# Incomplete Data Analysis
# Assignment 3

Josephine Li s2346729

## Question 1

### Sub-Question (a)

```r
# load the package and data
require(mice)
```

```
## Loading required package: mice

##
## Attaching package: 'mice'

## The following object is masked from 'package:stats':
##
##     filter

## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```r
NHANES <- nhanes
# checking the percentage of incomplete cases
# use function ic in mice package to count incomplete cases
incomplete.perc <- nrow(ic(NHANES))/(nrow(NHANES))
incomplete.perc
```

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

The percentage of incomplete cases is 48%.

### Sub-Question (b)

```r
# step 1, use default parameters, set seed = 1
imp <-  mice(NHANES, printFlag = FALSE,seed = 1)

# step 2, predict bmi from age, hyp, and chl
```

```r
fit <- with(imp, lm(bmi ~ age + hyp + chl))

# step 3, pool the result
m5.s1 <- pool(fit)

# obtain the proportions of variance due to the missing data
m5.s1$pooled[,c("term","lambda")]
```

```
##          term     lambda
## 1 (Intercept) 0.08938989
## 2         age 0.68640637
## 3         hyp 0.35043452
## 4         chl 0.30408063
```

By check the `lambda` column of our result from `pool()` function, we obtain `lambda` values for each parameters, which represent the proportions of variance due to the missing data. The proportions of variance due to the missing data for `age` is about 0.686, for `hyp` is about 0.350 and for `chl` is about 0.304. `Age` appears to be the most affected by the non-response.

### Sub-Question (c)

```r
# repeat for seed = 2
m5.s2 <- pool(with(mice(NHANES, printFlag = FALSE,seed = 2),
                   lm(bmi ~ age + hyp + chl)))
m5.s2$pooled[,c("term","lambda")]
```

```
##          term    lambda
## 1 (Intercept) 0.4144454
## 2         age 0.4033924
## 3         hyp 0.1430995
## 4         chl 0.2959966
```

From the repeat results, we can find that, when seed = 2, the proportions of variance due to the missing data for `age` is about 0.403, for `hyp` is about 0.143 and for `chl` is about 0.296. `Age` appears to be the most affected by the non-response.

```r
# repeat for seed = 3
m5.s3 <- pool(with(mice(NHANES, printFlag = FALSE,seed = 3),
                   lm(bmi ~ age + hyp + chl)))
m5.s3$pooled[,c("term","lambda")]
```

```
##          term    lambda
## 1 (Intercept) 0.2772900
## 2         age 0.5895051
## 3         hyp 0.4101152
## 4         chl 0.5621346
```

From the repeat results, we can find that, when seed = 3, the proportions of variance due to the missing data for `age` is about 0.590, for `hyp` is about 0.410 and for `chl` is about 0.562. `Age` appears to be the most affected by the non-response.

```r
# repeat for seed = 4
m5.s4 <- pool(with(mice(NHANES, printFlag = FALSE,seed = 4),
                   lm(bmi ~ age + hyp + chl)))
m5.s4$pooled[,c("term","lambda")]
```

```
##          term    lambda
## 1 (Intercept) 0.1315114
## 2         age 0.2189333
## 3         hyp 0.1961083
## 4         chl 0.3305334
```

From the repeat results, we can find that, when seed = 4, the proportions of variance due to the missing data for `age` is about 0.219, for `hyp` is about 0.196 and for `chl` is about 0.331. `Chl` appears to be the most affected by the non-response.

```r
# repeat for seed = 5
m5.s5 <- pool(with(mice(NHANES, printFlag = FALSE,seed = 5),
                   lm(bmi ~ age + hyp + chl)))
m5.s5$pooled[,c("term","lambda")]
```

```
##          term    lambda
## 1 (Intercept) 0.4855733
## 2         age 0.4511896
## 3         hyp 0.5942866
## 4         chl 0.2346065
```

From the repeat results, we can find that, when seed = 5, the proportions of variance due to the missing data for `age` is about 0.451, for `hyp` is about 0.594 and for `chl` is about 0.235. `hyp` appears to be the most affected by the non-response.

```r
# repeat for seed = 6
m5.s6 <- pool(with(mice(NHANES, printFlag = FALSE,seed = 6),
                   lm(bmi ~ age + hyp + chl)))
m5.s6$pooled[,c("term","lambda")]
```

```
##          term    lambda
## 1 (Intercept) 0.4168136
## 2         age 0.6549523
## 3         hyp 0.2960364
## 4         chl 0.5196295
```

From the repeat results, we can find that, when seed = 6, the proportions of variance due to the missing data for `age` is about 0.655, for `hyp` is about 0.296 and for `chl` is about 0.520. `age` appears to be the most affected by the non-response.

