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 `chl`.

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.

```r
# 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.

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

# 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(seed = i, age = bmi.pool$pooled[2, 10],
                   hyp = bmi.pool$pooled[3, 10],
                   chl = bmi.pool$pooled[4, 10])
  bmi.lambda <- rbind(bmi.lambda, df)
}
bmi.lambda
```

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

```r
# 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(seed = i, age = bmi.pool$pooled[2, 10],
                   hyp = bmi.pool$pooled[3, 10],
                   chl = bmi.pool$pooled[4, 10])
  bmi.lambda.100 <- rbind(bmi.lambda.100, df)
}
bmi.lambda.100
```

```
##   seed       age       hyp       chl
## 1    1 0.4324680 0.2915346 0.3217837
## 2    2 0.4031077 0.2825108 0.2939693
## 3    3 0.3093072 0.2425105 0.3281911
## 4    4 0.3943223 0.2565132 0.2835232
## 5    5 0.3322570 0.2893046 0.2461956
## 6    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.

```r
# 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 $y = \boldsymbol{x\beta} + \epsilon$ 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) $\boldsymbol{\hat{y}} = \left(\hat{y}^{(1)}, \hat{y}^{(2)}, \cdots, \hat{y}^{(M)}\right)$. 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 $\tilde{\beta_0} = \frac{1}{M}\sum\limits_{i=1}^{M}\beta^{(i)}$ and $\tilde{\beta_j} = \frac{1}{M}\sum\limits_{i=1}^{M}\beta_j^{(i)},\ j = 1, \cdots, n$ 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 $\tilde{\beta_0} = \frac{1}{M}\sum\limits_{i=1}^{M}\beta^{(i)}$ and $\tilde{\beta_j} = \frac{1}{M}\sum\limits_{i=1}^{M}\beta_j^{(i)},\ j = 1, \cdots, n$ and further the same predicting expression $\tilde{y} = \tilde{\beta_0} + \sum\limits_{j=1}^{n}\tilde{\beta}_j x_j$ 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 $x_1 x_2$ 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. $\hat{\beta_1} = 1.411 \;\; 95\% CI = (1.219, 1.603)$
2. $\hat{\beta_2} = 1.966 \;\; 95\% CI = (1.861, 2.071)$
3. $\hat{\beta_3} = 0.755 \;\; 95\% CI = (0.642, 0.868)$

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, the coefficient of complete column x_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.

```r
# 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 $x_1 x_2$.

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 $x_1 x_2 = NA$ too.

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

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

# 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$method['x1x2'] <- "~I(x1*x2)"
# x_1 and x_2 should not be imputed by x1x2
imp.q4b.0$predictorMatrix[c("x1", "x2"), "x1x2"] <- 0
# x1x2 should not be imputed by y
imp.q4b.0$predictorMatrix["x1x2", "y"] <- 0

# final 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.

```r
# 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 $x_1 x_2$ and x_1 is violated. In other words, we impute the missing values of $x_1 x_2$ independently from x_1, without considering the deterministic relationship between them.

Question 5

Solution

Exploratory Data Analysis

We start our study with inspecting our data. Our dataset consists of 500 cases with 12 variables including 8 continuous variables and 4 factors:

- `wgt`: weight in kg
- `gender`: male vs female
- `bili`: bilirubin concentration in mg/dL
- `age`: in years
- `chol`: total serum cholesterol in mg/dL,
- `HDL`: High-density lipoprotein cholesterol in mg/dL,
- `hgt`: height in metres,
- `educ`: educational status; 5 ordered categories,
- `race`: 5 unordered categories,
- `SBP`: systolic blood pressure in mmHg,
- `hypten`: hypertensive status; binary,
- `WC`: waist circumference in cm

According to the results from `summary()` function, variables `bili`, `chol`, `HDL`, `hgt`, `educ`, `SBP`, `hypten` and `WC` contains missing values. Note that some missing values in the dataset are presented by the string "NaN" rather than `NA`. Since we are going to use `mice` package for multiple imputation, which expects missing values coded as `NA`, we convert "NaN" to `NA` before any further steps, also for the consistency of notations.

We investigate the missing data patterns intuitively using `md_pattern()` function from `JointAI` package. We can observe that the number of missing values for each variable is not large (47 for `bili` at most and 1 for `educ` at least) and the number of fully observed cases is 411 (out of 500). (We'll revisit this fact later when determining the number of imputed datasets, m)

Moreover, we visualize the correlations between variables using `corrplot()` function from `corrplot` package. Recall that our model of interest is

$$wgt = \beta_0 + \beta_1 gender + \beta_2 age + \beta_3 hgt + \beta_4 WC + \epsilon, \quad \epsilon \sim N(0, \sigma^2)$$

According to the heat map below, we observe that there exists a strong positive correlation between the response variable `wgt` and `WC`. `gender` and `hgt` also have moderate correlations with `wgt`. However, the independent variable `age` shows a relatively low correlation with `wgt`, with a correlation efficient of 0.069 (3 decimal places), which may lead to poor regression performance later. Another noteworthy fact is that `hgt` has a strong negative correlation with `gender`, which will be revisited later as it affects the imputation results of `hgt`.

Finally, we explore the distribution of each variable by visualizing it with `plot_all()` function from `JointAI` package. Continuous variables, except `hgt`, appear to have right-skewed distributions. Thus, using normal distribution for imputation method probably results in poor approximations and we keep using predictive mean matching (pmm) as default. Since we consider `hgt` can be well approximated by normal distribution, the imputation method of it will be modified to `norm`, the normal linear stochastic regression imputation regarding the uncertainty of data.

```r
# read dataset into R
load("NHANES2.Rdata")

# data structure and summary
str(NHANES2)
```

```
## 'data.frame':    500 obs. of  12 variables:
##  $ wgt   : num  78 78 75.3 90.7 112 ...
##  $ gender: Factor w/ 2 levels "male","female": 1 1 2 1 2 1 2 2 1 1 ...
##  $ bili  : num  1.1 0.7 0.5 0.8 0.6 0.7 1.1 0.8 0.8 0.5 ...
##  $ age   : num  67 39 64 36 33 62 56 63 55 20 ...
##  $ chol  : num  6.13 4.65 4.14 3.47 6.31 4.47 6.41 5.51 7.01 3.75 ...
##  $ HDL   : num  1.09 1.14 1.29 1.37 1.27 0.85 1.81 2.38 2.79 1.03 ...
##  $ hgt   : num  1.75 1.78 1.63 1.93 1.73 ...
##  $ educ  : Ord.factor w/ 5 levels "Less than 9th grade"<..: 5 3 5 4 4 3 4 5 4 2 ...
##  $ race  : Factor w/ 5 levels "Mexican American",..: 5 3 5 3 4 5 4 5 3 3 ...
##  $ SBP   : num  139 103 NaN 115 107 ...
##  $ hypten: Factor w/ 2 levels "no","yes": 2 1 2 2 1 2 NA 1 2 1 ...
##  $ WC    : num  91.6 84.5 91.6 95.4 119.6 ...
```

