Incomplete Data Analysis - Assignment 3

Chen Tao

I have used R Markdown to complete this assignment this time and you can find the R Markdown document for this assignment in my GitHub repository (https://github.com/TAO-Chen632/Incomplete-Data-Analysis.git). It is stored in the folder named "Assignment 3" and my GitHub Username is "TAO-Chen632".

```
# Setup - Load all the packages that will be used in this assignment
knitr::opts_chunk$set(echo = TRUE)
library(mice)
library(JointAI)
library(ggplot2)
library(devtools)
library(reshape2)
library(RColorBrewer)
library(knitr)
```

Question 1. Solutions.

(a) Let me load and examine the data set `nhanes`.

```
data("nhanes")
dim(nhanes)
```

```
## [1] 25  4
```

```
summary(nhanes)
```

```
##       age            bmi             hyp             chl
##  Min.   :1.00   Min.   :20.40   Min.   :1.000   Min.   :113.0
##  1st Qu.:1.00   1st Qu.:22.65   1st Qu.:1.000   1st Qu.:185.0
##  Median :2.00   Median :26.75   Median :1.000   Median :187.0
##  Mean   :1.76   Mean   :26.56   Mean   :1.235   Mean   :191.4
##  3rd Qu.:2.00   3rd Qu.:28.93   3rd Qu.:1.000   3rd Qu.:212.0
##  Max.   :3.00   Max.   :35.30   Max.   :2.000   Max.   :284.0
##                 NA's   :9       NA's   :8       NA's   :10
```

From the output, we can see that this data set has 4 variables and 25 observations. The variable `age` is fully observed and the variables `bmi`, `hyp` and `chl` are subject to missingness. Overall, this data set has 27 missing values in total and 12 observations have missing values. Hence, the percentage of the cases is incomplete is $12/25 \times 100\% = 48\%$.

(b) Now, I am going to impute the data with the function `mice`. In step 2, our substantive model of interest is the normal linear regression model:

$$bmi = \beta_0 + \beta_1 age + \beta_2 hyp + \beta_3 chl + \varepsilon, \quad \varepsilon \sim N(0, \sigma^2).$$

```
# Impute the data with `mice` using the defaults with seed = 1
impu_Q1 <- mice(nhanes, seed = 1, printFlag = FALSE)
# Predict `bmi` from `age`, `hyp` and `chl`
fit_Q1 <- with(impu_Q1, lm(bmi ~ age + hyp + chl))
# Pool the results
estims_Q1 <- pool(fit_Q1)
# Show the pooled results
estims_Q1
```

```
## Class: mipo    m = 5
##          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
```

From the outcome, we can see that the proportions of variance due to the missing data for the parameters β_0, β_1, β_2 and β_3 are 0.08938989, 0.68640637, 0.35043452 and 0.30408063 respectively, which are the values in the column `lambada`. The parameter β_1 which corresponds to the variable `age` appears to be most affected by the nonresponse, because its proportion of variance due to the missing data is the largest among all the parameters.

(c) The R code for this question is shown below.

```
# Repeat the analysis for seed = 2, 3, 4, 5, 6
pool(with(mice(nhanes, seed = 2, printFlag = FALSE), lm(bmi ~ age + hyp + chl)))
```

```
## Class: mipo    m = 5
##          term m   estimate        ubar            b           t dfcom
## 1 (Intercept) 5 19.9464142 11.149036812 6.5759123307 19.040131609    21
## 2         age 5 -4.0615093  0.963673574 0.5429846700  1.615255178    21
## 3         hyp 5  1.5304762  3.480239781 0.4843237089  4.061428231    21
## 4         chl 5  0.0628349  0.000345515 0.0001210591  0.000490786    21
##          df       riv    lambda       fmi
## 1  7.595481 0.7077826 0.4144454 0.5249745
## 2  7.827560 0.6761435 0.4033924 0.5135940
## 3 15.210844 0.1669967 0.1430995 0.2372083
## 4 10.450088 0.4204476 0.2959966 0.4006804
```

```r
pool(with(mice(nhanes, seed = 3, printFlag = FALSE), lm(bmi ~ age + hyp + chl)))
```

```
## Class: mipo    m = 5
##          term m    estimate         ubar            b            t dfcom
## 1 (Intercept) 5 20.55844343 1.140358e+01 3.6461139762 1.577892e+01    21
## 2         age 5 -3.85753338 9.589920e-01 1.1476609660 2.336185e+00    21
## 3         hyp 5  1.35281238 2.848804e+00 1.6505168435 4.829425e+00    21
## 4         chl 5  0.05872834 3.258177e-04 0.0003485725 7.441047e-04    21
##          df       riv    lambda       fmi
## 1 10.976712 0.3836809 0.2772900 0.3807063
## 2  4.685401 1.4360840 0.5895051 0.6963297
## 3  7.685607 0.6952461 0.4101152 0.5205225
## 4  5.059752 1.2838067 0.5621346 0.6707894
```

```r
pool(with(mice(nhanes, seed = 4, printFlag = FALSE), lm(bmi ~ age + hyp + chl)))
```

```
## Class: mipo    m = 5
##          term m    estimate         ubar            b            t dfcom
## 1 (Intercept) 5 19.39540373 1.111012e+01 1.4019630652 1.279247e+01    21
## 2         age 5 -3.50603350 8.876847e-01 0.2073486445 1.136503e+00    21
## 3         hyp 5  2.75053046 3.247001e+00 0.6600844963 4.039103e+00    21
## 4         chl 5  0.04920611 2.813334e-04 0.0001157514 4.202351e-04    21
##          df       riv    lambda       fmi
## 1 15.591350 0.1514256 0.1315114 0.2249407
## 2 12.740142 0.2803004 0.2189333 0.3181785
## 3 13.470677 0.2439486 0.1961083 0.2937232
## 4  9.532048 0.4937264 0.3305334 0.4373741
```

```r
pool(with(mice(nhanes, seed = 5, printFlag = FALSE), lm(bmi ~ age + hyp + chl)))
```

```
## Class: mipo    m = 5
##          term m    estimate         ubar            b            t dfcom
## 1 (Intercept) 5 19.17135935 1.146234e+01 9.016196e+00 2.228178e+01    21
## 2         age 5 -3.49672250 9.930838e-01 6.803640e-01 1.809521e+00    21
## 3         hyp 5  1.50954775 3.548122e+00 4.331055e+00 8.745388e+00    21
## 4         chl 5  0.06081272 3.331396e-04 8.509425e-05 4.352527e-04    21
##          df       riv    lambda       fmi
## 1  6.252823 0.9439114 0.4855733 0.5967667
## 2  6.870551 0.8221228 0.4511896 0.5623912
## 3  4.622453 1.4647938 0.5942866 0.7007388
## 4 12.250235 0.3065175 0.2346065 0.3349844
```

```r
pool(with(mice(nhanes, seed = 6, printFlag = FALSE), lm(bmi ~ age + hyp + chl)))
```

```
## Class: mipo    m = 5
##          term m    estimate         ubar            b            t dfcom
## 1 (Intercept) 5 20.52083805 10.289853169 6.1286156960 1.764419e+01    21
## 2         age 5 -2.92141353  0.750542947 1.1872026088 2.175186e+00    21
## 3         hyp 5  1.22474596  3.190304998 1.1180105516 4.531918e+00    21
## 4         chl 5  0.04949218  0.000308532 0.0002781228 6.422793e-04    21
##           df       riv    lambda       fmi
## 1  7.546621 0.7147176 0.4168136 0.5274057
## 2  3.879068 1.8981500 0.6549523 0.7552705
## 3 10.448987 0.4205280 0.2960364 0.4007229
## 4  5.693290 1.0817266 0.5196295 0.6301448
```

From the results, we can see that the proportions of variance due to the missing data for each parameter, i.e., the values in the column `lambada`, vary a lot as the random seed changes, and which is larger or smaller is not fixed. Therefore, the conclusions of question (b) do not remain the same. The parameter that appears to be most affected by the missing values varies as the random seed changes. When `seed = 1`, such parameter is β_0, and when `seed = 3` or `seed = 6`, such parameter is β_1, and when `seed = 5`, such parameter is β_2, and when `seed = 4`, such parameter is β_3.