In conclusion, for seed equal to $2, 3, 6$, age is still the most affected parameter by the non-response. When seed equal to 4, `chl` appears to be the most affected by the non-response and when seed equal to 5, `hyp` appears to be the most affected by the non-response.

**Sub-Question (d)**

First, we run repeat the process for seed equal to 1 to 6, while set $m = 100$ rather than 5.

```r
# change m to m=100, repeat for same seeds
m100.s1 <- pool(with(mice(NHANES, printFlag = F,seed = 1,m=100),
                   lm(bmi ~ age + hyp + chl)))
m100.s2 <- pool(with(mice(NHANES, printFlag = F,seed = 2,m=100),
                   lm(bmi ~ age + hyp + chl)))
m100.s3 <- pool(with(mice(NHANES, printFlag = F,seed = 3,m=100),
                   lm(bmi ~ age + hyp + chl)))
m100.s4 <- pool(with(mice(NHANES, printFlag = F,seed = 4,m=100),
                   lm(bmi ~ age + hyp + chl)))
m100.s5 <- pool(with(mice(NHANES, printFlag = F,seed = 5,m=100),
                   lm(bmi ~ age + hyp + chl)))
m100.s6 <- pool(with(mice(NHANES, printFlag = F,seed = 6,m=100),
                   lm(bmi ~ age + hyp + chl)))
```

Compare the result when using M=100 and M=5.

```r
# seed = 1 compare M = 100 and M = 5
knitr::kable(data.frame(
  m100 <- m100.s1$pooled[2:4,c("term","estimate","lambda")],
  m5 <- m5.s1$pooled[2:4,c("term","estimate","lambda")]),
  col.names = c("term","m100.estimator","m100.lambda",
              "term","m5.estimator","m5.lambda"),
  main = "seed = 1")
```

| | term | m100.estimator | m100.lambda | term | m5.estimator | m5.lambda |
|---|---|---|---|---|---|---|
| 2 | age | -3.6412551 | 0.4324680 | age | -3.5528716 | 0.6864064 |
| 3 | hyp | 1.6857944 | 0.2915346 | hyp | 2.1970175 | 0.3504345 |
| 4 | chl | 0.0544971 | 0.3217837 | chl | 0.0537808 | 0.3040806 |

```r
# seed = 2 compare M = 100 and M = 5
knitr::kable(data.frame(
  m100 <- m100.s2$pooled[2:4,c("term","estimate","lambda")],
  m5 <- m5.s2$pooled[2:4,c("term","estimate","lambda")]),
  col.names = c("term","m100.estimator","m100.lambda",
              "term","m5.estimator","m5.lambda"),
  main = "seed = 2")
```

| | term | m100.estimator | m100.lambda | term | m5.estimator | m5.lambda |
|---|---|---|---|---|---|---|
| 2 | age | -3.6337887 | 0.4031077 | age | -4.0615093 | 0.4033924 |
| 3 | hyp | 1.7199145 | 0.2825108 | hyp | 1.5304762 | 0.1430995 |
| 4 | chl | 0.0535212 | 0.2939693 | chl | 0.0628349 | 0.2959966 |

```r
# seed = 3 compare M = 100 and M = 5
knitr::kable(data.frame(
  m100 <- m100.s3$pooled[2:4,c("term","estimate","lambda")],
  m5 <- m5.s3$pooled[2:4,c("term","estimate","lambda")]),
  col.names = c("term","m100.estimator","m100.lambda",
                "term","m5.estimator","m5.lambda"),
  main = "seed = 3")
```

| | term | m100.estimator | m100.lambda | term | m5.estimator | m5.lambda |
|---|---|---|---|---|---|---|
| 2 | age | -3.5570609 | 0.3093072 | age | -3.8575334 | 0.5895051 |
| 3 | hyp | 1.5575621 | 0.2425105 | hyp | 1.3528124 | 0.4101152 |
| 4 | chl | 0.0544541 | 0.3281911 | chl | 0.0587283 | 0.5621346 |

```r
# seed = 4 compare M = 100 and M = 5
knitr::kable(data.frame(
  m100 <- m100.s4$pooled[2:4,c("term","estimate","lambda")],
  m5 <- m5.s4$pooled[2:4,c("term","estimate","lambda")]),
  col.names = c("term","m100.estimator","m100.lambda",
                "term","m5.estimator","m5.lambda"),
  main = "seed = 4")
```

