case, and single imputation.

Complete case analyses proceed by basically removing all observations with any missing data. This approach will only work if missing data are MCAR. However, in the early stages of an analysis, it can be useful to beginning the process fo writing code by focusing first on a complete case analysis. We've been using complete case exclusively thus far in the course:

```
file_loc <- url("https://cdn1.sph.harvard.edu/wp-content/uploads/sites/1268/1268/20/nhefs.csv")
nhefs <- read_csv(file_loc) %>%
    select(qsmk, wt82_71, sex, age, race, income, marital, school,
        asthma, bronch)  #%>%
# na.omit(.)  this 'na.omit' function removes any
# observations with missing data, thus conducting a
# complete case analysis.
```

```
nhefs
```

```
## # A tibble: 1,629 x 10
##     qsmk wt82_71   sex   age  race income marital school asthma bronch
##    <dbl>   <dbl> <dbl> <dbl> <dbl>  <dbl>   <dbl>  <dbl>  <dbl>  <dbl>
##  1     0 -10.1       0    42     1     19       2      7      0      0
##  2     0   2.60      0    36     0     18       2      9      0      0
##  3     0   9.41      1    56     1     15       3     11      0      0
##  4     0   4.99      0    68     1     15       3      5      0      0
##  5     0   4.99      0    40     0     18       2     11      0      0
##  6     0   4.42      1    43     1     11       4      9      0      0
##  7     0  -4.08      1    56     0     19       2     12      0      0
##  8     0   0.227     1    29     0     22       2     12      0      0
##  9     0  -2.72      0    51     0     18       2     10      0      0
## 10     0   9.86      0    43     0     16       2     11      0      0
## # i 1,619 more rows
```

One thing to consider is that, if a complete case analysis is to be used, you should select the relevant variables needed for an analysis **before** removing observations with missing data. Otherwise, you may end up removing observations that only have missing data in variables that are not relevant to your analysis, thus reducing the sample size unnecessarily.

A second approach that is not as involved as MICE is single imputation. There are generally two versions of this: marginal single imputation and conditional single imputation. The latter proceeds by imputing missing data using a single regression model.

A simpler version of single imputation is marginal imputation. This often proceeds by setting any missing values in a given variable to be the mean, median, or mode of that variable. For example:

```
# number of missing observations in wt82_71
sum(is.na(nhefs$wt82_71))
```

```
## [1] 63
```

```r
# mean imputation of missing data one could just as easily
# use the median, important if variable is skewed
nhefs$wt82_71[is.na(nhefs$wt82_71)] <- mean(nhefs$wt82_71, na.rm = TRUE)

# function for the mode NB: this function will not work if
# NA is the mode!
getmode <- function(v) {
    uniqv <- unique(v)
    uniqv[which.max(tabulate(match(v, uniqv)))]
}

sum(is.na(nhefs$income))
```

```
## [1] 62
```

```r
nhefs$income[is.na(nhefs$income)] <- getmode(nhefs$income)
```

6 Multiple Imputation: Some Initial Considerations

Let's start with a concrete (simulated) example, let's first load relevant packages we'll need to address missing data:

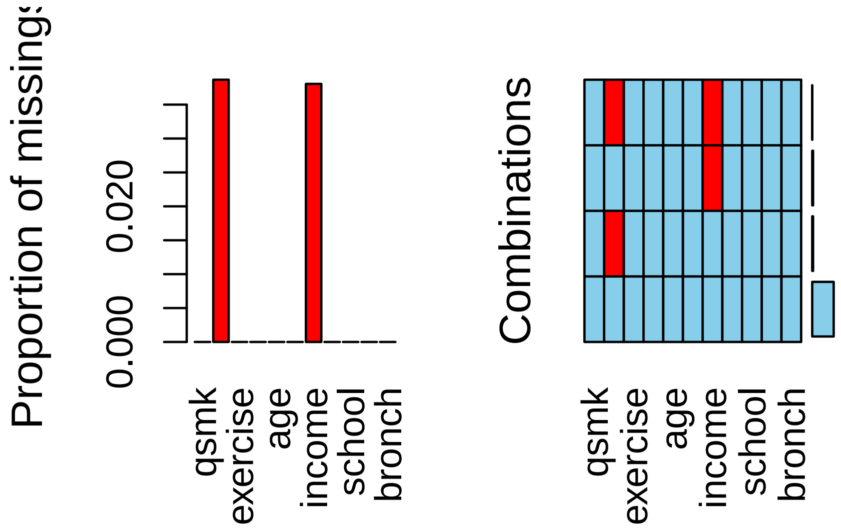
```r
packages <- c("here", "tidyverse", "ggExtra", "VIM", "mice")

for (package in packages) {
    if (!require(package, character.only = T, quietly = T)) {
        install.packages(package, repos = "http://lib.stat.cmu.edu/R/CRAN")
    }
}

for (package in packages) {
    library(package, character.only = T)
}
```

And use data from the NHEFS:

```
file_loc <- url("https://cdn1.sph.harvard.edu/wp-content/uploads/sites/1268/1268/20/nhefs.csv")
nhefs <- read_csv(file_loc) %>%
    select(qsmk, wt82_71, exercise, sex, age, race, income, marital,
        school, asthma, bronch)
```

The first thing we should do is evaluate the extent and distribution of missing data. We can do this in a number of ways, and the VIM package has some helpful tools:

```
VIM::aggr(nhefs)
```



We can also write a function that allows us to tally missing data:

```
miss_func <- function(x) {
    mean_miss <- mean(is.na(x), na.rm = T)
    count_miss <- sum(is.na(x), na.rm = T)

    return(c(mean_miss, count_miss))
}

apply(nhefs, 2, miss_func)
```

Handling missing outcome data can be confusing in certain research settings. For example, applied researchers are sometimes under the assumption that the outcome should never be imputed. This confusion can be traced back to a quote in a paper by Roderick Little (1992):

> If the X's are complete and the missing values of Y are missing at random, then the incomplete cases contribute no information to the regression of Y on $X_1, \ldots, X_p$.

In effect, this quote states that if the missing Y values are MAR, then it will make no difference whether we include or exclude the missing Y's, and thus a complete case analysis would work fine. Based in part Little's article, Stef van Buuren states in a post on Cross Validated that[1]:

[1] However, please read the full post for the important exceptions which van Buuren highlights: https://stats.stackexchange.com/questions/46226/multiple-imputation-for-outcome-variables

> Under MAR, there are generally no benefits to impute the outcome, and for a low number of imputations the results may even be somewhat more variable because of simulation error.

Quotes such as these can and have lead to considerable confusion. This confusion often manifests as "one should never impute the outcome," which is definitely not the case.[2]

[2] If, for instance, there is a variable that is highly predictive of the outcome, has no missing observations, and is not part of the covariates one is adjusting for, one might be better off imputing the outcome, even with a low number of imputations.

To explain why these statements are true (and clarify when they are not), let's introduce some notation. We'll let Y denote the outcome we are studying, X denote our exposure of interest, and we'll let C denote the minimally sufficient adjustment set we obtained from the analysis of our DAG. Recall that the minimally sufficient adjustment set is the set of variables that renders the potential outcomes independent of the observed exposure. We'll come back to this shortly.

Similarly, we'll need a set of variables that render the potential outcome independent of the indicator of missingness for the outcome. So, if we let R denote a variable that is set to 1 if Y is missing and zero otherwise. If there the missing at random assumption will hold if there exists a set of variables Z such that

$$E(Y^x \mid X, Z, R = 0) = E(Y^x \mid X, Z)$$

This above equation reads that the missingness indicator is independent of the outcome conditional on some set of variables Z.

We are now able to explain the context in which Little's statement applies, and where it doesn't. When we're interested in estimating causal effects, we do our best to adjust for the variables in C. Let's assume that these are the same type of variables that Little was referring to as $X_1, \dots, X_p$.[3]

[3] Thus, henceforth, we assume $C = X_1 \dots X_p$.