(d) The R code for this question is shown in the following.

```
# Repeat the analysis with M = 100 and the seed = 1, 2, 3, 4, 5, 6
pool(with(mice(nhanes, m = 100, seed = 1, printFlag = FALSE), lm(bmi ~ age + hyp + chl)))
```

```
## Class: mipo    m = 100
##          term   m    estimate         ubar            b            t dfcom
## 1 (Intercept) 100 20.41190804 1.094142e+01 3.2184206709 1.419202e+01    21
## 2         age 100 -3.64125507 9.577488e-01 0.7225930508 1.687568e+00    21
## 3         hyp 100  1.68579437 3.320966e+00 1.3530518833 4.687548e+00    21
## 4         chl 100  0.05449706 3.267466e-04 0.0001534919 4.817734e-04    21
##          df       riv    lambda       fmi
## 1 14.72509 0.2970918 0.2290445 0.3160348
## 2 10.70407 0.7620151 0.4324680 0.5152948
## 3 13.48013 0.4115015 0.2915346 0.3775127
## 4 12.87979 0.4744559 0.3217837 0.4072025
```

```
pool(with(mice(nhanes, m = 100, seed = 2, printFlag = FALSE), lm(bmi ~ age + hyp + chl)))
```

```
## Class: mipo    m = 100
##          term   m   estimate         ubar           b            t dfcom
## 1 (Intercept) 100 20.4677977 1.075030e+01 2.468337199 1.324332e+01    21
## 2         age 100 -3.6337887 9.515254e-01 0.636244576 1.594132e+00    21
## 3         hyp 100  1.7199145 3.289619e+00 1.282460003 4.584904e+00    21
## 4         chl 100  0.0535212 3.185157e-04 0.000131307 4.511357e-04    21
##          df       riv    lambda       fmi
## 1 15.53932 0.2319024 0.1882474 0.2758183
## 2 11.27749 0.6753440 0.4031077 0.4867207
## 3 13.65957 0.3937491 0.2825108 0.3686461
## 4 13.43174 0.4163689 0.2939693 0.3799043
```

```
pool(with(mice(nhanes, m = 100, seed = 3, printFlag = FALSE), lm(bmi ~ age + hyp + chl)))
```

```
## Class: mipo    m = 100
##          term   m    estimate         ubar            b            t dfcom
## 1 (Intercept) 100 20.37418806 1.127465e+01 3.1478225057 1.445395e+01    21
## 2         age 100 -3.55706093 9.804071e-01 0.4347005920 1.419455e+00    21
## 3         hyp 100  1.55756211 3.341601e+00 1.0592224955 4.411416e+00    21
## 4         chl 100  0.05445409 3.289884e-04 0.0001591257 4.897054e-04    21
##         df       riv    lambda       fmi
## 1 14.90637 0.2819867 0.2199607 0.3070849
## 2 13.12717 0.4478217 0.3093072 0.3949630
## 3 14.45645 0.3201504 0.2425105 0.3292967
## 4 12.75289 0.4885186 0.3281911 0.4134845
```

```r
pool(with(mice(nhanes, m = 100, seed = 4, printFlag = FALSE), lm(bmi ~ age + hyp + chl)))
```

```
## Class: mipo    m = 100
##          term   m    estimate         ubar            b            t dfcom
## 1 (Intercept) 100 20.38340913 1.101247e+01 2.9769590315 1.401920e+01    21
## 2         age 100 -3.64815327 9.385945e-01 0.6050152967 1.549660e+00    21
## 3         hyp 100  1.65946951 3.328433e+00 1.1369855186 4.476789e+00    21
## 4         chl 100  0.05511595 3.223373e-04 0.0001262919 4.498921e-04    21
##         df       riv    lambda       fmi
## 1 15.01591 0.2730295 0.2144722 0.3016760
## 2 11.44963 0.6510431 0.3943223 0.4781553
## 3 14.17726 0.3450138 0.2565132 0.3430796
## 4 13.63943 0.3957186 0.2835232 0.3696411
```

```r
pool(with(mice(nhanes, m = 100, seed = 5, printFlag = FALSE), lm(bmi ~ age + hyp + chl)))
```

```
## Class: mipo    m = 100
##          term   m    estimate         ubar            b            t dfcom
## 1 (Intercept) 100 20.29458168 1.078780e+01 3.1802645878 1.399987e+01    21
## 2         age 100 -3.76297245 9.535370e-01 0.4697652835 1.428000e+00    21
## 3         hyp 100  1.80283168 3.145219e+00 1.2676555641 4.425551e+00    21
## 4         chl 100  0.05534382 3.186734e-04 0.0001030496 4.227535e-04    21
##         df       riv    lambda       fmi
## 1 14.71729 0.2977500 0.2294356 0.3164200
## 2 12.67241 0.4975821 0.3322570 0.4174695
## 3 13.52446 0.4070725 0.2893046 0.3753219
## 4 14.38295 0.3266041 0.2461956 0.3329248
```

```r
pool(with(mice(nhanes, m = 100, seed = 6, printFlag = FALSE), lm(bmi ~ age + hyp + chl)))
```

```
## Class: mipo    m = 100
##          term   m    estimate         ubar            b            t dfcom
## 1 (Intercept) 100 20.26848393 1.103092e+01 3.5875749339 1.465437e+01    21
## 2         age 100 -3.59309185 9.478791e-01 0.7465052532 1.701849e+00    21
## 3         hyp 100  1.86219629 3.242906e+00 1.2865589489 4.542330e+00    21
## 4         chl 100  0.05319323 3.210547e-04 0.0001436891 4.661807e-04    21
##         df       riv    lambda       fmi
## 1 14.36172 0.3284812 0.2472607 0.3339733
## 2 10.49851 0.7954288 0.4430300 0.5255532
## 3 13.58878 0.4006976 0.2860700 0.3721439
## 4 13.08746 0.4520290 0.3113085 0.3969269
```

I prefer the analyses with $M = 100$ to the analyses with $M = 5$. On one hand, it is because when the number M of the copies of data sets increases, the results of the analysis become more stable and we can be more confident about them in any one specific run. For example, when $M = 100$ the estimates of the regression model parameters only differ by a small amount in each case. On the other hand, the extra simulation variance B/M caused by the fact that the multiple imputation estimate of the parameter is estimated by finite M will decrease if M increases. Consequently, when M increases, the total variance of the estimate $V^{MI} = \bar{U} + B + \frac{B}{M}$ and the proportion of variance in the parameter due to missing values $\lambda = \frac{B+\frac{B}{M}}{V^{MI}}$ will all decrease.

Question 2. Solutions.