| | term | m100.estimator | m100.lambda | term | m5.estimator | m5.lambda |
|---|---|---|---|---|---|---|
| 2 | age | -3.6481533 | 0.3943223 | age | -3.5060335 | 0.2189333 |
| 3 | hyp | 1.6594695 | 0.2565132 | hyp | 2.7505305 | 0.1961083 |
| 4 | chl | 0.0551159 | 0.2835232 | chl | 0.0492061 | 0.3305334 |

```r
# seed = 5 compare M = 100 and M = 5
knitr::kable(data.frame(
  m100 <- m100.s5$pooled[2:4,c("term","estimate","lambda")],
  m5 <- m5.s5$pooled[2:4,c("term","estimate","lambda")]),
  col.names = c("term","m100.estimator","m100.lambda",
                "term","m5.estimator","m5.lambda"),
  main = "seed = 5")
```

| | term | m100.estimator | m100.lambda | term | m5.estimator | m5.lambda |
|---|---|---|---|---|---|---|
| 2 | age | -3.7629724 | 0.3322570 | age | -3.4967225 | 0.4511896 |
| 3 | hyp | 1.8028317 | 0.2893046 | hyp | 1.5095477 | 0.5942866 |
| 4 | chl | 0.0553438 | 0.2461956 | chl | 0.0608127 | 0.2346065 |

```r
# seed = 6 compare M = 100 and M = 5
knitr::kable(data.frame(
  m100 <- m100.s6$pooled[2:4,c("term","estimate","lambda")],
  m5 <- m5.s6$pooled[2:4,c("term","estimate","lambda")]),
  col.names = c("term","m100.estimator","m100.lambda",
                "term","m5.estimator","m5.lambda"),
  main = "seed = 6")
```

|   | term | m100.estimator | m100.lambda | term | m5.estimator | m5.lambda |
|---|---|---|---|---|---|---|
| 2 | age | -3.5930919 | 0.4430300 | age | -2.9214135 | 0.6549523 |
| 3 | hyp | 1.8621963 | 0.2860700 | hyp | 1.2247460 | 0.2960364 |
| 4 | chl | 0.0531932 | 0.3113085 | chl | 0.0494922 | 0.5196295 |

Based on the results, different seeds can lead to changes in estimated coefficients and variance estimates, but for each seed, the estimates obtained with m=100 and m=5 are very similar. Therefore, in this case, choosing m=5 or m=100 may not have a significant impact on the results.

However, if we want more accurate estimates and standard errors, you can choose m=100 because this value is closer to the magnitude of the population and may produce more accurate estimates. Additionally, if you have sufficient computational resources, choosing a larger m value may be even better.

## Question 2

The goal of this question is compare the coverage of the confidence intervals if we acknowledge parameters uncertainty in step 1 or not.

The methods `norm` and `norm.boot` implement (normal linear) stochastic regression but taking parameter uncertainty into account, while `norm.nob` does not take parameter uncertainty into account. For this problem, according to the problem description, for acknowledging parameter uncertainty case, use `norm.nob`, for not acknowledging parameter uncertainty case, use `norm.boot`.

To analyse, first, use 2 methods impute 100 subdatasets, calculate how may times that the interval contains the true value of the parameter $\beta_1$ with 95% interval confidence. In the end, calculate the empirical coverage probability.

```r
# load the data
load('dataex2.Rdata')
real.beta <- 3

# consider parameters uncertainty
y.inter <- 0
for(i in 1:100){
  # select data set
  data <- dataex2[,,i]
  # fit models
  y.un.obj <- pool(with(mice(data, printFlag = F,seed = 1,m=20,method = "norm.nob"),
                   lm(Y~X)))
  # evaluate if the interval contains the true value
  y.un <- summary(y.un.obj,conf.int = TRUE)
  if(real.beta>=y.un[2,7] && real.beta<=y.un[2,8]){
    y.inter <- y.inter + 1
  }
}

# not consider parameters uncertainty
n.inter <- 0
for(i in 1:100){
  # select data set
  data <- dataex2[,,i]
  # fit models
  n.un.obj <- pool(with(mice(data,printFlag=F,seed=1,m=20,method = "norm.boot"),
                   lm(Y~X)))
  # evaluate if the interval contains the true value
  n.un <- summary(n.un.obj,conf.int = TRUE)
  if(real.beta>=n.un[2,7] && real.beta<=n.un[2,8]){
    n.inter <- n.inter + 1
  }
}

# calculate the empirical coverage probability
cat("condsidering parameters uncertainty:",y.inter,"%\n")
```

```
## condsidering parameters uncertainty: 88 %
```

```r
cat("not condsidering parameters uncertainty:",n.inter,"%")
```

```
## not condsidering parameters uncertainty: 95 %
```

