Incomplete Data Analysis - Assignment 3

Yile Shi (s2168022)

2022/3/31

Question 1

(a) - Solution

Function `cc()` in `mice` package returns the complete cases in a dataset. Thus, we count the number of complete cases in dataset 'nhanes' and further compute the percentage of incomplete cases by $1 - \frac{\#complete\ cases}{\#all\ cases}$.

As a result, we obtain the percentage of incomplete cases is 48% (12 cases out of 25).

```
# the percentage of incomplete cases
1 - nrow(cc(nhanes)) / nrow(nhanes)
```

```
## [1] 0.48
```

(b) - Solution

Recall that to estimate the variance of $\hat{\theta}^{MI}$, the multiple imputation of θ, we calculate the following statistics:

1. **between-imputation variance:** $B = \frac{1}{M-1} \sum_{i=1}^{M} \left(\hat{\theta}^{(i)} - \hat{\theta}^{MI} \right)^2$
2. **within-imputation variance:** $\overline{U} = \frac{1}{M} \sum_{i=1}^{M} \hat{U}^{(i)}$, where $\hat{U}^{(i)}$ is the estimated variance of $\hat{\theta}^{(i)}$

Then we obtain the **total variance** $V^{MI} = \overline{U} + \left(1 + \frac{1}{M}\right) B$. Note that M denotes the number of imputed datasets by `mice` ($M = 5$ by default).

According to the work by Buuren et al. 2011, the proportion of total variance that is attributed to the missing values is

$$\lambda = \frac{B + \frac{B}{M}}{V^{MI}}$$

We yield the following results with the standard MICE procedure - `mice()`, `with()` and `pool()`. Note that in step 2 (`with()`), we fit the normal linear regression of `bmi` over `age`, `hyp` and `cyl`.

As mentioned before, we look at the column `lambda` to get the proportion of variance due to missing data for each parameter (3 decimal places, seed = 1):

1. $\lambda_{age} = 0.686$
2. $\lambda_{hyp} = 0.350$
3. $\lambda_{chl} = 0.304$

Based on the proportion of variance due to missing data, we conclude that `age` appears to be the most affected by the non-response with the largest value of λ.

```
# impute dataset with mice
imp.list <- mice(nhanes, printFlag = FALSE, seed = 1)

# linear model to predict bmi
bmi.fit <- with(imp.list, lm(bmi ~ age + hyp + chl))

# pool the results
bmi.pool <- pool(bmi.fit)
bmi.pool$pooled
```

```
##          term m    estimate         ubar            b            t dfcom
## 1 (Intercept) 5 19.61789252 10.588884721 0.8662133972 11.628340797    21
## 2         age 5 -3.55287155  0.744810536 1.3585594461  2.375081872    21
## 3         hyp 5  2.19701748  2.886391704 1.2976511612  4.443573098    21
## 4         chl 5  0.05378081  0.000287288 0.0001046083  0.000412818    21
##          df        riv     lambda       fmi
## 1 16.936189 0.09816483 0.08938989 0.1807424
## 2  3.528053 2.18884032 0.68640637 0.7824821
## 3  9.035494 0.53949067 0.35043452 0.4583762
## 4 10.228828 0.43694808 0.30408063 0.4092932
```

(c) - Solution

We repeat same analysis on λ by using different random seeds from 2 to 6. The result are shown in the data frame below.

We can observe that the conclusions in 1.(b) do not remain the same for different seeds. The values of λ and the parameter most affected by the non-response vary dramatically.

For seed 2, 3 and 6, `age` has the largest value of λ (0.403, 0.590, 0.655 of each, 3 decimal places) and is considered to be most affected by the non-response, which is consistent with conclusions in 1.(b), using random seed 1. On the other hand, `chl` and `hyp` take the largest proportion of variance for seed 4 and 5 respectively, with values 0.331 and 0.594 of each.