The first point that should be noted is that *the above statement assumes* $C \equiv Z$. That is, the variables in Z that render the missingness indicator independent of the potential outcomes are the same variables in C that render the exposure independent of the potential outcomes. The above statement also assumes that the estimation approach is, in fact, a conditional regression model. Importantly, Little's statement would not apply if one adjusted for the C variables (i.e., $X_1, \dots, X_p$) using IP-weighting.

7 g Computation and GLM

For an outcome with missing data that are MAR, if we obtain a conditional effect estimate by conditioning for the variables upon which the missing data mechanism depends using a regression model (e.g., GLM), no additional adjustment is required. In fact, this is the exact situation in which Little's statement directly applies. If we obtain a marginal estimate by taking predictions from this model, and average those predictions (i.e., g computation), the same reasoning still applies.

However, there are several exceptions to the statement above. For example, what if we need a separate set of variables Z to account for missing data? That is, we have a set of variables to adjust for confounding C, and a different set of variables that predict missingness Z. In this case, if the variables in Z do not create problems if we adjust for them in our regression model (i.e., mediators, colliders, variables that lead to positivity violations), then we can simply adjust for them.

If the variables in Z do create these problems, then we need to model the missing data mechanisms separately from the confounding mechanisms, which can be done via multiple imputation, discussed below.

8 IP-Weighting

However, if we obtain marginal effect estimates by IP-weighting, Little's statement no longer applies. In fact, to account for missing outcome data when we are not *conditioning* on the set of variables we need an additional step to adjust for the potential selection bias due to missing outcome data. This can be done via inverse probability of censoring weights. Again, assuming $C \equiv Z$, these weights are defined as:

$$w_i = \begin{cases} \frac{P(M=0)}{P(M=0|X,C)}, & \text{if } M_i = 0 \\ 0, & \text{if } M_i = 1 \end{cases}$$

These weights can be obtained in the same way we obtain exposure weights, via logistic regression. One can then multiply the exposure weights by these censoring weights to obtain weights that adjust for both confounding and selection bias (due to missing outcome data).

Because the missing at random assumption corresponds to the exchangeability assumption that we need to identify causal effect, it is possible to demonstrate the problems that result from missing data using directed acyclic graphs (Daniel et al., 2011).

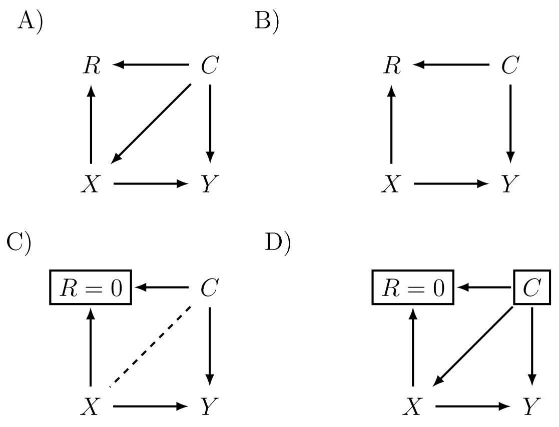


Figure 3: Causal diagrams demonstrating the impact of restricting on observed data when the missingness mechanism depends on the exposure and covariates.

Figure 1 displays four diagrams demonstrating the impact of conducting a complete case analysis under missing outcome data when the missingness mechanism depends on the exposure X and the covariates C. Figure 1A shows a simple DAG with a confounding structure, but where the missingness mechanism is fully explained by the exposure and the outcome. Figure 1B demonstrates the modified DAG when inverse probability weights are applied. However, this figure is incomplete: because we do not observe $R = 1$ any analysis of the data generated from this mechanism would lead to Figure 1C, in which we are forced to restrict to the stratum $R = 0$. In doing so, we again create an association between X and C, thus re-opening a backdoor path that was removed via IP-weighting. Figure 1D shows how this problem does not arise when we use an outcome regression model to adjust for C. That is, even if we restrict to $R = 0$, all non-causal information from X to Y is blocked by conditioning on C.