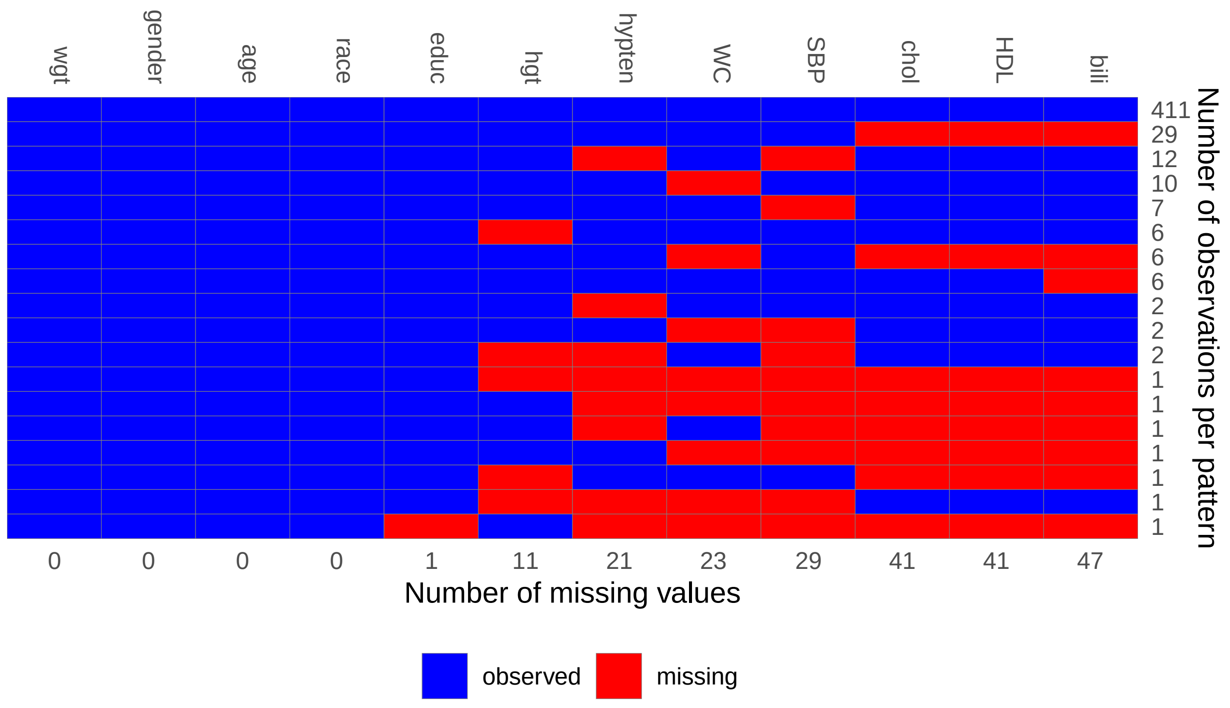
```r
summary(NHANES2)
```

```
##       wgt             gender          bili             age             chol      
##  Min.   : 39.01   male  :252   Min.   :0.2000   Min.   :20.00   Min.   : 2.07  
##  1st Qu.: 65.20   female:248   1st Qu.:0.6000   1st Qu.:31.00   1st Qu.: 4.27  
##  Median : 76.20                Median :0.7000   Median :43.00   Median : 4.86  
##  Mean   : 78.25                Mean   :0.7404   Mean   :45.02   Mean   : 5.00  
##  3rd Qu.: 86.41                3rd Qu.:0.9000   3rd Qu.:58.00   3rd Qu.: 5.64  
##  Max.   :167.38                Max.   :2.9000   Max.   :79.00   Max.   :10.68  
##                                NA's   :47                       NA's   :41     
##       HDL              hgt                           educ    
##  Min.   :0.360   Min.   :1.397   Less than 9th grade : 31  
##  1st Qu.:1.110   1st Qu.:1.626   9-11th grade        : 69  
##  Median :1.320   Median :1.676   High school graduate:115  
##  Mean   :1.395   Mean   :1.687   some college        :148  
##  3rd Qu.:1.590   3rd Qu.:1.753   College or above    :136  
##  Max.   :3.130   Max.   :1.930   NA's                :  1  
##  NA's   :41      NA's   :11                                
##                  race          SBP           hypten          WC        
##  Mexican American  : 52   Min.   : 81.33   no  :354   Min.   : 61.90  
##  Other Hispanic    : 58   1st Qu.:109.00   yes :125   1st Qu.: 84.80  
##  Non-Hispanic White:182   Median :118.67   NA's: 21   Median : 95.00  
##  Non-Hispanic Black:112   Mean   :120.05              Mean   : 96.07  
##  other             : 96   3rd Qu.:128.67              3rd Qu.:104.80  
##                           Max.   :202.00              Max.   :154.70  
##                           NA's   :29                  NA's   :23      
```

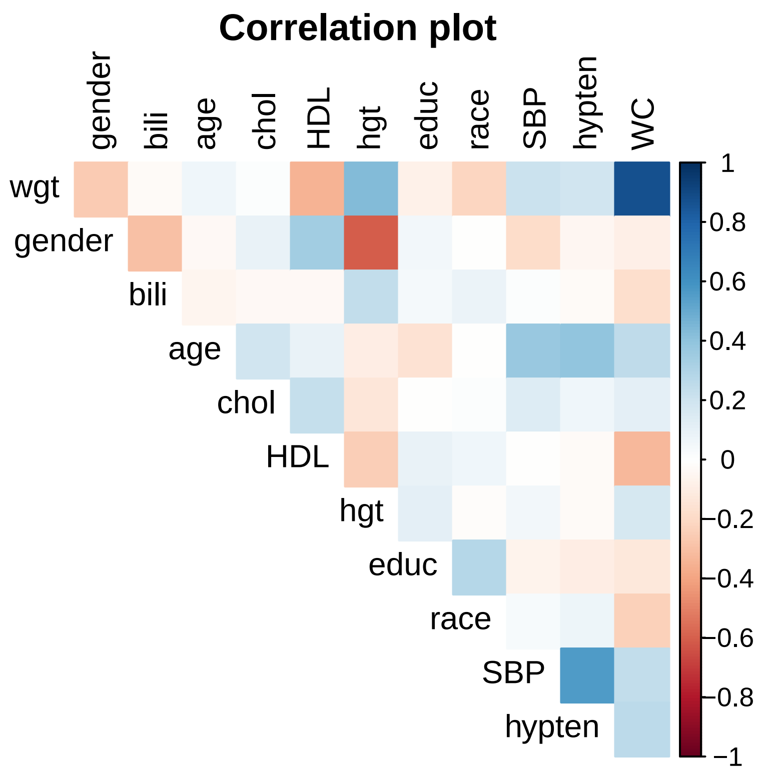
```r
# convert NaN to NA
NHANES2 <- na_if(NHANES2, "NaN")

# transform factors to numeric
col.factor <- which(sapply(NHANES2, is.factor))
```

```
NHANES2.numeric <- NHANES2
NHANES2.numeric[, col.factor] <- sapply(NHANES2[, col.factor], as.numeric)
```