Recall that the larger amount of missing data, the larger the variability of values of λ will be. Since nearly half (48%) of cases in the dataset are incomplete, it is reasonable to have different conclusions every time we adjust the random seed.

```
# create a data frame to store the results from different seeds
bmi.lambda <- data.frame(age = bmi.pool$pooled[2, 10],
                         hyp = bmi.pool$pooled[3, 10],
                         chl = bmi.pool$pooled[4, 10],
                         row.names = "seed.1")

# repeat analysis with different seeds
for (i in seq(2, 6)){
  imp.list <- mice(nhanes, printFlag = FALSE, seed = i)
  bmi.fit <- with(imp.list, lm(bmi ~ age + hyp + chl))
  bmi.pool <- pool(bmi.fit)
  df <- data.frame(age = bmi.pool$pooled[2, 10],
                   hyp = bmi.pool$pooled[3, 10],
                   chl = bmi.pool$pooled[4, 10],
                   row.names = paste("seed.", i, sep = ""))
  bmi.lambda <- rbind(bmi.lambda, df)
}
bmi.lambda
```

```
##              age       hyp       chl
## seed.1 0.6864064 0.3504345 0.3040806
## seed.2 0.4033924 0.1430995 0.2959966
## seed.3 0.5895051 0.4101152 0.5621346
## seed.4 0.2189333 0.1961083 0.3305334
## seed.5 0.4511896 0.5942866 0.2346065
## seed.6 0.6549523 0.2960364 0.5196295
```

(d) - Solution

From 1.(c), we observe that the parameter most affected by the non-response varies as random seed changes. Thus, we expect that increasing the number of imputed datasets can lead to more consistent and stable results which has low dependence on the choice of random seed.

We keep using same random seeds from 1 to 6, but change the number of imputed datasets, M, from 5 to 100. Again, we obtain a data frame with values of λ for each parameter with different random seed.

Now we can find that `age` always takes the largest proportion of variance due to missing cases (largest value of λ) in 5 of 6 seeds. For seed 3, the variable `chl` becomes the parameter which is affected by the non-response, with $\lambda = 0.328$. It appears that the conclusions become more consistent with $M = 100$, as we expected.

As far as I am concerned, I prefer the analyses with $M = 100$.

Recall that the **total variance** of θ (`bmi` in this question) is calculated by

$$V^{MI} = \overline{U} + \left(1 + \frac{1}{M}\right) B$$

where $B = \frac{1}{M-1} \sum_{i=1}^{M} \left(\hat{\theta}^{(i)} - \hat{\theta}^{MI}\right)^2$ and $\overline{U} = \frac{1}{M} \sum_{i=1}^{M} \hat{U}^{(i)}$.

Therefore, large value of M can reduce the values of both $\overline{U}$ and B, and further the total variance, increasing the reliability of the estimates. In terms of high accuracy of predictions, larger number of imputed datasets is a better choice.

However, time consumption and low statistical efficiency can be the problems of large value of M. When imputing large scaled dataset with a large number of missing values, large M will relatively increase the running time. Moreover, the imputation procedure arrives at stable and statistically significant conclusions when M reaches a certain value. Increasing M beyond this value does not significantly influence the conclusions but consumes resource and time.

Overall, I still consider $M = 100$ as a better choice as the dataset in this question only have 25 observations and 100 imputed datasets can be easily handled.

```
# initialize the data frame
bmi.lambda.100 <- NULL

# repeat analysis with m = 100
for (i in seq(1, 6)){
  imp.list <- mice(nhanes, m = 100, printFlag = FALSE, seed = i)
  bmi.fit <- with(imp.list, lm(bmi ~ age + hyp + chl))
  bmi.pool <- pool(bmi.fit)
  df <- data.frame(age = bmi.pool$pooled[2, 10],
                   hyp = bmi.pool$pooled[3, 10],
                   chl = bmi.pool$pooled[4, 10],
                   row.names = paste("seed.", i, sep = ""))
  bmi.lambda.100 <- rbind(bmi.lambda.100, df)
}
bmi.lambda.100
```

```
##              age       hyp       chl
## seed.1 0.4324680 0.2915346 0.3217837
## seed.2 0.4031077 0.2825108 0.2939693
## seed.3 0.3093072 0.2425105 0.3281911
```

```
## seed.4 0.3943223 0.2565132 0.2835232
## seed.5 0.3322570 0.2893046 0.2461956
## seed.6 0.4430300 0.2860700 0.3113085
```

Question 2

Solution

As required, we apply stochastic regression imputation (SRI) and bootstrap sampling to generate the data in step 2. To calculate the empirical coverage probability, according to **NOTE 1**, we define a counter to count the times when the ground truth value of β_1, i.e. 3, is covered in its 95% empirical confidence interval for each method and further obtain the probability with the corresponding frequency.