9 Multiple Imputation: Implementation

It is entirely plausible that in some settings, one would need to model the missing data mechanisms separately from the confounding mechanisms. In addition, in most scenarios, data will be missing not only for the outcome, but for several of the variables needed to conduct the analysis (non monotone). In these cases, multiple imputation is a commonly used tool to address missing data.

Let's use mice to impute the missing NHEFS data. MICE stands for multiple imputation via chained equations. It is sometimes referred to as "multivariate imputation via chained equation", "fully conditional specification" or "sequential regression multiple imputation". This differs from the multiple imputation based on multivariate normal imputation (as implemented in SAS proc mi; Stata does both). The "chained equations" procedure is meant to connote that each variable with missing data is imputed from a model that takes the variable form into account (e.g., binomial, polytomous, ordinal, continuous), instead of just assuming everything is MVN. Furthermore, these regression equations are "chained" together into an algorithm that allows us to predict values of non-monotone (or monotone) missing data iteratively.

To proceed with `mice` in R, we first install and load the package:

```r
install.packages("mice", repos = "http://lib.stat.cmu.edu/R/CRAN/")
```

```
##
## The downloaded binary packages are in
##  /var/folders/zm/rqfqp5xs0fs86qs2mcxk6q0r0000gr/T//RtmplB2iDt/downloaded_packages
```

```
library(mice)
```

The first step in running an analysis via mice is to look at our data and ensure that all variables are properly coded. Specifically, to work properly within the `mice` algorithm, categorical variables should be coded as factors (even if initially coded as numeric dummy variables). In our dataset, we have to recode the following variables:

```
factor_names <- c("exercise", "income", "marital", "sex", "race",
    "asthma", "bronch")
nhefs[, factor_names] <- lapply(nhefs[, factor_names], factor)

nhefs <- nhefs %>%
    mutate(u1 = runif(nrow(nhefs)), school = if_else(u1 < 0.15,
        NA_real_, school)) %>%
    select(-u1)

nhefs
```

```
## # A tibble: 1,629 x 11
##     qsmk wt82_71 exercise sex     age race  income marital school asthma bronch
##    <dbl>   <dbl> <fct>    <fct> <dbl> <fct> <fct>  <fct>    <dbl> <fct>  <fct>
##  1     0 -10.1   2        0        42 1     19     2            7 0      0
##  2     0   2.60  0        0        36 0     18     2            9 0      0
##  3     0   9.41  2        1        56 1     15     3           11 0      0
##  4     0   4.99  2        0        68 1     15     3            5 0      0
##  5     0   4.99  1        0        40 0     18     2           11 0      0
##  6     0   4.42  1        1        43 1     11     4            9 0      0
##  7     0  -4.08  1        1        56 0     19     2           NA 0      0
##  8     0   0.227 2        1        29 0     22     2           12 0      0
##  9     0  -2.72  2        0        51 0     18     2           10 0      0
## 10     0   9.86  2        0        43 0     16     2           11 0      0
## # i 1,619 more rows
```

Next, we implement an **initialization run** with the mice package. This basically requires that we run the mice function, but with no iterations. For a particularly large dataset, we can also set the number of imputations to 1 via the `m=1` command in the mice function.

The initialization run enables us to populate many of the mice objects with values, so we can explore and modify accordingly:

```
ini <- mice(nhefs, seed = 123, maxit = 0)
```

This initialization run enables us to obtain a number of basic tools we will need to properly run mice. The first step is to check that the correct variables are being imputed, and that the method of imputation is appropriate.

```
ini$method
```

```
##      qsmk   wt82_71  exercise       sex       age      race    income   marital
##        ""     "pmm"        ""        ""        ""        "" "polyreg"        ""
##    school    asthma    bronch
##     "pmm"        ""        ""
```

This output tells us that `wt82_71` and `income` are being imputed. These correspond to the variables that, in fact, contain missing observations according to the Figure we obtained from the `aggr` function above. Furthermore, the output tells us that both predictive mean matching `pmm` and polytomous logistic regression `polyreg` are being used to impute missing data. A description of the available methods can be found in Table 1 of van Buuren and Groothuis-Oudshoorn (2011).