The R code for this question is shown in the following.

```r
# Load the data and check the dimension of it
load("dataex2.Rdata")
dim(dataex2)
```

```
## [1] 100   2 100
```

```r
# Initialize some variables
beta1 <- 3
# The variables `times1` and `times2` represent the times that the 95% confidence interval
# contains the true value of beta1 under the two imputation approaches respectively.
times1 <- times2 <- 0
# n is the number of the data sets.
n <- dim(dataex2)[3]

# Do the setup run of `mice()` function
impu0_Q2 <- mice(dataex2[ , , 1], maxit = 0)
# Extract and modify the imputation methods
meth1 <- meth2 <- impu0_Q2$method
meth1["Y"] <- "norm.nob"
meth2["Y"] <- "norm.boot"

for (i in seq(n)){
  data <- dataex2[ , , i]
  # Perform the improper multiple imputation using stochastic regression imputation
  impu1_Q2 <- mice(data, method = meth1, m = 20, seed = 1, printFlag = FALSE)
  fit1_Q2 <- with(impu1_Q2, lm(Y ~ X))
  estims1_Q2 <- pool(fit1_Q2)

  # Calculate the lower bound and upper bound of the 95% confidence interval
  # for beta1 under the first approach
  lowerbound <- summary(estims1_Q2, conf.int = TRUE)[[7]][2]
  upperbound <- summary(estims1_Q2, conf.int = TRUE)[[8]][2]
  # Update the variable `times1`
  if (beta1 >= lowerbound & beta1 <= upperbound){
    times1 <- times1 + 1
  }
  # Perform the proper multiple imputation using the bootstrap
  # based stochastic regression imputation
  impu2_Q2 <- mice(data, method = meth2, m = 20, seed = 1, printFlag = FALSE)
  fit2_Q2 <- with(impu2_Q2, lm(Y ~ X))
  estims2_Q2 <- pool(fit2_Q2)

  # Calculate the lower bound and upper bound of the 95% confidence interval
  # for beta1 under the second approach
  lowerbound <- summary(estims2_Q2, conf.int = TRUE)[[7]][2]
  upperbound <- summary(estims2_Q2, conf.int = TRUE)[[8]][2]
  # Update the variable `times2`
  if (beta1 >= lowerbound & beta1 <= upperbound){
    times2 <- times2 + 1
  }
}

# Display the empirical coverage probabilities of the 95% confidence intervals
# for beta1 under the two imputation approaches
times1 / n
```

```
## [1] 0.88
```

```
times2 / n
```

```
## [1] 0.95
```

In this question, in order to explore the effect that acknowledging or not acknowledging parameter uncertainty when performing step 1 of multiple imputation (MI) might have on the coverage of the corresponding confidence intervals, I have respectively performed the improper multiple imputation with the stochastic regression imputation (SRI) method and the proper multiple imputation with the bootstrap based stochastic regression imputation method on the given 100 data sets.

From the results, we can see that when acknowledging the parameter uncertainty, i.e., performing the proper MI using the bootstrap based SRI, the empirical coverage probability of the 95% confidence intervals for β_1 is just 0.95, which means we repeat the experiment 100 times and the true value of β_1 falls in the confidence interval just 95 times. It coincides with the meaning of the 95% confidence interval and is a very sensible result. However, when we perform the improper MI using SRI, the parameter uncertainty is not acknowledged since the same estimates of the parameters are used for imputing all M copies of the data set. In this case, the empirical coverage probability of the 95% confidence intervals for β_1 is only 0.88, which is lower than the expected probability 0.95. It is because the improper MI approach may lead to confidence intervals that are too narrow and so it will reduce the coverage probability of the confidence intervals.

Question 3. Solutions.

Without loss of generality, we can assume that the linear (in the coefficients) regression model is given by

$$Y_i = \beta_1 f_1(x_{i,1}, x_{i,2}, \ldots, x_{i,p}) + \beta_2 f_2(x_{i,1}, x_{i,2}, \ldots, x_{i,p}) + \cdots + \beta_k f_k(x_{i,1}, x_{i,2}, \ldots, x_{i,p}) + \varepsilon_i,$$

where $x_{i,j}$ $(i = 1, 2, \ldots, n)$ are the values of the covariate variables X_j $(j = 1, 2, \ldots, p)$ and Y_i $(i = 1, 2, \ldots, n)$ are the response variables, and $\beta_1, \beta_2, \ldots, \beta_k$ are the coefficients of the regression model, and $f_1, f_2, \ldots, f_k$ are the functions of the values of the covariate variables $X_1, X_2, \ldots, X_p$, and ε_i $(i = 1, 2, \ldots, n)$ are uncorrelated random variables with $E(\varepsilon_i) = 0$ and $Var(\varepsilon_i) = \sigma^2$ where σ is a constant, and $k, p, n \in N^+$ are constants.

For the strategy (i), it first computes the predicted values from each fitted model in step 2:

$$\widehat{y}_i^{(m)} = \widehat{\beta}_1^{(m)} f_1(x_{i,1}, x_{i,2}, \ldots, x_{i,p}) + \widehat{\beta}_2^{(m)} f_2(x_{i,1}, x_{i,2}, \ldots, x_{i,p}) + \cdots + \widehat{\beta}_k^{(m)} f_k(x_{i,1}, x_{i,2}, \ldots, x_{i,p}),$$

where $\widehat{\beta}_s^{(m)}$ is the estimate of β_s obtained from the m-th complete data set after imputation $(s = 1, 2, \ldots, k;\ m = 1, 2, \ldots, M)$ and M is the number of the complete data set, and $i = 1, 2 \ldots, n$.

Then, it pools them according to Rubin's rule for point estimates, i.e., averaging the predicted values across the imputed data sets. Thus, the final predicted values after pooling are

$$\begin{aligned} \widehat{y}_i &= \frac{1}{M} \sum_{m=1}^{M} \widehat{y}_i^{(m)}, \\ &= \frac{1}{M} \sum_{m=1}^{M} \widehat{\beta}_1^{(m)} f_1 + \frac{1}{M} \sum_{m=1}^{M} \widehat{\beta}_2^{(m)} f_2 + \cdots + \frac{1}{M} \sum_{m=1}^{M} \widehat{\beta}_k^{(m)} f_k, \\ &= \overline{\widehat{\beta}}_1 f_1 + \overline{\widehat{\beta}}_2 f_2 + \cdots + \overline{\widehat{\beta}}_k f_k, \qquad (1) \end{aligned}$$

where $\overline{\widehat{\beta}}_s = \frac{1}{M} \sum_{m=1}^{M} \widehat{\beta}_s^{(m)}$ is the average of the estimates $\widehat{\beta}_s^{(1)}, \widehat{\beta}_s^{(2)}, \ldots, \widehat{\beta}_s^{(m)}$ across the imputed data sets $(s = 1, 2, \ldots, k)$, and $f_s = f_s(x_{i,1}, x_{i,2}, \ldots, x_{i,p})$ $(s = 1, 2, \ldots, k)$, and $i = 1, 2 \ldots, n$.

Next, for the strategy (ii), it first pools the regression coefficients from each fitted model in step 2 using Rubin's rule for point estimates. Therefore, after pooling, the final estimated regression coefficients $\widehat{\beta}_1^*, \widehat{\beta}_2^*, \ldots, \widehat{\beta}_k^*$ are

$$\begin{aligned} \widehat{\beta}_1^* &= \frac{1}{M} \sum_{m=1}^{M} \widehat{\beta}_1^{(m)} = \overline{\widehat{\beta}}_1, \\ \widehat{\beta}_2^* &= \frac{1}{M} \sum_{m=1}^{M} \widehat{\beta}_2^{(m)} = \overline{\widehat{\beta}}_2, \\ &\cdots\cdots \\ \widehat{\beta}_k^* &= \frac{1}{M} \sum_{m=1}^{M} \widehat{\beta}_k^{(m)} = \overline{\widehat{\beta}}_k, \end{aligned}$$

where the meanings of $\widehat{\beta}_s^{(m)}$ and $\overline{\widehat{\beta}}_s$ $(m = 1, 2, \ldots, M;\ s = 1, 2, \ldots, k)$ are the same as above.