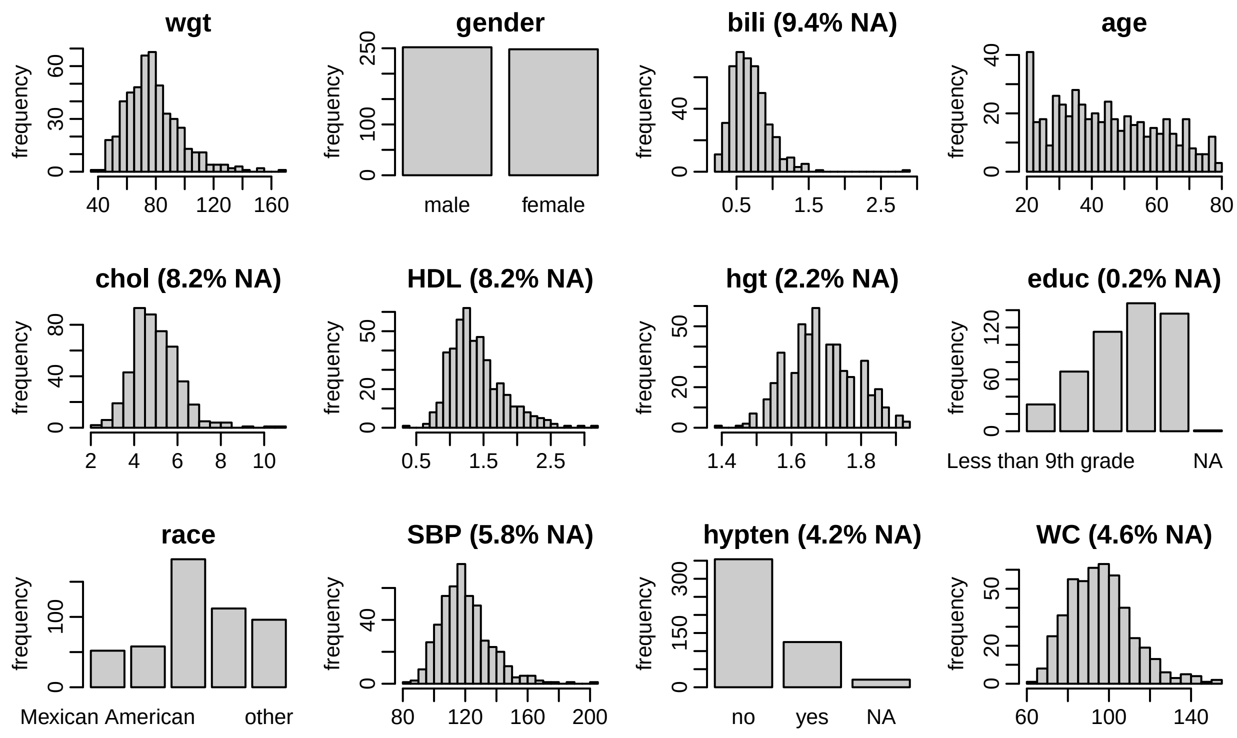
```
# visualize missing pattern
mdp <- md_pattern(NHANES2, pattern = TRUE, color = c("blue", "red"))
mdp$plot
```



```
# correlation plots
cor.NHANES2 <- cor(NHANES2.numeric, use = "complete.obs")
corrplot(cor.NHANES2, method = "color", title = "Correlation plot",
         tl.col = "black", diag = FALSE, type = "upper", mar = c(0, 0, 1, 0))
```



```
# visualize the distribution of observed values in each variable
par(mar = c(3, 3, 2, 1), mgp = c(2, 0.6, 0))
plot_all(NHANES2, breaks = 30, ncol = 4)
```



Multiple Imputation

We begin multiple imputation with a dry/set-up run of `mice()` procedure. From the imputation summary, we observe the default imputation method for continuous variables is `pmm`. As mentioned before, we modify this imputation method for `hgt` to `norm`. Moreover, since `hgt` represents the individual's height in metres, we need to restrict the support the it. Here, we set a reasonable range of `hgt` to $[0.5, 2.8]$.

For factors, the methods are different depending on the number of classes in the factor. `logreg` is used for `hypten` as it is a binary variable, while `polr` is used for `educ` since this variable consists of multiple categories.

Before any further imputation checks, we determine a proper number of imputed datasets, i.e. `m` argument in `mice()` function, based on both accuracy and statistical efficiency. To this end, we run complete multiple imputation procedures including `mice()`, `with()` and `pool()` with a set of candidates of `m` $\in \{5, 10, 15, 20, 25\}$ and repeat the procedures for `seed` $\in \{1, 2, 3, 4, 5, 6\}$. We report the estimates for the regression coefficients as well as the running time using different seed for each candidate of `m`.

As the data frames shown below, the variance of estimated regression coefficients gets smaller as `m` increases. However, this improvement becomes insignificant from `m` $= 20$ to `m` $= 25$. Meanwhile, we notice that the running time of each multiple imputation increases by around 10 seconds as the value of `m` increases by 5 (50 seconds for `m` $= 25$).

Therefore, we prefer a multiple imputation on our dataset with `m` $= 20$, which obtains both computationally efficiency and robust to random variation due to seed choice. Meanwhile, we set `maxit` argument to 20 and keep random seed as 1 for model reproducibility, without the loss of generality.