It is important to note that choosing methods for continuous variables when the variable being imputed is integer can sometimes lead to problems. For example, in the `nhefs` data, `school` might be considered an ordinal variable (i.e., ordered factor), but we we are treating it as an integer (numeric). Because it is numeric, using predictive mean matching may lead to noninteger imputations (for example, we may end up with a number of years of school attained of 10.125). If this is likely to cause problems in the analysis, one option is to treat `school` as a factor variable. However, because there are so many levels to this variable (18, in fact), this may result in a considerably slow imputation algorithm.

Another option is to impute using a fast continuous method, such as `pmm`, and then in the post processing phase, convert all non-integer values to integers (using, e.g., the `round()`, `ceil()` or `floor()` functions in R). Before doing this however, one could evaluate the imputed values for school to see if they are problematic:

```
apply(ini$imp$school, 2, table)
```

```
## $`1`
##
##   0   1   2   5   6   7   8   9  10  11  12  13  15  16  17
##   2   1   1   3   6   9  18  14  24  24 101  10   6  21  13
##
## $`2`
##
##   0   2   3   5   6   7   8   9  10  11  12  13  15  16  17
##   1   1   2   4   2   6  21  12  41  14 100  14   4  17  14
##
## $`3`
##
##   0   2   3   4   5   6   7   8   9  10  11  12  13  15  16  17
##   5   2   2   1   2   6   8  23  11  29  21 104  13   1  16   9
##
## $`4`
##
##   0   3   4   5   6   7   8   9  10  11  12  13  15  16  17
##   1   2   1   1   6   8  27  14  25  13 110  10   4  20  11
##
## $`5`
##
##  0  1  2  3  4  5  6  7  8  9 10 11 12 13 15 16 17
##  1  2  3  2  2  2  4  8 28 12 22 23 94 15  6 16 13
```

The output from this apply function tells us that all of the imputed school data in each of the five imputations are integers, so there is no need to worry about noninteger values here.

Note that if we wanted to change the imputation method, we would simply have to replace the text in the `ini$method` object with the appropriate selection from Table 1 of van Buuren and Groothuis-Oudshoorn (2011), and then ensure that this set of methods is the selected option the next time we run `mice()`. For the school variable, for example, this could be done with the following code:

```
nhefs$school <- factor(nhefs$school)
ini$method["school"] <- "polr"
mice_run <- mice(nhefs, seed = 123, maxit = 0, method = ini$method)
```

The next element we need to examine is the predictor matrix. This is an essential object that should be modified to properly impute the missing data. It is advisable to save this predictor matrix as an excel or csv file, so as to easily manipulate it.

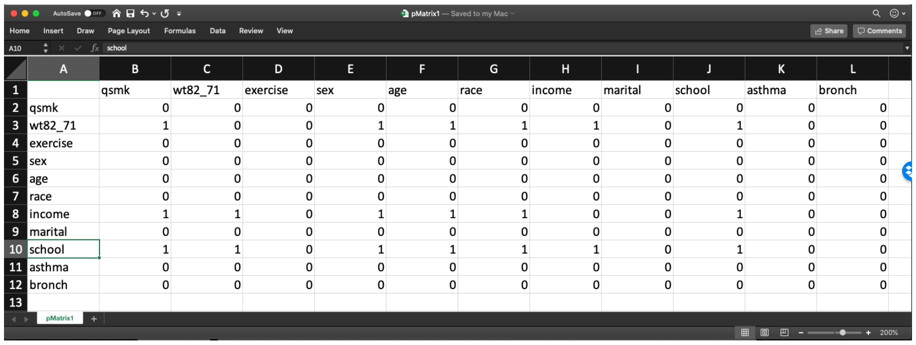
```
options(width = 90)
pMatrix <- ini$predictorMatrix
write.csv(pMatrix, here("Section4", "pMatrix.csv"))
pMatrix
```

```
##            qsmk wt82_71 exercise sex age race income marital school asthma bronch
## qsmk          0       1        1   1   1    1      1       1      1      1      1
## wt82_71       1       0        1   1   1    1      1       1      1      1      1
## exercise      1       1        0   1   1    1      1       1      1      1      1
## sex           1       1        1   0   1    1      1       1      1      1      1
## age           1       1        1   1   0    1      1       1      1      1      1
## race          1       1        1   1   1    0      1       1      1      1      1
## income        1       1        1   1   1    1      0       1      1      1      1
## marital       1       1        1   1   1    1      1       0      1      1      1
## school        1       1        1   1   1    1      1       1      0      1      1
## asthma        1       1        1   1   1    1      1       1      1      0      1
## bronch        1       1        1   1   1    1      1       1      1      1      0
```