From the results below, we observe the empirical coverage probability of bootstrap imputation is larger than the probability of SRI (0.95 vs 0.88). The reason for this is that stochastic regression imputation does not incorporate the variability of function weights, which means the uncertainty of imputed values is not considered, unlike bootstrap imputation. Thus, generally the 95% confidence intervals obtained using SRI are more narrow and less likely to contain the ground truth value.

```
# read dataset into R
load("dataex2.Rdata")

# initialize counters for empirical coverage probabilities of two methods
count.sri <- count.boot <- 0

# MICE standard procedure for each dataset
for (i in 1:100){

  # step 1 - mice()
  imp.sri <- mice(dataex2[, , i], m = 20, seed = 1,
                  printFlag = FALSE, method = "norm.nob")
  imp.boot <- mice(dataex2[, , i], m = 20, seed = 1,
                   printFlag = FALSE, method = "norm.boot")

  # step 2 - with()
  fit.sri <- with(imp.sri, lm(Y ~ X))
  fit.boot <- with(imp.boot, lm(Y ~ X))

  # step 3 - pool()
  pool.sri <- summary(pool(fit.sri), conf.int = TRUE)
  pool.boot <- summary(pool(fit.boot), conf.int = TRUE)

  # if the ground truth, 3, is covered in the 95% CI of beta_1 ...
  if (pool.sri$`2.5 %`[2] <= 3 & 3 <= pool.sri$`97.5 %`[2]){
    count.sri = count.sri + 1
  }
  if (pool.boot$`2.5 %`[2] <= 3 & 3 <= pool.boot$`97.5 %`[2]){
    count.boot = count.boot + 1
  }
}

# report the empirical coverage probability
data.frame(sri = count.sri / 100, bootstrap = count.boot / 100,
           row.names = "prob.")
```

```
##        sri bootstrap
## prob. 0.88      0.95
```

Question 3

Solution

We assume a general linear regression model $y = \boldsymbol{x}\boldsymbol{\beta} + \epsilon$ for the multiple imputation procedure. Here,

1. $y = (y_1, y_2, \cdots, y_m)$: response variable
2. $\boldsymbol{x} = (\mathbf{1}, \boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_n})$: covariates (including intercept term)
3. $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2, \cdots, \beta_n)$: coefficients
4. $\epsilon \sim \mathcal{N}(0, \sigma^2)$: noise

We start with strategy (i).

Assume that step 1 of the multiple imputation is performed and the number of imputed datasets is M. We fit the regression for each imputed dataset based on the pre-defined linear model and obtain the predicted values (point estimates) $\hat{\boldsymbol{y}} = \left(\hat{y}^{(1)}, \hat{y}^{(2)}, \cdots, \hat{y}^{(M)}\right)$. We pool the predicted values according to Rubin's rule for point estimates:

$$
\begin{aligned}
\tilde{y} &= \frac{1}{M}\sum_{i=1}^{M} \hat{y}^{(i)} \\
&= \frac{1}{M}\sum_{i=1}^{M} \left(\beta_0^{(i)} + \sum_{j=1}^{n} \beta_j^{(i)} x_j\right) \\
&= \tilde{\beta}_0 + \frac{1}{M}\sum_{i=1}^{M}\sum_{j=1}^{n} \beta_j^{(i)} x_j \\
&= \tilde{\beta}_0 + \sum_{j=1}^{n} \left(x_j \cdot \frac{1}{M}\sum_{i=1}^{M} \beta_j^{(i)}\right) \\
&= \tilde{\beta}_0 + \sum_{j=1}^{n} \tilde{\beta}_j x_j
\end{aligned}
$$

where $\tilde{\beta}_0 = \frac{1}{M}\sum_{i=1}^{M} \beta^{(i)}$ and $\tilde{\beta}_j = \frac{1}{M}\sum_{i=1}^{M} \beta_j^{(i)}$ are the pooled regression coefficients with intercept.

Now we work on strategy (ii) with same notations used in (i).