[**Note:** There is no doubt to use larger value of m (e.g. 50, 100) to improve the model accuracy. However, in this case, the percentage of missing values in each variable is low (9.4% for `bili` at most) and over 80% cases in the dataset are complete. Thus, we believe smaller m can satisfy the requirements of both less variation and statistical efficiency.]

```r
# multiple imputation
# dry mice() procedure
imp0 <- mice(data = NHANES2, maxit = 0)
imp0
```

```
## Class: mids
## Number of multiple imputations:  5
## Imputation methods:
##      wgt   gender     bili      age     chol      HDL      hgt     educ
##       ""       ""    "pmm"       ""    "pmm"    "pmm"    "pmm"   "polr"
##     race      SBP   hypten       WC
##       ""    "pmm" "logreg"    "pmm"
## PredictorMatrix:
##        wgt gender bili age chol HDL hgt educ race SBP hypten WC
## wgt      0      1    1   1    1   1   1    1    1   1      1  1
## gender   1      0    1   1    1   1   1    1    1   1      1  1
## bili     1      1    0   1    1   1   1    1    1   1      1  1
## age      1      1    1   0    1   1   1    1    1   1      1  1
## chol     1      1    1   1    0   1   1    1    1   1      1  1
## HDL      1      1    1   1    1   0   1    1    1   1      1  1
```

```r
# modify the imputation method of 'hgt'
imp0$method["hgt"] <- "norm"
# restrict the support of 'hgt'
imp0$post["hgt"] <- "imp[[j]][,i] <- squeeze(imp[[j]][,i], c(0.5, 2.8))"
```

```r
# MI for m = {5, 10, 15, 20, 25} with seed = {1, 2, 3, 4, 5, 6}
m.set <- c(5, 10, 15, 20, 25)
seed.set <- c(1, 2, 3, 4, 5 ,6)

for (i in m.set){
  est.df <- NULL
  for (j in seed.set){
    time.start <- Sys.time()
    imp.q5 <- mice(NHANES2, maxit = 20, m = i, seed = j,
                   method = imp0$method,
                   post = imp0$post, printFlag = FALSE)
    fit.q5 <- with(imp.q5, lm(wgt ~ gender + age + hgt + WC))
    pool.q5 <- pool(fit.q5)
    time.end <- Sys.time()
    df <- data.frame(seed = j,
                     genderfemale = pool.q5$pooled[2, "estimate"],
                     age = pool.q5$pooled[3, "estimate"],
                     hgt = pool.q5$pooled[4, "estimate"],
                     WC = pool.q5$pooled[5, "estimate"],
                     running.time = time.end - time.start)
    est.df <- rbind(est.df, df)
  }
  cat("regression estimates (m = ", i, "): \n", sep = "")
  print(est.df)
}
```

```
## regression estimates (m = 5):
##   seed genderfemale        age      hgt       WC   running.time
## 1    1    -1.369652 -0.1588398 52.46588 1.026546  9.587889 secs
## 2    2    -1.286194 -0.1565471 53.26393 1.026603  9.383312 secs
## 3    3    -1.363540 -0.1606729 52.83642 1.023533  9.461433 secs
## 4    4    -1.391903 -0.1611415 52.35877 1.028648  9.820780 secs
## 5    5    -1.331400 -0.1588171 52.89258 1.025179 10.974349 secs
## 6    6    -1.334292 -0.1569351 52.97270 1.025940  9.446540 secs
## regression estimates (m = 10):
##   seed genderfemale        age      hgt       WC  running.time
## 1    1    -1.318411 -0.1567395 52.58736 1.026748 18.85325 secs
## 2    2    -1.305026 -0.1589174 52.90901 1.027096 21.18386 secs
## 3    3    -1.362416 -0.1546543 52.36162 1.024514 18.34972 secs
## 4    4    -1.420964 -0.1583875 52.39275 1.025138 18.41450 secs
## 5    5    -1.306057 -0.1576264 52.85176 1.024285 18.34911 secs
## 6    6    -1.368114 -0.1558868 52.13114 1.026035 18.44879 secs
## regression estimates (m = 15):
##   seed genderfemale        age      hgt       WC  running.time
## 1    1    -1.416849 -0.1592273 52.25315 1.026394 27.45702 secs
## 2    2    -1.361419 -0.1571292 52.56111 1.025913 28.22571 secs
## 3    3    -1.324025 -0.1561752 52.43867 1.025362 28.92964 secs
## 4    4    -1.331270 -0.1579811 52.66479 1.024820 28.99621 secs
## 5    5    -1.400804 -0.1581106 52.03306 1.025732 27.50643 secs
## 6    6    -1.351421 -0.1570642 52.42774 1.025911 27.38810 secs
## regression estimates (m = 20):
##   seed genderfemale        age      hgt       WC  running.time
## 1    1    -1.323852 -0.1555430 52.49801 1.024904 37.10128 secs
## 2    2    -1.363742 -0.1569720 52.70480 1.025105 35.88593 secs
## 3    3    -1.346148 -0.1575175 52.72683 1.025153 36.03132 secs
## 4    4    -1.347170 -0.1561035 52.44223 1.025811 35.86080 secs
## 5    5    -1.368667 -0.1584354 52.42954 1.025529 35.82679 secs
## 6    6    -1.373698 -0.1594694 52.28251 1.026873 36.91763 secs
## regression estimates (m = 25):
##   seed genderfemale        age      hgt       WC  running.time
## 1    1    -1.354632 -0.1591073 52.48408 1.025561 47.56478 secs
## 2    2    -1.319659 -0.1570791 52.70967 1.025361 45.04032 secs
## 3    3    -1.358916 -0.1568233 52.43961 1.025951 48.57463 secs
## 4    4    -1.375729 -0.1576124 52.16183 1.025448 48.26946 secs
## 5    5    -1.333896 -0.1579284 52.52684 1.025330 45.77623 secs
## 6    6    -1.311311 -0.1584013 52.75073 1.024948 45.76226 secs
```

```
# MI with m = 20 and seed = 1
imp.q5 <- mice(NHANES2, maxit = 20, m = 20, seed = 1,
               method = imp0$method,
               post = imp0$post, printFlag = FALSE)
fit.q5 <- with(imp.q5, lm(wgt ~ gender + age + hgt + WC))
pool.q5 <- pool(fit.q5)
```

(The imputation check continues in the next page.)

Imputation Check

After applying a multiple imputation with proper arguments to our data, we conduct necessary checks.

We first check if `mice()` found any problems during imputation with object `loggedEvents` and obtain the result `NULL`, indicating no problems detected. Furthermore, we check the Monte Carlo chains to ensure the convergence across all 12 variables. From the plots with 20 MC chains for each variable, there exists no obvious pattern among well-mixing chains, suggesting good convergence for all variables. Note that the plot for the standard deviation of `educ` is blank as there is only 1 missing values in this column.

Next, we inspect if the distribution of the imputed values agree with the distribution of the observed values, using `densityplot()` as well as `bwplot()` functions from `mice` package for continuous variables and `propplot()` function for categorical variables.

For continuous variables, except `hgt`, comparing with the distribution of their observed values (*blue*), their imputed values follow the similar distribution (*red*). On the other hand, we can observe a clear left-shifted distribution of the imputed values of `hgt`.