The pMatrix is what determines which covariates are used to impute missing data. In this matrix, a value of 1 means that the column variable is being used as a predictor to impute the row variable. The default pMatrix is a square matrix populated entirely with 1's, except for the diagonal. So, in effect, all variables are used to impute any missingness (van Buuren and Groothuis-Oudshoorn, 2011). This may lead to problems, particularly if the dataset includes variables that should not be used to impute (such as ID variables, or similar administrative variables).

Thus, one must always modify the pMatrix to match the missing data mechanisms that arise in the study. In effect, one must appropriately determine the form of the imputation models. This leads to an important consideration of any approach to imputing missing data: the imputation model should always be more flexible than the analysis model.

The following figure shows how we've modified our pMatrix:



	qsmk	wt82_71	exercise	sex	age	race	income	marital	school	asthma	bronch
qsmk	0	0	0	0	0	0	0	0	0	0	0
wt82_71	1	0	0	1	1	1	1	0	1	0	0
exercise	0	0	0	0	0	0	0	0	0	0	0
sex	0	0	0	0	0	0	0	0	0	0	0
age	0	0	0	0	0	0	0	0	0	0	0
race	0	0	0	0	0	0	0	0	0	0	0
income	1	1	0	1	1	1	0	0	1	0	0
marital	0	0	0	0	0	0	0	0	0	0	0
school	1	1	0	1	1	1	1	0	1	0	0
asthma	0	0	0	0	0	0	0	0	0	0	0
bronch	0	0	0	0	0	0	0	0	0	0	0

To use this pMatrix, we must import it and create an appropriate matrix object with row and column names:

```r
pMat <- read_csv(here("Section4", "pMatrix1.csv"))[, -1]
pMat <- as.matrix(pMat)
row.names(pMat) <- colnames(pMat)
pMat
```

```
##          qsmk wt82_71 exercise sex age race income marital school asthma bronch
## qsmk        0       0        0   0   0    0      0       0      0      0      0
## wt82_71     1       0        0   1   1    1      1       0      1      0      0
## exercise    0       0        0   0   0    0      0       0      0      0      0
## sex         0       0        0   0   0    0      0       0      0      0      0
## age         0       0        0   0   0    0      0       0      0      0      0
## race        0       0        0   0   0    0      0       0      0      0      0
## income      1       1        0   1   1    1      0       0      1      0      0
## marital     0       0        0   0   0    0      0       0      0      0      0
## school       1          1          0   1   1    1        1          0         1         0         0
## asthma       0          0          0   0   0    0        0          0         0         0         0
## bronch       0          0          0   0   0    0        0          0         0         0         0
```

With the relevant methods selected and the pMatrix defined appropriately, we are ready to run the imputation. This can be done by calling the `mice` function with the appropriate arguments.

Two of the arguments needed are the number of imputations and the number of iterations. In the early literature on multiple imputation, much of the literature cited the need for 5 imputations, which was shown to be sufficient from an efficiency perspective (Rubin, 1987). However, with today's computing power, increasing this number can only help us with simulation error (though too many imputations may hurt us in terms of computing time). As a general rule of thumb, one should use an imputation number equal to the largest percent missing (White et al., 2011). So, if the data has a variable with up to 20% missing, one should use at least 20 imputations.