Before predicting, we pool the regression coefficients from each model, i.e. $\boldsymbol{\beta^{(i)}} = \left(\beta_0^{(i)}, \beta_1^{(i)}, \cdots, \beta_n^{(i)}\right),\ i = 1, 2, \cdots, M$ in step 2 using Rubin's rule for point estimates. In this way, we also obtain $\tilde{\beta}_0 = \frac{1}{M}\sum_{i=1}^{M} \beta^{(i)}$ and $\tilde{\beta}_j = \frac{1}{M}\sum_{i=1}^{M} \beta_j^{(i)}$ and further the same predicting expression $\tilde{y} = \tilde{\beta}_0 + \sum_{j=1}^{n} \tilde{\beta}_j x_j$ as we derived for strategy (i).

Therefore, the equations above prove that two strategies are mathematically equivalent and lead to same results and conclusions.

Question 4

(a) - Solution

We need to prevent x_2 to be imputed and only impute y and x_1 variables in step 1. To this end, we pre-define the predictor matrix where the row for x_2 are all 0s so that x_2 will not be imputed. Then we apply MICE on the dataset and consider the interaction term x_1x_2 in `with()` procedure.

According to the `pool()` procedure, we obtain the estimates and 95% confidence intervals of β_1, β_2 and β_3 as follows (3 decimal places):

1. $\hat{\beta}_1 = 1.411 \quad 95\%CI = (1.219, 1.603)$
2. $\hat{\beta}_2 = 1.966 \quad 95\%CI = (1.861, 2.071)$
3. $\hat{\beta}_3 = 0.755 \quad 95\%CI = (0.642, 0.868)$

Comparing the results above with the ground truth values ($\beta_1 = 1, \beta_2 = 2, \beta_3 = 1$), we observe that only the 95% confidence interval corresponding to β_2 covers its true value, with a reasonable estimate.

On the other hand, the estimates for imputed variables, y and x_1, are relatively inaccurate and the corresponding 95% confidence intervals fail to cover their true values.

Based on these. we consider that the *impute and then transform* method leads to biased estimates for imputed variables.

```
# read dataset into R
load("dataex4.Rdata")

# create predictor matrix so that x_2 won't be imputed
pred.mat <- matrix(c(0, 1, 0, 1, 0, 0, 1, 1, 0), ncol = 3)
colnames(pred.mat) <- c("y", "x1", "x2")
rownames(pred.mat) <- c("y", "x1", "x2")

# MI only imputing y and x_1
imp.q4a <- mice(dataex4, m = 50, seed = 1, printFlag = FALSE,
                    predictorMatrix = pred.mat)  # mice() procedure
fit.q4a <- with(imp.q4a, lm(y ~ x1 + x2 + x1*x2)) # with() procedure
pool.q4a <- pool(fit.q4a)                         # pool() procedure

# report regression coefficients and 95% CIs
summary(pool.q4a, conf.int = TRUE)[, c("term", "estimate", "2.5 %", "97.5 %")]
```

```
##          term  estimate    2.5 %    97.5 %
## 1 (Intercept) 1.5929831 1.404501 1.7814655
## 2          x1 1.4112333 1.219397 1.6030697
## 3          x2 1.9658191 1.860657 2.0709812
## 4       x1:x2 0.7550367 0.642302 0.8677715
```

(b) - Solution

Now we consider using *passive imputation* to impute the missing values in the interaction variable x_1x_2.

Since the original dataset does not contain the column for interaction variable, we calculate and add the interaction column into the dataset. Note that if $x_1 = NA$, the corresponding $x_1x_2 = NA$ too.

To apply *passive imputation*, we change the `method` argument for x_1x_2 so that its imputation method is the interaction of x_1 and x_2, i.e. $\sim I(x1 * x2)$, in the `mice()` procedure. Moreover, we make some restrictions to the predictor matrix such that y is not used to impute x_1x_2 and x_1x_2 is not considered when imputing x_1 and x_2.

With everything ready, we implement multiple imputation on the dataset with pre-defined method and predictor matrix and obtain the estimates of each parameter with their 95% confidence intervals.

1. $\hat{\beta}_1 = 1.193 \quad 95\%CI = (1.003, 1.382)$
2. $\hat{\beta}_2 = 1.996 \quad 95\%CI = (1.899, 2.094)$
3. $\hat{\beta}_3 = 0.874 \quad 95\%CI = (0.762, 0.987)$

Comparing with *impute and then transform* method, we obtain a more accurate estimate for β_2, with a narrower 95% confidence interval including the true value, indicating higher accuracy of *passive imputation*.