The result shows that in this problem, when considering the uncertainty of the estimated coefficients, the coverage probability is 88%, which is lower than the nominal level of 95%. This suggests that the confidence intervals obtained from this method may not be reliable. However, when not considering the uncertainty of the estimated coefficients, the coverage probability is 95%, which is closer to the confidence level. This indicates that ignoring the uncertainty may lead to more reliable confidence intervals in this problem.

## Question 3

The two strategies are coincident because they both follow Rubin's rule for combining multiply imputed data. Rubin's rule states that the parameter estimates (including both the coefficients and the predicted values) should be pooled across the imputed datasets.

- **Stretegy (i)**

  In this strategy, the predicted values are first computed for each imputed dataset and then averaged across the datasets using Rubin's rule.

- **Strategy (ii)**

  In this strategy, the regression coefficients are first pooled across the imputed datasets using Rubin's rule, and then the predicted values are computed using the pooled coefficients.

Since Rubin's rule ensures that the parameter estimates are combined in an appropriate and efficient way, both cases will produce the same results. Therefore, the two cases are equivalent and coincide.

More specifically, an linear regression model can be written as:

$$\hat{y} = \beta_i * x_i$$

For strategy(i), in step 2, for each $m \in seq(1 : M)$ (M was defined in step 1 with default value equal to 5), we have $\hat{y}^m = \hat{x}_i^m * \hat{\beta}_i^m$. And in step 3, after pooled, final result $\hat{y} = \sum_{m=1}^{M} \hat{y}^m$.

For strategy(ii), in step2, for each $m \in seq(1 : M)$, calculate $\hat{\beta}_i^m$. Then for each $i$ , calculate $\hat{\beta}_i = \frac{\sum_{m=1}^{M} \hat{\beta}_m^i}{M}$. Use this $\hat{\beta}_i$, calculate $\hat{y} = \hat{\beta}_i * \hat{x}_i$.

Comparing the 2 $\hat{y}$, we can find that the result is same based on Rubin's rule.

## Question 4

### Sub-Question (a)

As $x_2$ is complete, if we only using $x_1$ and $y$ variables in step1, we can use default function.

```
# load data
load('dataex4.Rdata')
mice.4a <- mice(dataex4,m=50,seed=1,printFlag = F)

# check if we only impute x_1 and y or not
mice.4a$method
```

```
##     y    x1    x2
## "pmm" "pmm"    ""
```

As the method for $x_2$ is " ", we only using $x_1$ and $y$ variables in step1.

```
# impute incomplete data
result.4a <- pool(with(mice.4a,lm(y~x1+x2+x1*x2)))
# calculate 95% confidence interval
summary(result.4a, conf.int = TRUE)[, c(1, 2, 7, 8)]
```

```
##          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
```

We obtain $\beta_1 = 1.411$, the confidence interval with 95% confidence level is $(1.219, 1.603]$; $\beta_2 = 1.966$, the confidence interval with 95% confidence level is $(1.861, 2.071]$; $\beta_3 = 0.755$, the confidence interval with 95% confidence level is $(0.642, 0.868]$. $\beta_1$,$\beta_2$ and $\beta_3$ are all in corresponding 95% confidence intervals.