Then, it computes the predicted values using the pooled coefficients $\widehat{\beta}_1^*, \widehat{\beta}_2^*, \ldots, \widehat{\beta}_k^*$. Thus, the final predicted values are

$$\widehat{y}_i = \widehat{\beta}_1^* f_1 + \widehat{\beta}_2^* f_2 + \cdots + \widehat{\beta}_k^* f_k = \overline{\widehat{\beta}}_1 f_1 + \overline{\widehat{\beta}}_2 f_2 + \cdots + \overline{\widehat{\beta}}_k f_k, \qquad (2)$$

where the meanings of $\overline{\widehat{\beta}}_s$ and f_s $(s = 1, 2, \ldots, k)$ are also the same as above, and $i = 1, 2 \ldots, n$.

We can see that the formula (1) and formula (2) which are the expressions of the final predicted values derived by these two strategies are the same. Hence, the strategies (i) and (ii) coincide.

Question 4. Solutions.

Firstly, I load the data set for this question and check the dimension of it.

```r
load("dataex4.Rdata")
dim(dataex4)
```

```
## [1] 1000    3
```

(a) The R code for this question is shown below.

```r
# Only impute the variables `y` and `x1` in step 1
impu1_Q4 <- mice(dataex4, m = 50, seed = 1, printFlag = FALSE)
# Estimate the regression coefficients of the model of interest in step 2
fit1_Q4 <- with(impu1_Q4, lm(y ~ x1 + x2 + x1*x2))
# Pool the results in step 3
estims1_Q4 <- pool(fit1_Q4)
# Display the outcome of the parameter estimation
summary(estims1_Q4, conf.int = TRUE)
```

```
##          term  estimate  std.error statistic       df p.value    2.5 %
## 1 (Intercept) 1.5929831 0.09541331  16.69561 154.5617       0 1.404501
## 2          x1 1.4112333 0.09732912  14.49960 216.0125       0 1.219397
## 3          x2 1.9658191 0.05323220  36.92913 153.5344       0 1.860657
## 4       x1:x2 0.7550367 0.05701458  13.24287 138.0530       0 0.642302
##      97.5 %
## 1 1.7814655
## 2 1.6030697
## 3 2.0709812
## 4 0.8677715
```

By only imputing the variables `y` and `x1` in step 1, the estimates of β_1, β_2 and β_3 are 1.4112333, 1.9658191 and 0.7550367 respectively, and the 95% confidence intervals for β_1, β_2 and β_3 are [1.219397, 1.6030697], [1.860657, 2.0709812] and [0.642302, 0.8677715] respectively. Only the estimate of β_2 is relatively accurate and the true value of β_2 falls into the estimated 95% confidence interval of it. However, the estimate of β_1 is significantly larger and the estimate of β_3 is significantly smaller and they all fall outside their corresponding estimated 95% confidence intervals. In the imputation process, the interaction variable is left outside and the variables `y` and `x1` are imputed only by other individual variables. Therefore, the imputation model in step 1 deviates from the original model used to generate the data, and so the imputation process would not be so accurate and may contain some errors. Consequently, there may be some deviations between the final results and the real values.

(b) The R code for this question is shown in the following.

```r
# Calculate the intersection variable and append it as a variable to the new data set
# The new variable is named `x1x2`
data_Q4 <- cbind(dataex4, "x1x2" = dataex4$x1 * dataex4$x2)

# Do the setup run of `mice()` function
impu0_Q4 <- mice(data_Q4, maxit = 0)
# Extract and modify the imputation methods
meth_Q4 <- impu0_Q4$method
```

```r
# Use passive imputation to impute the intersection variable `x1x2`
meth_Q4["x1x2"] <- "~I(x1*x2)"
# Modify the predictor matrix to prevent feedback from the
# intersection variable `x1x2` in the imputation of `x1` and `x2`
pred_Q4 <- impu0_Q4$predictorMatrix
pred_Q4[c("x1", "x2"), "x1x2"] <- 0

# Perform multiple imputation on the new data set as the requirements of this question
impu2_Q4 <- mice(data_Q4, method = meth_Q4, predictorMatrix = pred_Q4,
                 m = 50, seed = 1, printFlag = FALSE)
fit2_Q4 <- with(impu2_Q4, lm(y ~ x1 + x2 + x1*x2))
estims2_Q4 <- pool(fit2_Q4)
summary(estims2_Q4, conf.int = TRUE)
```

```
##          term  estimate  std.error statistic       df p.value     2.5 %
## 1 (Intercept) 1.5534782 0.08842211  17.56889 161.1274       0 1.3788626
## 2          x1 1.1926170 0.09584345  12.44339 180.3188       0 1.0034980
## 3          x2 1.9964402 0.04936582  40.44175 159.8398       0 1.8989468
## 4       x1:x2 0.8740573 0.05678521  15.39234 114.4704       0 0.7615712
##      97.5 %
## 1 1.7280939
## 2 1.3817360
## 3 2.0939336
## 4 0.9865434
```

The estimates of β_1, β_2 and β_3 are 1.1926170, 1.9964402 and 0.8740573 respectively, and the 95% confidence intervals for β_1, β_2 and β_3 are [1.0034980, 1.3817360], [1.8989468, 2.0939336] and [0.7615712, 0.9865434] respectively. Compared with only imputing the variables `y` and `x1` in step 1 as done in question (a), the results are much more accurate when the passive imputation is used to impute the intersection variable in the imputation process. We can see that the estimate of β_2 is extremely close to its true value and the estimated 95% confidence interval of it is slightly narrower than the last question. Although the true values of β_1 and β_3 are still outside their estimated 95% confidence intervals, the estimates of β_1 and β_3 are obviously more accurate than question (a) and they are very close to the bounds of their confidence intervals.

(c) The R code for this question is shown as follows.

```r
# Perform the multiple imputation and impute the intersection variable
# `x1x2` as it is just another variable
impu3_Q4 <- mice(data_Q4, m = 50, seed = 1, printFlag = FALSE)
# Directly use the variable named `x1x2` for the intersection term in step 2
fit3_Q4 <- with(impu3_Q4, lm(y ~ x1 + x2 + x1x2))
estims3_Q4 <- pool(fit3_Q4)
summary(estims3_Q4, conf.int = TRUE)
```

```
##          term estimate  std.error statistic       df p.value     2.5 %   97.5 %
## 1 (Intercept) 1.499714 0.07821436  19.17441 153.8212       0 1.3452011 1.654227
## 2          x1 1.003930 0.08228372  12.20083 169.4517       0 0.8414967 1.166363
## 3          x2 2.026180 0.04371605  46.34864 152.2588       0 1.9398113 2.112548
## 4        x1x2 1.017793 0.04428071  22.98501 161.2567       0 0.9303479 1.105238
```

The estimates of β_1, β_2 and β_3 are 1.003930, 2.026180 and 1.017793 respectively, and the 95% confidence intervals for β_1, β_2 and β_3 are [0.8414967, 1.166363], [1.9398113, 2.112548] and [0.9303479, 1.105238] respectively. When we treat the intersection variable `x1x2` as just another variable, i.e., an extra variable named `x1x2`, the final results are much more accurate than the previous two questions. The estimates of β_1, β_2 and β_3 are very close to their true values and the true values of β_1, β_2 and β_3 are just located in their 95% confidence intervals. Also, the confidence intervals in this question are further slightly narrower than question (a) and (b). Consequently, the *just another variable* approach performs very well in estimating the parameters of the substantive model this time.

(d) Although the *just another variable* approach for imputing interactions performs very well in this question, it also has some conceptual drawbacks. The obvious conceptual drawback of this approach is that the relationship between the intersection variable `x1x2` and the individual variables `x1` and `x2` is treated as stochastic relation instead of deterministic relation. However, the intersection variable `x1x2` is a deterministic function of `x1` and `x2`, whose value is exactly determined by `x1` and `x2`. Therefore, if we still impute `x1x2` from other variables directly, it will violate the real relationship between `x1x2` and other variables and the values of `x1x2` may be inconsistent with the values or imputed values of `x1` and `x2`. Hence, the precision of the results of analysis may be consequently reduced.