The next option to choose is the number of iterations. `mice` is an iterative procedure, in that imputed values of one variable can be used to impute the values of another variable, which in turn will be used to impute missing values in the original variable. This iterative procedure requires a stopping criterion, and enough iterations to ensure that the procedure results in a stable set of values. Generally, the recommended number of iterations is often between 20-30, as things tend not to improve once we go above this these numbers.

10 MICE: IPW and GLM

Imputing missing data via MICE for estimators that do not require bootstrapping is typically how the procedure is described. The process occurs by (i) imputing missing data, (ii) estimating the effect of interest in each imputed dataset, and (iii) summarizing the estimates and their standard errors into a single measure.

For step (i), we can simply run the `mice` function with appropriately selected options:

```
mice_imp <- mice(nhefs, seed = 123, m = 10, maxit = 20, printFlag = F,
    method = ini$method, pMatrix = pMat)
```

This gives us an object that contains many things, including the 10 imputed datasets. If our objective is to fit either a linear or generalized linear model (with the `lm` or `glm` functions), then we can use this `mice_imp` object directly and obtain valid point estimates and standard errors. We do this using the `with`, `summary`, and `pool` functions:

```
mod1 <- with(mice_imp, lm(wt82_71 ~ qsmk + sex + age))
summary(pool(mod1))
```

```
##          term   estimate  std.error   statistic        df      p.value
## 1 (Intercept)  9.1850819 0.74354240  12.3531379 1513.8761 1.808006e-33
## 2        qsmk  3.0819619 0.44673108   6.8989198  779.7012 1.084526e-11
## 3        sex1 -0.3745884 0.37908177  -0.9881467 1492.0424 3.232410e-01
## 4         age -0.1639788 0.01583679 -10.3542906 1234.6012 3.767830e-24
```

However, if our goal is to use a more tailored estimator, such as IP-weighting, we have to first extract the imputed datasets, fit the IPW estimator to each, and then combine the results from each imputed dataset together into one. We can extract the imputed data using the `complete` function. Note that the option `action="long"` returns data with each imputed dataset stacked together. Note also that there is a tidyverse function named `complete` that conflicts with this one. The easiest strategy is to use the `mice::` call before the function call, such as:

```
imp_data <- mice::complete(mice_imp, action = "long")
```

This way, we force `R` to use the `complete` function in `mice`, and avoid any conflict with other packages. With this new `imp_data` object, we can now implement IP-weighting. The approach requires that we estimate the propensity score, create stabilized weights, and estimate the association of interest separately for each imputed dataset. We can then combine all estimates and their standard errors using an equation that accounts for the within and between imputation variance of the estimator.[4] Let's first apply the IPW estimator to each imputed dataset, which we can do with a for loop. We'll store the estimate and standard error for each imputation:

[4] These equations are sometimes referred to as Rubin's Rules.

```
est.psi <- est.se <- NULL
for (ii in 1:10) {
    ps <- glm(qsmk ~ sex + age, data = subset(imp_data, .imp ==
        ii))$fitted.values
    qsmk <- imp_data[imp_data$.imp == ii, ]$qsmk
    num <- mean(qsmk) * qsmk + (1 - mean(qsmk)) * (1 - qsmk)
    den <- ps * qsmk + (1 - ps) * (1 - qsmk)
    sw <- num/den
    mod <- lm(wt82_71 ~ qsmk, weights = sw, data = subset(imp_data,
        .imp == ii))
    est.psi <- rbind(est.psi, coef(mod)[2])
    est.se <- rbind(est.se, sqrt(sandwich::sandwich(mod)[2, 2]))
}
```