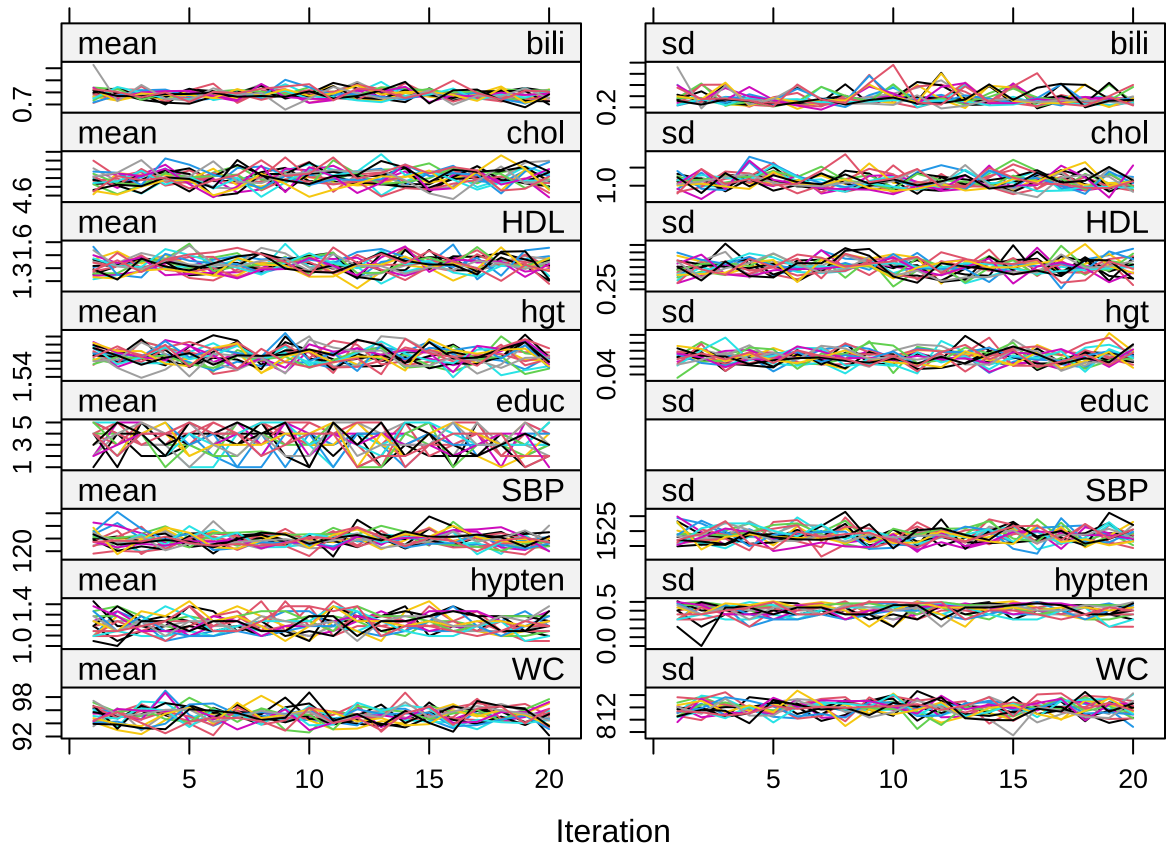
Recall that we observed high correlation between `hgt` and `gender` from the heat map. We guess this is the main reason for the shifted distribution of imputed `hgt`. To prove this, we next make a density plot of `hgt`, conditional on `gender`. As a result, we observe that `gender` does strongly influences the imputed datasets. Specifically, values in `male` group has much more narrow, concentrated distributions while the distribution in `female` group is wider.

For categorical variables, the distribution of imputed values of `educ` appears to be abnormal simply because we only imputed quite a few values each time, specifically 1. The similar reason can also explain the discrepancies between the observed and imputed values for `hypten` (the percentage of missing values in this column is 4.2%).

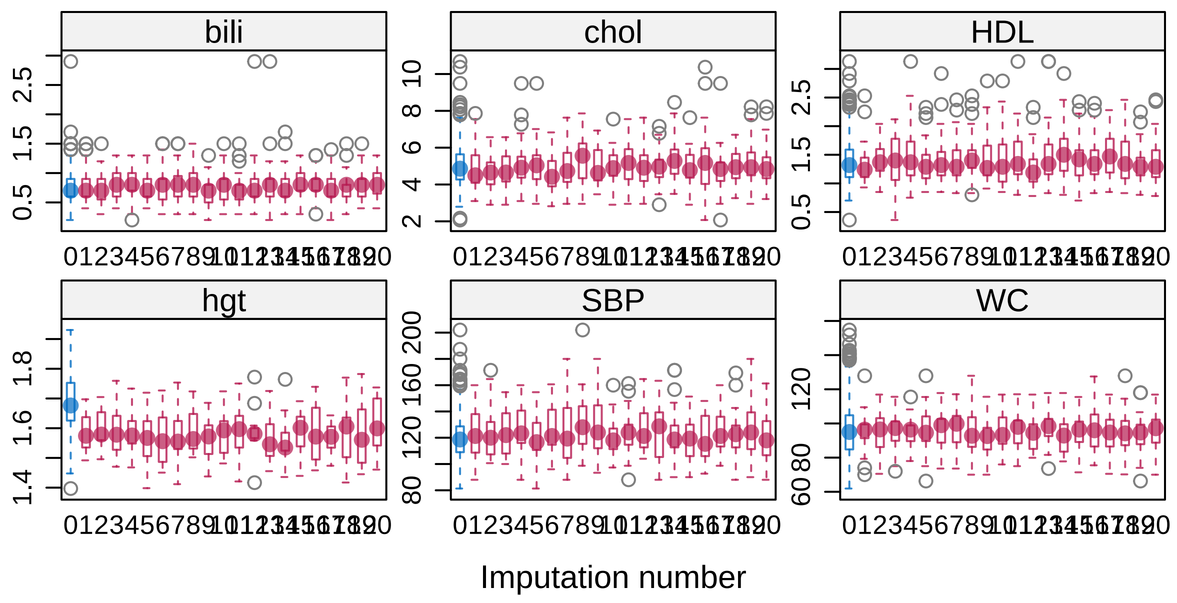
```r
# necessary checks for mice() procedure
# problems check during mice() procedure
imp.q5$loggedEvents
```

```
## NULL
```