Question 5. Solutions.

Let me first start by loading and inspecting the data set.

```r
load("NHANES2.Rdata")
dim(NHANES2)
```

```
## [1] 500  12
```

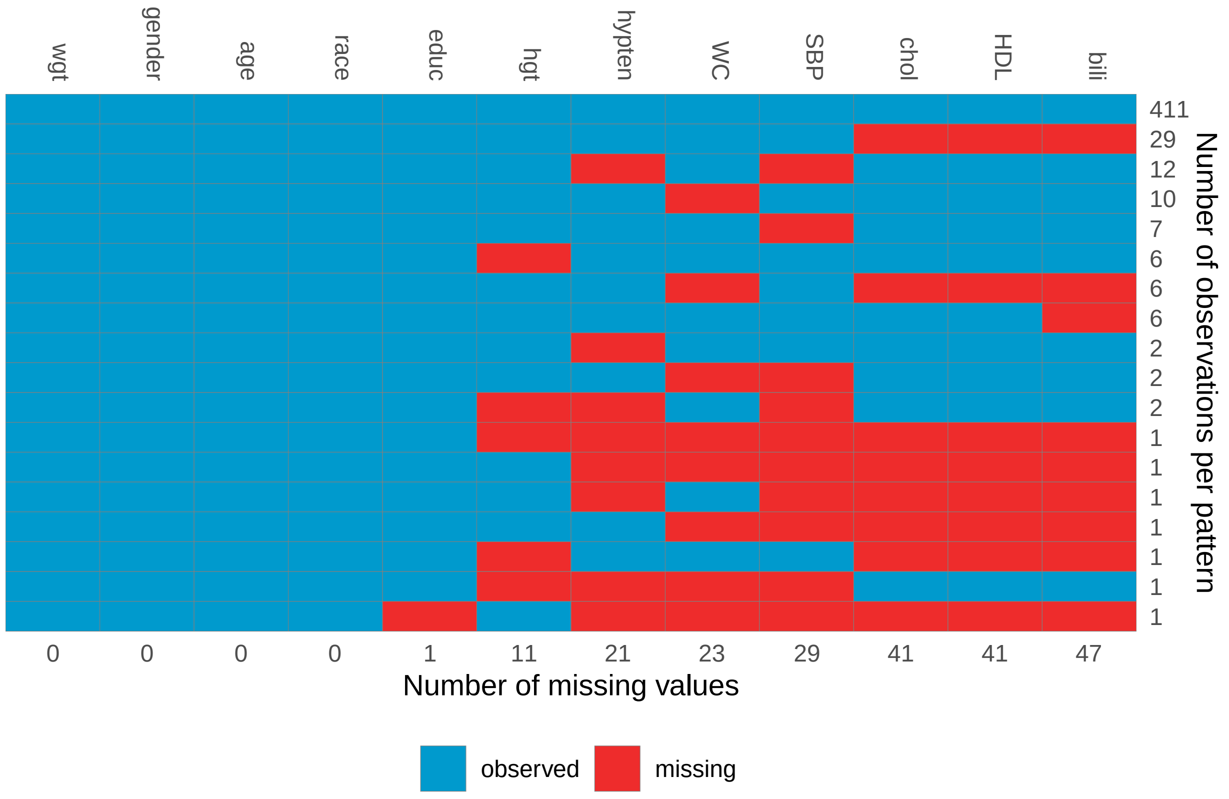
```r
summary(NHANES2)
```

```
##       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
```

This data set has 12 variables and 500 observations. The variables `bili`, `chol`, `HDL`, `hgt`, `educ`, `SBP`, `hypten` and `WC` are subject to missingness but the proportions of missing values of these variables are not large which are all less than 10%.

Let me now further inspect the missing data patterns.

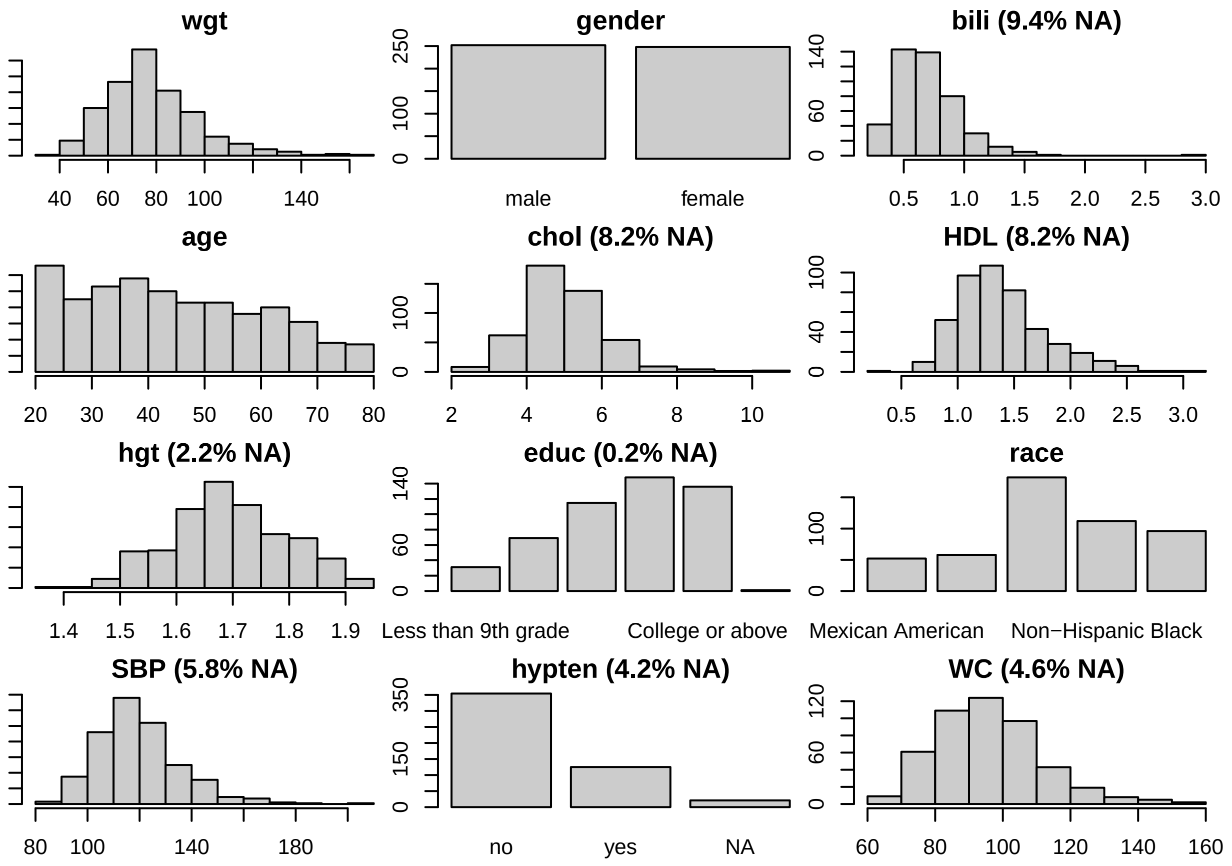
```r
pattern <- md_pattern(NHANES2, pattern = TRUE, color = c("deepskyblue3", 'firebrick2'))
pattern$plot
```



It can be seen from the above plot that there are 411 observations with no missing values in any of the variables, and some observations have missing values in only one variable but the others are subject to missingness in several variables. Overall, the rate of missingness in this data set is relatively low.