### Sub-Question (b)

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

## using passive imputation to impute the interaction variable
imp0.4b <- mice(dataex4,m=50,seed=1,printFlag = F)

# specify a formula to calculate x1*x2
method.4b <- imp0.4b$method
method.4b["x1x2"] <- "~I(x1*x2)"

pred.4b <- imp0.4b$predictorMatrix
# x1x2 will not be used as predictor of x1 and x2
pred.4b[c("x1","x2"),"x1x2"] <- 0
pred.4b[, c("x1", "x2")] <- 0
```

```r
# x1, x2 be included in the imputation model of each other
pred.4b["x1", "x2"] <- 1
pred.4b["x2", "x1"] <- 1

# ensure variables are imputed by right sequence
seq.4b <- c("y","x1","x2","x1x2")

# imp0.4b.passive <- mice(dataex4,m-50,see)
imp.4b.passive <- mice(dataex4, method = method.4b, predictorMatrix = pred.4b,
                       visitSequence = seq.4b, #maxit = 20,
                       m = 50, seed = 1, printFlag = FALSE)

# step2 and step3
result.4b <- pool(with(imp.4b.passive,lm(y~x1+x2+x1*x2)))

# calculate 95% confidence interval
summary(result.4b, conf.int = TRUE)[, c(1, 2, 7, 8)]
```

```
##          term  estimate     2.5 %   97.5 %
## 1 (Intercept) 2.1654541 1.8968644 2.434044
## 2          x1 0.9761881 0.6992222 1.253154
## 3          x2 1.6168272 1.4688180 1.764836
## 4       x1:x2 0.9470357 0.7999456 1.094126
```

After appending interaction variable to the dataset, by using *passive imputation* to impute the interaction variable, we obtain $\beta_1 = 0.976$, the confidence interval with 95% confidence level is $(0.699, 1.253]$; $\beta_2 = 1.618$, the confidence interval with 95% confidence level is $(1.469, 1.765]$; $\beta_3 = 0.947$, the confidence interval with 95% confidence level is $(0.800, 1.094]$. $\beta_1, \beta_2$ and $\beta_3$ are all in the corresponding 95% confidence intervals.

**Sub-Question (c)**

```r
# imputation
result.4c <- pool(with(mice(dataex4,m = 50, seed = 1,
                            printFlag = FALSE),lm(y~x1+x2+x1x2)))
# obtain 95% confidence interval
summary(result.4c, conf.int = TRUE)[, c(1, 2, 7, 8)]
```

```
##          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
```

By the above processes, we obtain $\beta_1 = 1.003$, the confidence interval with 95% confidence level is $(0.841, 1.166]$; $\beta_2 = 2.026$, the confidence interval with 95% confidence level is $(1.940, 2.112]$; $\beta_3 = 1.018$, the confidence interval with 95% confidence level is $(0.930, 1.105]$. $\beta_1, \beta_2$ and $\beta_3$ are all in the corresponding 95% confidence intervals.

### Sub-Question (d)

The obvious conceptual drawback of the just another variable approach for imputing interactions is that it ignores the fact that the interaction term is not just another variable in the dataset but rather a derived variable that depends on the values of the two interacting variables. Treating the interaction term $x_1 \times x_2$ as a distinct variable in imputation overlooks the fact that it is a product of $x_1$ and $x_2$.

By imputing the interaction variable as if it were just another variable, the imputed values may not be consistent with the values of the interacting variables, which can result in biased estimates of the interaction effect and incorrect statistical inferences and reduced efficiency in subsequent analyses. In contrast, the other two methods take into account the nature of the interaction term.

## Question 5

We have a dataset called **NHANES2**. It contains data from *National Health and Nutrition Examination Survey.* This survey's goal is to access the health and nutrition status of American adults and children. First, we need to have an overview of the dataset.

### Overview of the NHANES2

```
# load data
load('NHANES2.Rdata')
# check the dimension
cat("dimension:",dim(NHANES2),"\n")
```

```
## dimension: 500 12
```

```
# check not available value
colSums(is.na(NHANES2))
```

```
##    wgt gender   bili    age   chol    HDL    hgt   educ   race    SBP hypten
##      0      0     47      0     41     41     11      1      0     29     21
##     WC
##     23
```

```
# check incomplete cases percentage
cat("incomplete case percentage:",
    nrow(ic(NHANES2))/(nrow(NHANES2)),'\n')
```

```
## incomplete case percentage: 0.178
```

```
# check incomplete cases percentage for interested columns
cat("incomplete case percentage for intersted clumns:",
    nrow(ic(NHANES2[,c("WC","gender","age","hgt")]))/(nrow(NHANES2)))
```

```
## incomplete case percentage for intersted clumns: 0.064
```

We have 12 columns corresponding to 12 variables and 500 records(cases/samples). Among 12 variables, `bili` has 47 NA values, `HDL` has 41 NA values, `hgt` has 11 NA values, `educ` has 1 NA value, `SBP` has 29 values, `hypten` has 21 NA values and `WC` has 23 NA values. The incomplete case percentage is 17.8%. And among 4 variables our analysis interested in, the incomplete case percentage is 6.4%.

Then, using package `JointAI`, by the useful visualisation functions, inspect the missing data patterns.