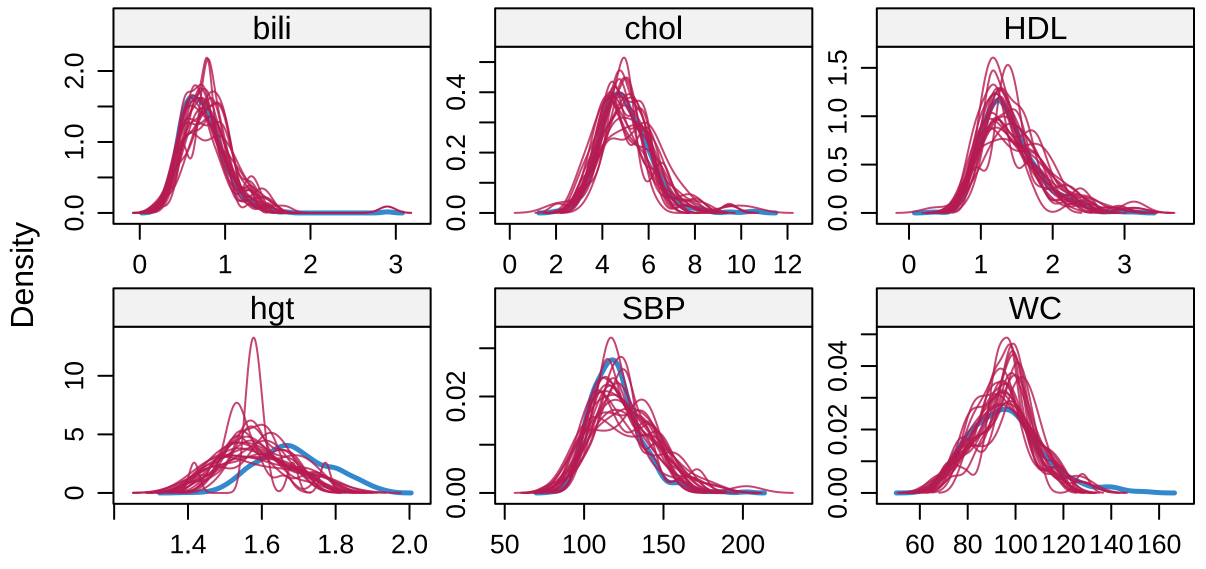
```r
# convergence check
plot(imp.q5, layout = c(2, 8))
```



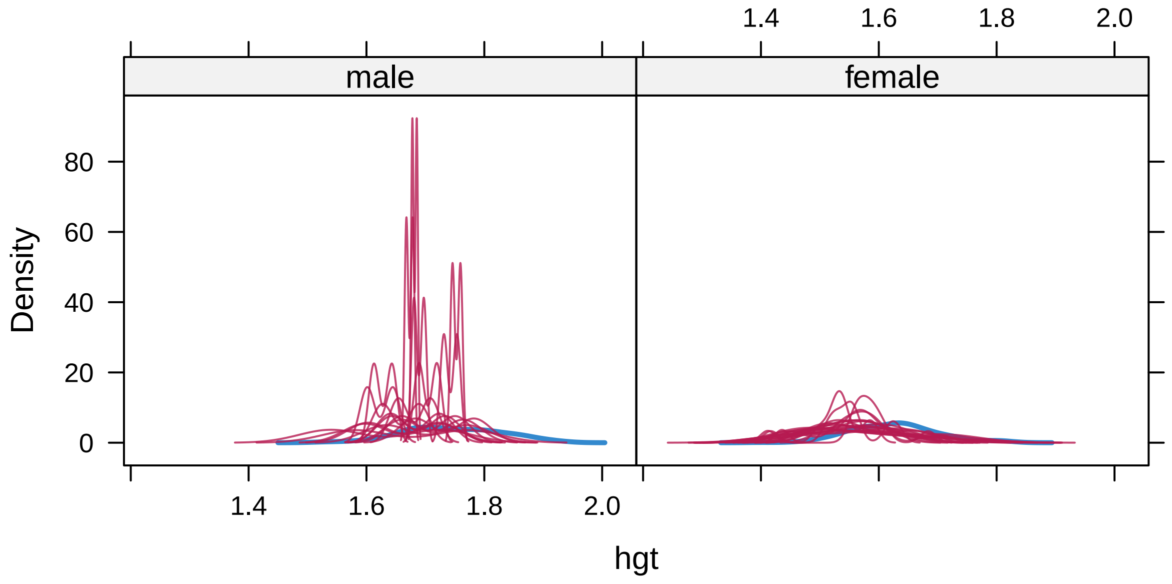
```
# distribution check
# case for continuous variables
bwplot(imp.q5)[c(2, 4, 5, 6, 7, 8)]
```



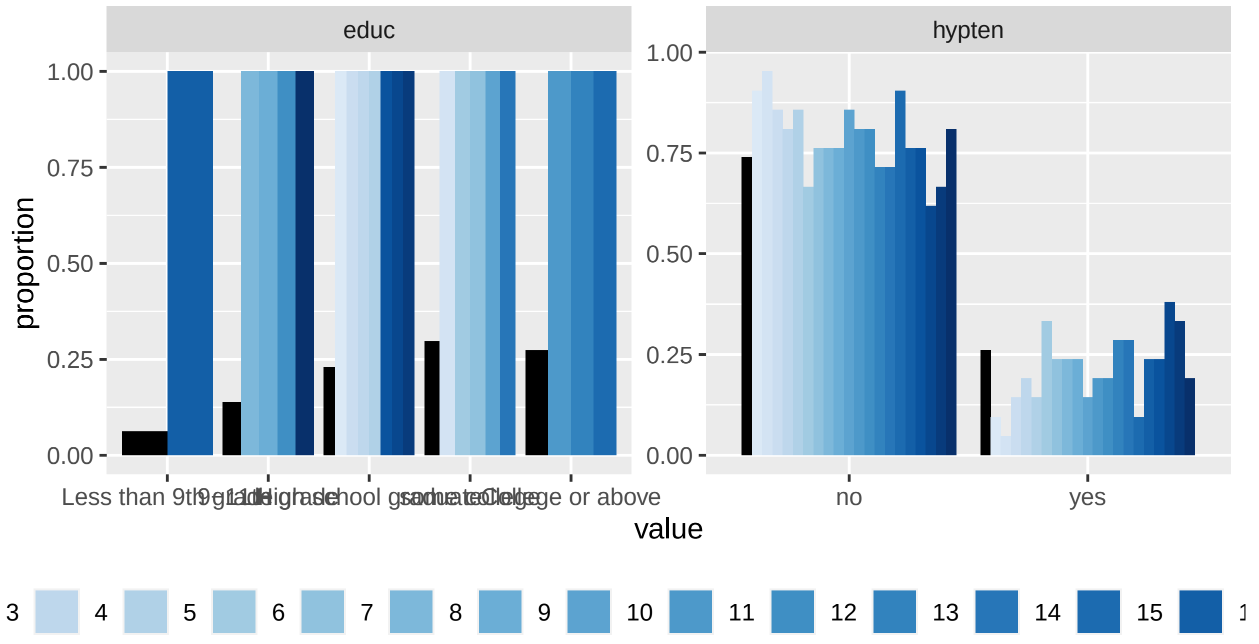
```r
densityplot(imp.q5)
```



```r
# distribution of hgt conditional on gender
densityplot(imp.q5, ~ hgt | gender)
```



```r
# case for categorised variables
propplot(imp.q5)
```



(The regression check and conclusions continue in the next page.)