Let me now visualize how the distributions of the observed values in different variables look like.

```r
par(mar = c(2, 1, 2, 1), mgp = c(3, 1, 0))
plot_all(NHANES2)
```



We can see from the above figure that the distributions of the continuous variables, except `hgt`, are somewhat skewed, and so we could choose predictive mean matching to impute the missing values.

I will now proceed to the imputation step. We do not have any variables that can be written as a deterministic function of other variables in the data set, and furthermore from the information available there is not any reason to change the `predictorMatrix`. The variables that are not in our substantive model, act here as auxiliary variables which typically also improve the plausibility of the missing at random assumption. Consequently, there is no need to perform the setup run of `mice()`, so I proceed to step one of multiple imputation directly. For the final imputation, I will use `maxit = 30` and `M = 20`.

```
impu_Q5 <- mice(NHANES2, maxit = 30, m = 20, seed = 1, printFlag = FALSE)
```

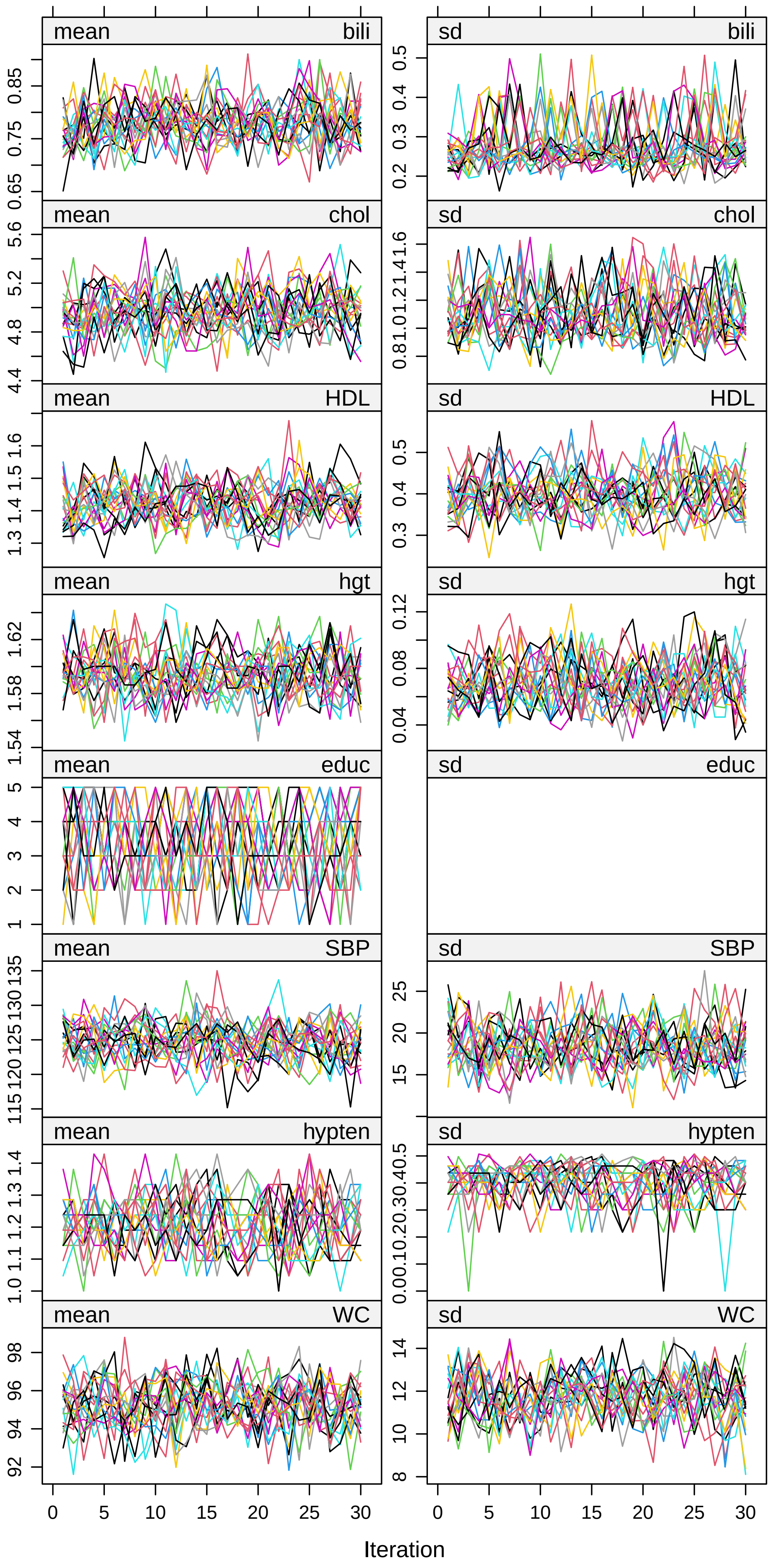
Let me check if `mice()` found any problem during the imputation.

```
impu_Q5$loggedEvents
```

```
## NULL
```

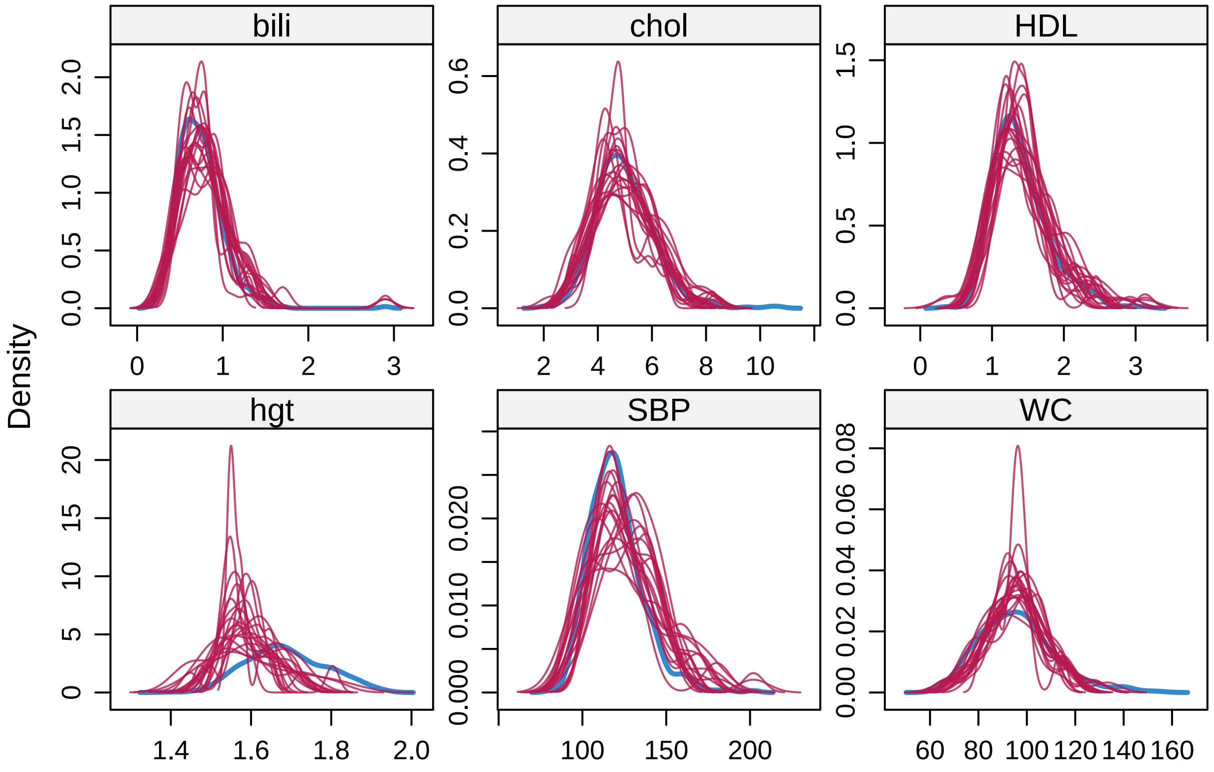
Let me now look at the chains of the imputed values to check whether there are convergence problems. The figure below indicates good convergence of the chains.

```
plot(impu_Q5, layout = c(2, 8))
```