The above code gives us two vectors, one with an estimate for each imputation (`est.psi`), and one with the estimate's standard error (`est.se`):

```
cbind(est.psi, est.se)
```

```
##           qsmk
##  [1,] 3.280303 0.4751226
##  [2,] 2.909562 0.4708745
##  [3,] 3.050859 0.4696739
##  [4,] 3.082768 0.4790880
##  [5,] 2.886119 0.4752341
##  [6,] 3.013372 0.4720176
##  [7,] 3.050844 0.4749498
##  [8,] 3.100899 0.4738779
##  [9,] 3.093514 0.4724099
## [10,] 2.996139 0.4748453
```

To combine the estimates from each imputation into a single estimate, we can simply take their average. However, combining the standard errors across all estimates requires that we average the standard errors across all imputations, and add the additional variation from the simulations. To do both, we can use the following function:

```r
# PROGRAM FOR RUBIN'S RULES
RubinsRules <- function(estimate, se) {
    q <- mean(estimate)
    u <- mean(se)
    b <- var(estimate)
    m <- nrow(as.matrix(estimate))
    t <- u + (1 + (1/m)) * b
    se <- sqrt(t)
    df <- (m - 1) * (1 + ((m * u)/((m + 1) * b)))^2
    ucl <- q + qt(0.975, df) * se
    lcl <- q - qt(0.975, df) * se
    results <- c(q, se, lcl, ucl)
    names(results) <- c("psi", "se", "lcl", "ucl")
    return(results)
}

# THIS FUNCTION TAKES TWO ARGUMENTS: THE POINT ESTIMATES
# AND THE STANDARD ERRORS FOR THESE POINT ESTIMATES
imp_res <- RubinsRules(est.psi, est.se)
imp_res
```

```
##       psi        se       lcl       ucl
## 3.0464379 0.6979739 1.6782958 4.4145800
```

These results can be interpreted as our estimates of interest adjusted for missing data.

11 MICE: g Computation

When the bootstrap must be used to estimate standard errors, the approach to multiple imputation differs slightly. In effect, the only difference is that one sets the number of imputations to `m=1`, but incorporates the imputation procedure into the bootstrap loop. In our case, this can be accomplished as follows:

```r
est.psi <- NULL
set.seed(123)
for(i in 1:20){ ## set the max number to 200!!
  index <- sample(1:nrow(nhefs),nrow(nhefs),replace=T)
  boot_dat <- nhefs[index,]
  mice_imp <- mice(boot_dat,
                   #seed=123,
                   m=1,
                   maxit=20,
                   printFlag = F,
                   method=ini$method,
                   pMatrix=pMat)
  imp_data <- complete(mice_imp,action="long")
  modY <- lm(wt82_71 ~ qsmk+age+sex,data=imp_data)
  pY1 <- mean(predict(modY,newdata=transform(imp_data,qsmk=1)))
  pY0 <- mean(predict(modY,newdata=transform(imp_data,qsmk=0)))
  est.psi <- rbind(est.psi,pY1-pY0)
}
```

This for loop generates the following object:

```r
head(est.psi)
```

```
##          [,1]
## [1,] 2.214095
## [2,] 3.991112
## [3,] 3.173535
## [4,] 2.814814
## [5,] 3.194648
## [6,] 3.253501
```

To get point estimates and standard errors, we simply need to take the mean of `est.psi` and the standard deviation of `est.psi`, which gives 3.11 and 0.503, respectively.

References

Rhian M. Daniel, Michael G Kenward, Simon N Cousens, and Bianca L De Stavola. Using causal diagrams to guide analysis in missing data problems. *Stat Methods in Med Res*, 2011.

Roderick J. A. Little. Regression with missing x's: A review. *Journal of the American Statistical Association*, 87(420):1227–1237, 1992.

Donald B. Rubin. *Multiple imputation for nonresponse in surveys*. Wiley series in probability and mathematical statistics. Applied probability and statistics, 0271-6232. New York ; Wiley, c1987., 1987.

Stef van Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software*, 45(3), 2011.

Ian R. White, Patrick Royston, and Angela M. Wood. Multiple imputation using chained equations: Issues and guidance for practice. *Statistics in Medicine*, 30(4):377–399, 2011.