Regression Check

We have confirmed that our imputation step is successful. Before we reach the final conclusions, we also need to check if our model of interest fits well the *completed* data. Moreover, we check if model performances across different subsets are consistent. Thus, we display the regression summaries for both subset 1 and 20 with plots.

According to the regression summaries, we observe that all independent variables, except `genderfemale` (`gender`), are significant with relatively low p-values. Moreover, R^2 around 85 indicates good model explanation on response variable. Looking at the residual plots, we cannot observe obvious pattern of residuals, suggesting the linear assumption is met. The well-performed normal Q-Q plots shows the normality of error terms is satisfied. Thus, we consider our model to be well fitted.

The insignificance of `genderfemale` (`gender`) is also confirmed by implementing a multivariate Wald test. We fit another model without `gender` and compare the two models. The p-value of 0.1113 indicates `gender` has no relevant contribution to the `wgt` model.

Recall that there is a strong linear relationship between `gender` and `hgt`, which influences the distribution of imputed values of `hgt`. Here, we may also consider it as a reason for the insignificant contribution of `gender`. Though the regression model appears to be good, we still need to solve this problem in future research to improve model performance. A easy and reasonable way is to drop `gender` from the model.

Since the results from different subsets are similar, we consider our models are consistent across subsets. Therefore, we finally pool the regression results and report the estimates of regression coefficients with their 95% confidence intervals and p-values. The regression expression is:

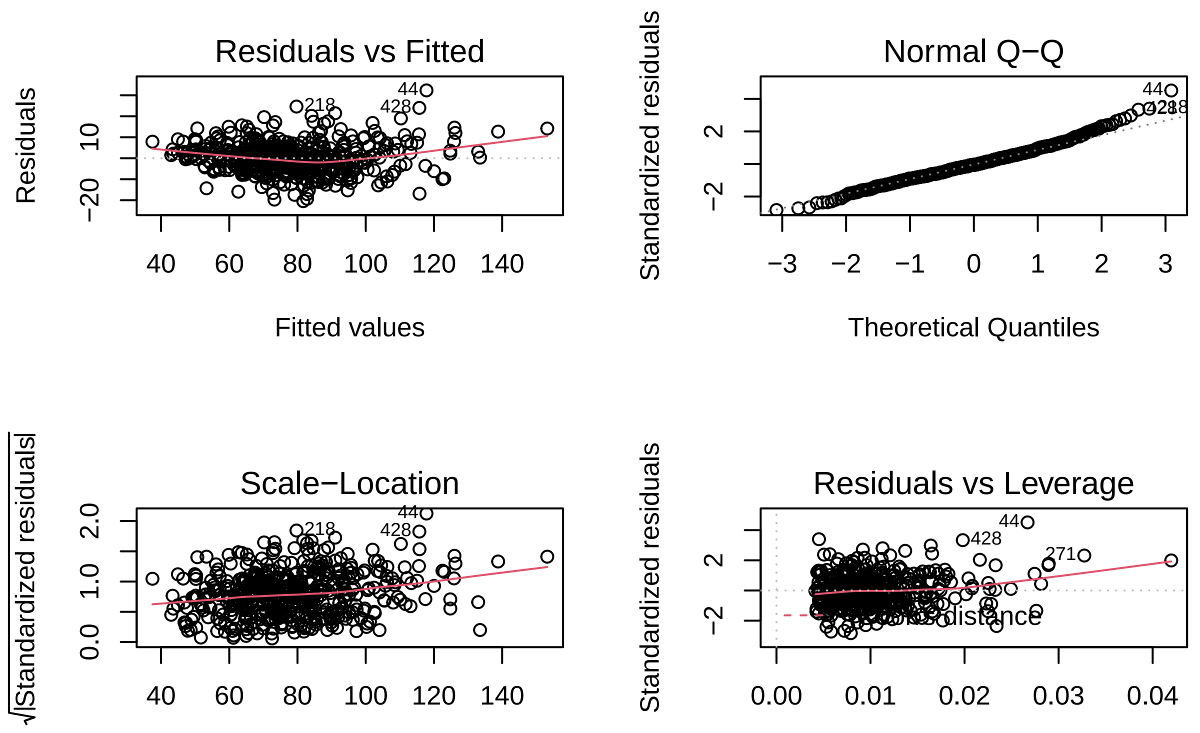
$$wgt = -100.960 - 1.324 \times genderfemale - 0.156 \times age + 52.498 \times hgt + 1.025 \times WC$$

As expected, the coefficient of `gender` is insignificant. Moreover, we report the pooled adjusted R^2, where the value of around 0.85 indicates good model explanation.

```r
# with() procedure (regression) check
summary(fit.q5$analyses[[1]])
```

```
##
## Call:
## lm(formula = wgt ~ gender + age + hgt + WC)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -20.463  -4.585  -0.385   4.162  32.323
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -104.98644    7.55808 -13.891  < 2e-16 ***
## genderfemale   -1.23947    0.83155  -1.491    0.137
## age            -0.15348    0.02106  -7.288 1.25e-12 ***
## hgt            55.12737    4.30910  12.793  < 2e-16 ***
## WC              1.01901    0.02225  45.792  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.259 on 495 degrees of freedom
## Multiple R-squared:  0.8543, Adjusted R-squared:  0.8531
## F-statistic: 725.6 on 4 and 495 DF,  p-value: < 2.2e-16
```