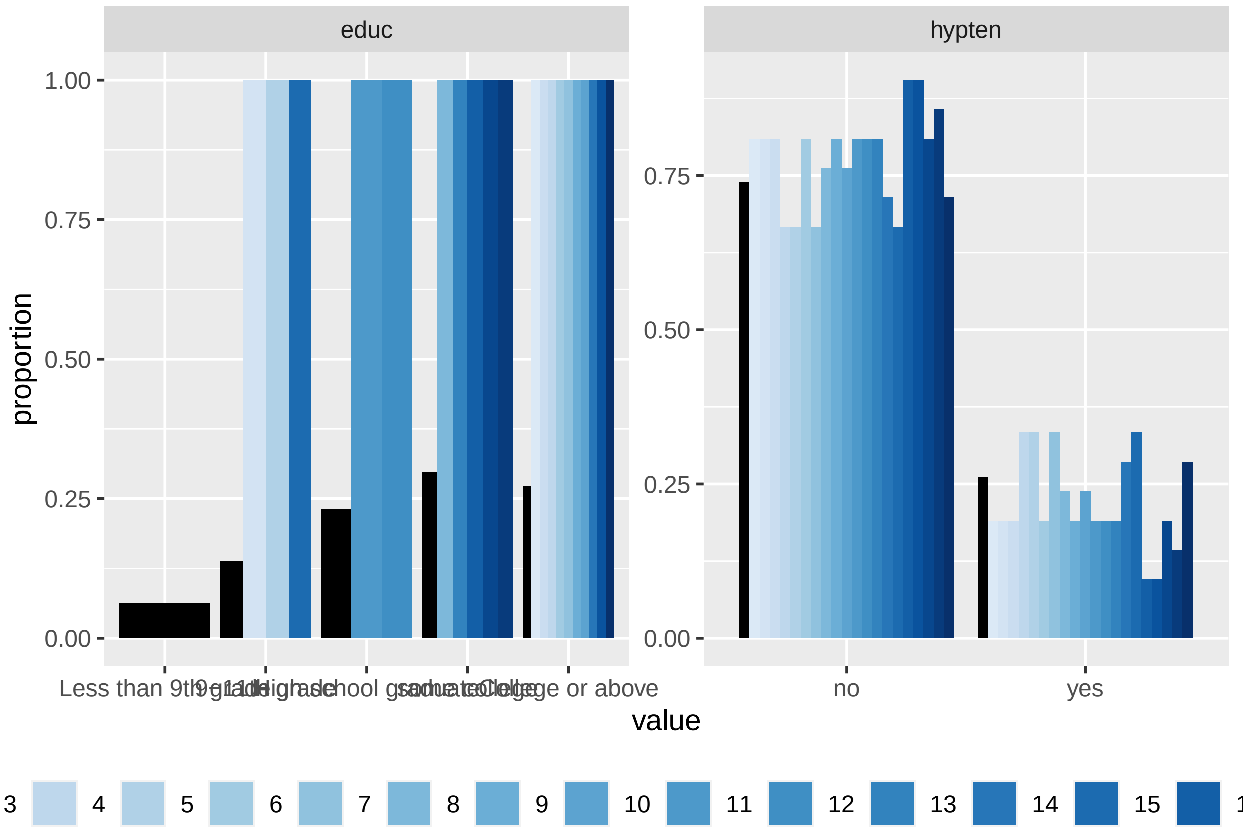

mean bili
sd bili
mean chol
sd chol
mean HDL
sd HDL
mean hgt
sd hgt
mean educ
sd educ
mean SBP
sd SBP
mean hypten
sd hypten
mean WC
sd WC
Iteration

Then, let me inspect if the distribution of the imputed values agrees with the distribution of the observed values.

```r
densityplot(impu_Q5)
```



```r
source("https://gist.githubusercontent.com/NErler/0d00375da460dd33839b98faeee2fdab/raw/c6f53
propplot(impu_Q5)
```



Although there are significant discrepancies between the observed and imputed values for educational status, we do not need to worry about it because this variable has only 1 missing values. For the variable `hgt`, there are also some discrepancies between the observed and imputed values, but this variable only has 11 missing values, so it is also not too problematic. In addition, everything looks reasonable.

After having confirmed that our imputation step is successful, we can proceed to the analysis of the imputed data and fit our substantive model of interest.

```
fit_Q5 <- with(impu_Q5, lm(wgt ~ gender + age + hgt + WC))
```

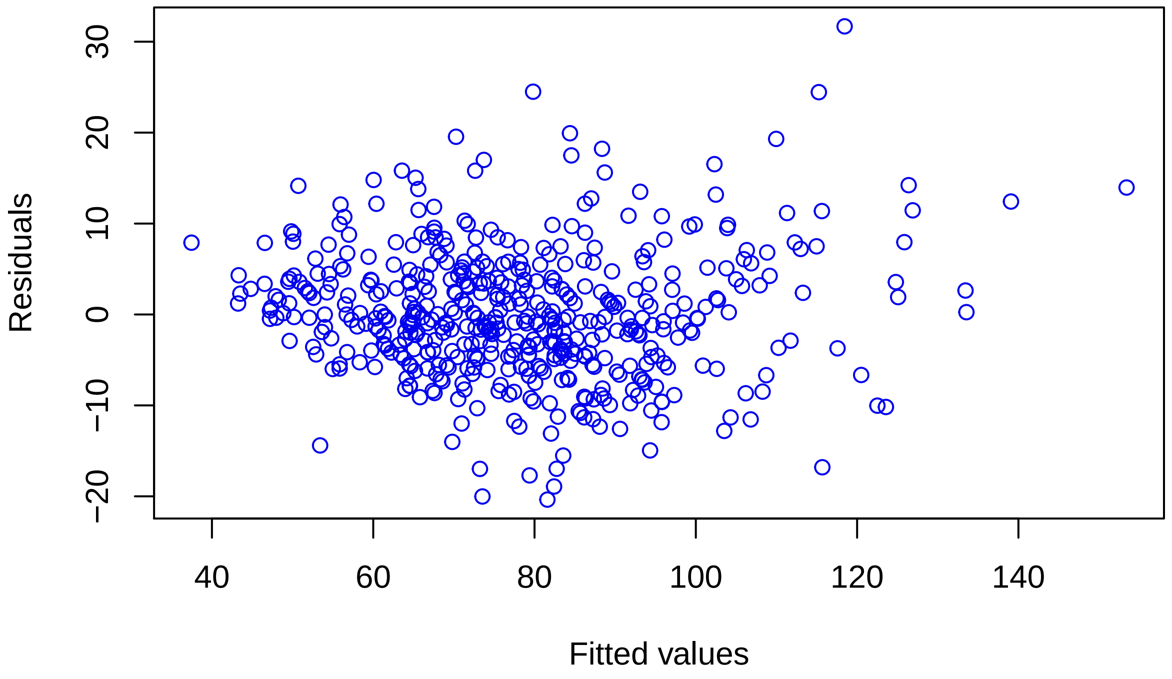
We can further explore the information contained in the object `fit_Q5`. For example, we can look at the summary of the fitted model in the first imputed data set.

```
summary(fit_Q5$analyses[[1]])
```

```
##
## Call:
## lm(formula = wgt ~ gender + age + hgt + WC)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -20.360  -4.493  -0.443   3.881  31.688
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -101.08672    7.36849 -13.719  < 2e-16 ***
## genderfemale    -1.34922    0.81018  -1.665   0.0965 .
## age            -0.16160    0.02063  -7.834 2.91e-14 ***
## hgt            52.46664    4.21911  12.435  < 2e-16 ***
## WC              1.02934    0.02191  46.970  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.102 on 495 degrees of freedom
## Multiple R-squared:  0.8605, Adjusted R-squared:  0.8594
## F-statistic: 763.5 on 4 and 495 DF,  p-value: < 2.2e-16
```