The estimates for β_1 and β_3 are also closer to the ground truth values, proving the improved estimation accuracy. However, the problem that the corresponding confidence intervals do not cover the true values of two parameters still arises.

Thus, we consider that the bias caused by the model is reduce by *passive imputation* but still remains.

```
# calculate interaction variable and add to dataset
dataex4$x1x2 <- dataex4$x1 * dataex4$x2

# apply passive imputation
# initial mice() to get the method
imp.q4b.0 <- mice(dataex4, m = 50, seed = 1, printFlag = FALSE)
# specify x1x2 is the interaction of x_1 and x_2
imp.q4b.0$method['x1x2'] <- "~I(x1*x2)"

# set corresponding row in predictor matrix to 0
# x_1 and x_2 should not be imputed by x1x2
imp.q4b.0$predictorMatrix[c("x1", "x2"), "x1x2"] <- 0
# x1x2 should not be imputed by y
imp.q4b.0$predictorMatrix["x1x2", "y"] <- 0

# MI
imp.q4b <- mice(dataex4, m = 50, seed = 1, printFlag = FALSE,
                predictorMatrix = imp.q4b.0$predictorMatrix,
                method = imp.q4b.0$method)         # mice() procedure
fit.q4b <- with(imp.q4b, lm(y ~ x1 + x2 + x1x2))   # with() procedure
pool.q4b <- pool(fit.q4b)                          # pool() procedure

# report estimates and 95% CIs
summary(pool.q4b, conf.int = TRUE)[, c("term", "estimate", "2.5 %", "97.5 %")]
```

```
##          term  estimate     2.5 %    97.5 %
```

```
## 1 (Intercept) 1.5534782 1.3788626 1.7280939
## 2          x1 1.1926170 1.0034980 1.3817360
## 3          x2 1.9964402 1.8989468 2.0939336
## 4        x1x2 0.8740573 0.7615712 0.9865434
```

(c) - Solution

In this case, we consider the interaction variable x_1x_2 as *just another variable*, meaning that we impute the missing values in x_1x_2 without depending on the missing values in x_1 variable. Thus, we directly construct the linear regression over x_1, x_2 and x_1x_2 in `with()` procedure, without any further steps.

As a result, we obtain the following estimates and 95% confidence intervals of β_1, β_2 and β_3 (3 decimal places):

1. $\hat{\beta}_1 = 1.000 \quad 95\%CI = (0.841, 1.166)$
2. $\hat{\beta}_2 = 2.026 \quad 95\%CI = (1.940, 2.113)$
3. $\hat{\beta}_3 = 1.018 \quad 95\%CI = (0.930, 1.105)$

We observe that the method treating the interaction x_1x_2 as *just another variable* derives more accurate estimates for each parameter. Moreover, this method significantly improves the problem of model bias as the true value of each parameters are covered in its corresponding confidence interval.

In terms of the data and model in our problem, we regard *just another variable* method as the best performed imputation method among 3 candidates. However, we need to point out that this method leads to the inconsistency of imputed values as we consider the interaction term independent with missing values in other variables when imputing.

```
# consider x1x2 just as another variable
# MI
imp.q4c <- mice(dataex4, m = 50, seed = 1, printFlag = FALSE) # mice() procedure
fit.q4c <- with(imp.q4c, lm(y ~ x1 + x2 + x1x2))              # with() procedure
pool.q4c <- pool(fit.q4c)                                     # pool() procedure

# report the estimates and 95% CIs
summary(pool.q4c, conf.int = TRUE)[, c("term", "estimate", "2.5 %", "97.5 %")]
```

```
##          term estimate     2.5 %   97.5 %
## 1 (Intercept) 1.499714 1.3452011 1.654227
## 2          x1 1.003930 0.8414967 1.166363
## 3          x2 2.026180 1.9398113 2.112548
## 4        x1x2 1.017793 0.9303479 1.105238
```

(d) - Solution

As mentioned in 4.(c), though *just another variable* approach obtains unbiased estimates for each parameter in the linear regression, the inner dependence between the interaction term x_1x_2 and x_1 is violated. In other words, we impute the missing values of x_1x_2 independently from x_1, without considering the deterministic relationship between them.

Question 5

Solution