```
require(JointAI)
```

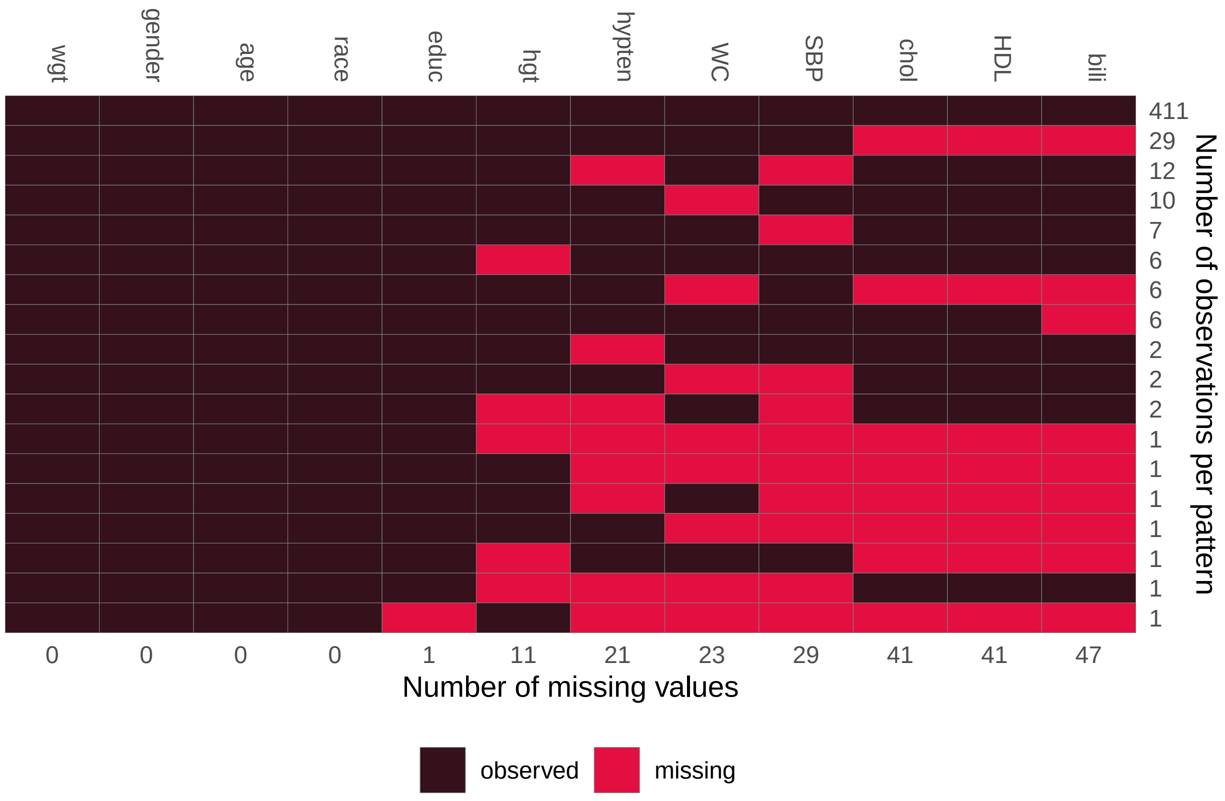
```
## Loading required package: JointAI
```

```
## Please report any bugs to the package maintainer (https://github.com/NErler/JointAI/issues).
```