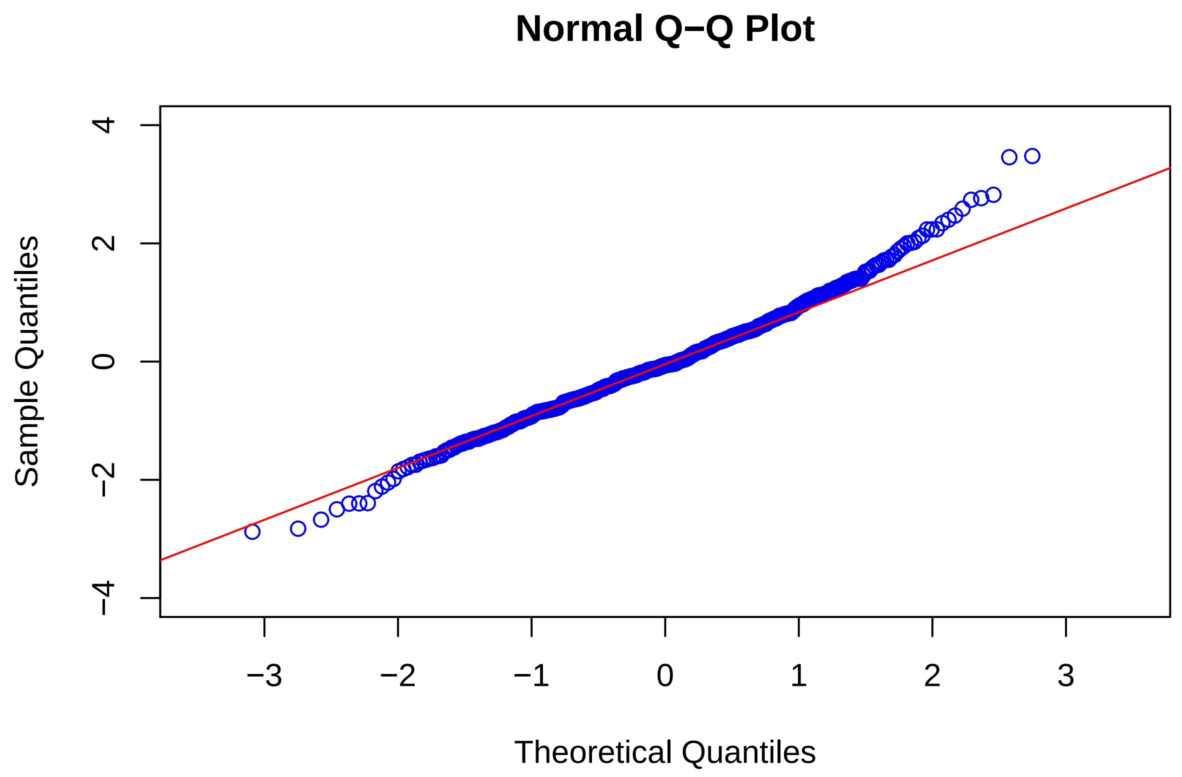
Then, let us check the validity of model's assumptions, i.e., performing diagnostic checks. To begin with, we can look at the plot of the residuals versus fitted values to check the assumptions of linearity and homoscedasticity.

```
plot(fit_Q5$analyses[[1]]$fitted.values, residuals(fit_Q5$analyses[[1]]),
xlab = "Fitted values", ylab = "Residuals", col = "blue2")
```



On the other hand, we can also check the QQ-plot.

```
qqnorm(rstandard(fit_Q5$analyses[[1]]), col = "blue2", xlim = c(-3.5, 3.5), ylim = c(-4, 4))
qqline(rstandard(fit_Q5$analyses[[1]]), col = "red")
```



From these figures, we can find nothing that looks suspicious, so now we can proceed to the step 3, pooling the results.

Finally, I am going to pool the results.

```
# Pool the results and display the summary of analysis
estims_Q5 <- pool(fit_Q5)
estims_Q5
```

```
## Class: mipo    m = 20
##            term  m      estimate         ubar            b            t dfcom
## 1   (Intercept) 20 -101.7202452 5.560931e+01 1.901353e+00 5.760573e+01   495
## 2 genderfemale 20   -1.2998960 6.724679e-01 1.050898e-02 6.835024e-01   495
## 3           age 20   -0.1594593 4.352132e-04 1.566635e-05 4.516628e-04   495
## 4           hgt 20   52.9480781 1.816420e+01 6.352469e-01 1.883121e+01   495
## 5            WC 20    1.0264179 4.889697e-04 1.390506e-05 5.035700e-04   495
##         df        riv     lambda        fmi
## 1 462.0256 0.03590082 0.03465662 0.03880841
## 2 481.8469 0.01640886 0.01614396 0.02020238
## 3 459.8070 0.03779681 0.03642024 0.04058431
## 4 461.0711 0.03672109 0.03542042 0.03957745
## 5 468.7888 0.02985934 0.02899361 0.03310988
```

```
summary_Q5 <- summary(estims_Q5, conf.int = TRUE)
summary_Q5
```

```
##            term     estimate  std.error  statistic       df      p.value
## 1   (Intercept) -101.7202452 7.58984411 -13.402152 462.0256 0.000000e+00
## 2 genderfemale   -1.2998960 0.82674202  -1.572312 481.8469 1.165347e-01
## 3           age   -0.1594593 0.02125236  -7.503135 459.8070 3.250733e-13
## 4          hgt   52.9480781 4.33949408  12.201440 461.0711 0.000000e+00
## 5           WC    1.0264179 0.02244037  45.739804 468.7888 0.000000e+00
##          2.5 %      97.5 %
## 1 -116.6351369 -86.8053535
## 2   -2.9243609   0.3245690
## 3   -0.2012231  -0.1176955
## 4   44.4204410  61.4757152
## 5    0.9823218   1.0705141
```

```r
# Show the estimates of the parameters of the regression model in a table
df <- data.frame("Estimate" = summary_Q5[, 2],
                 "Lq" = summary_Q5[, 7], "Uq" = summary_Q5[, 8])
rownames(df) <- c("$\\beta_0$", "$\\beta_1$", "$\\beta_2$", "$\\beta_3$", "$\\beta_4$")
colnames(df) <- c("Estimate", "2.5% quantile", "97.5% quantile")
kable(df, escape = FALSE, digits = 4, caption =
      "Regression coefficient estimates and corresponding 95% confidence intervals")
```

Table 1: Regression coefficient estimates and corresponding 95% confidence intervals

	Estimate	2.5% quantile	97.5% quantile
β_0	-101.7202	-116.6351	-86.8054
β_1	-1.2999	-2.9244	0.3246
β_2	-0.1595	-0.2012	-0.1177
β_3	52.9481	44.4204	61.4757
β_4	1.0264	0.9823	1.0705

The (pooled) regression coefficient estimates of β_0, β_1, β_2, β_3 and β_4, and the corresponding 95% confidence intervals for these parameters are shown in Table 1. In addition, what is worth being noticed is that the value of M should be increased if the results change by a large extent as different random seeds are tried.