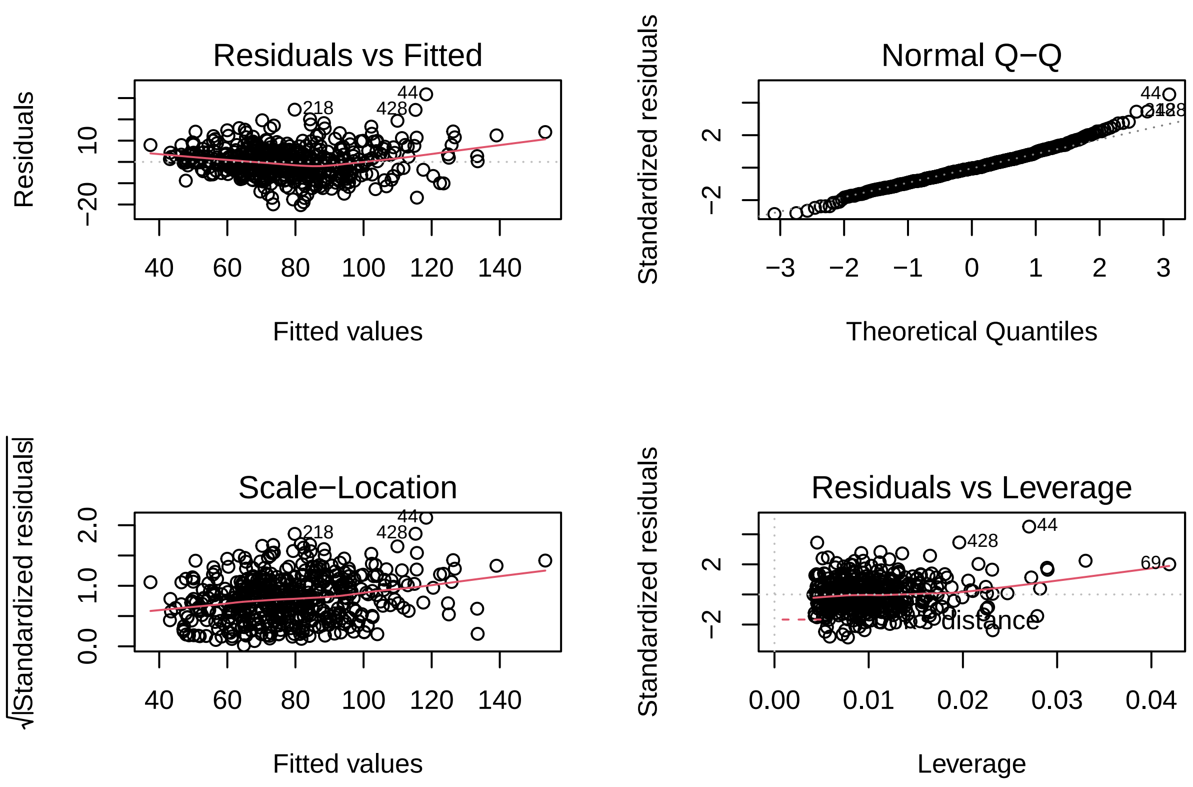
```
par(mfrow = c(2, 2))
plot(fit.q5$analyses[[1]])
```



```
summary(fit.q5$analyses[[20]])
```

```
##
## Call:
## lm(formula = wgt ~ gender + age + hgt + WC)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -20.321  -4.552  -0.353   3.933  31.751
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -101.63310    7.39209 -13.749  < 2e-16 ***
## genderfemale   -1.31788    0.81265  -1.622    0.106
## age            -0.15988    0.02073  -7.714 6.76e-14 ***
## hgt            52.72579    4.22001  12.494  < 2e-16 ***
## WC              1.02899    0.02197  46.843  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.138 on 495 degrees of freedom
## Multiple R-squared:  0.8591, Adjusted R-squared:  0.858
## F-statistic: 754.6 on 4 and 495 DF,  p-value: < 2.2e-16
```

```
par(mfrow = c(2, 2))
plot(fit.q5$analyses[[20]])
```



```
# Wald test
fit.q5.nogender <- with(imp.q5, lm(wgt ~ age + hgt + WC))
D1(fit.q5, fit.q5.nogender)
```

```
##    test statistic df1      df2 dfcom  p.value        riv
##  1 ~~ 2  2.544746   1 483.5937   495 0.111315 0.02206217
```

```
# pool() procedure check
pool.q5
```

```
## Class: mipo    m = 20
##            term  m    estimate         ubar            b            t dfcom
## 1   (Intercept) 20 -100.960341 5.526751e+01 3.949302e+00 5.941428e+01   495
## 2 genderfemale 20   -1.323852 6.738407e-01 1.415847e-02 6.887071e-01   495
## 3          age 20   -0.155543 4.346036e-04 2.988707e-05 4.659850e-04   495
## 4          hgt 20   52.498011 1.805798e+01 1.323640e+00 1.944780e+01   495
## 5           WC 20    1.024904 4.878153e-04 1.249150e-05 5.009314e-04   495
##         df        riv     lambda        fmi
## 1 410.3546 0.07503083 0.06979412 0.07429488
## 2 476.7304 0.02206217 0.02158594 0.02566495
## 3 414.3349 0.07220702 0.06734429 0.07181387
## 4 407.6217 0.07696442 0.07146422 0.07598680
## 5 471.9279 0.02688739 0.02618339 0.03028429
```

```
summary.q5 <- summary(pool.q5, conf.int = TRUE)
summary.q5
```

```
##           term    estimate  std.error  statistic       df      p.value
## 1  (Intercept) -100.960341 7.70806594 -13.098012 410.3546 0.000000e+00
## 2 genderfemale   -1.323852 0.82988381  -1.595226 476.7304 1.113244e-01
## 3          age   -0.155543 0.02158669  -7.205505 414.3349 2.753797e-12
## 4          hgt   52.498011 4.40996593  11.904403 407.6217 0.000000e+00
## 5           WC    1.024904 0.02238150  45.792460 471.9279 0.000000e+00
##          2.5 %      97.5 %
## 1 -116.1125628 -85.8081195
## 2   -2.9545344   0.3068302
## 3   -0.1979761  -0.1131099
## 4   43.8288964  61.1671255
## 5    0.9809240   1.0688835
```

```
# report regression coefficients and CIs with p-values
df <- data.frame(estimate = summary.q5[, 2],
                 lower = summary.q5[, 7],
                 upper = summary.q5[, 8],
                 p.value = summary.q5[, 6],
                 row.names = c("$\\beta_0$", "$\\beta_1$",
                               "$\\beta_2$", "$\\beta_3$", "$\\beta_4$"))
colnames(df) <- c("estimate", "2.5% quantile", "97.5% quantile", "p-value")
knitr::kable(df, escape = FALSE, digits = 3,
             caption = "Regression coefficients, 95% CIs and p-values")
```

Table 1: Regression coefficients, 95% CIs and p-values

	estimate	2.5% quantile	97.5% quantile	p-value
β_0	-100.960	-116.113	-85.808	0.000
β_1	-1.324	-2.955	0.307	0.111
β_2	-0.156	-0.198	-0.113	0.000
β_3	52.498	43.829	61.167	0.000
β_4	1.025	0.981	1.069	0.000

```
# pooled R^2
pool.r.squared(pool.q5, adjusted = TRUE)
```

```
##                est     lo 95     hi 95        fmi
## adj R^2 0.8563792 0.8306254 0.8784993 0.04115452
```