```
##
## Attaching package: 'JointAI'
```

```
## The following object is masked _by_ '.GlobalEnv':
##
##     NHANES
```

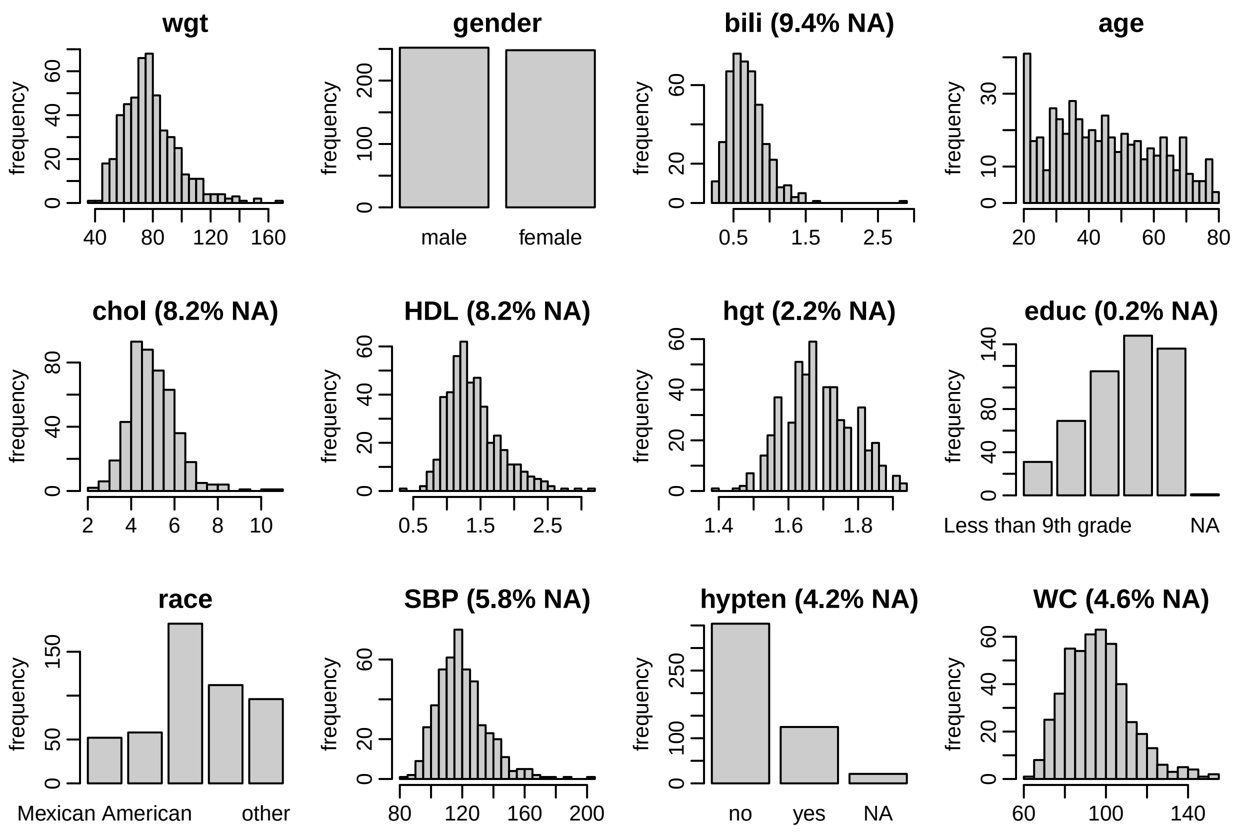
```
md_pattern(NHANES2, pattern = FALSE, color = c('#34111b', '#e30f41'))
```



The missing data of `chol` and `HDL` variables seem to be same. The missing data locations are corresponding and they both have 41 missing values. Missing values for variable `bili` 's locations and number is very similar to `HDL` and `chol`. The missing data for `WC`, `hypten` and `hgt` are similar. And `educ` only have 1 missing value, maybe that is totally because of random.

Then using package `JointAI`, visualising how the observed parts of the incomplete variables are distributed.

```
par(mar = c(3, 3, 2, 1), mgp = c(2, 0.6, 0))
plot_all(NHANES2, breaks = 30, ncol = 4)
```



We can obtain many information from figures above intuitionly.For example, the distribution shapes of `wgt`, `HDL` and `SBP` look similar, and `hgt` seems like a normal distribution. But we do not have enough evidence to prove they have exactly relationships, but we can have an overview of the distributions of the 12 variables.

In the end, check the class of each variables.

```
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 ...
```

We can find that `educ`, `race` and `gender` are factors. Other variables are continuous numeric.

Besides, based on problem description, it is possible that there are some correlations among the variables in the dataset. For example, age and gender may be associated with blood pressure and cholesterol levels. Height and weight may also be correlated with blood pressure and waist circumference. Education level and race may also be associated with health outcomes, although these relationships may be more complex and indirect.

### Imputation

First, start by doing a dry run of mice(), without any iterations, which will create the default versions of everything that needs to be specified.

```r
imp0 <- mice(NHANES2, maxit = 0,seed = 1)
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
```

The default method for each incomplete numeric variables are `pmm` and for factor variables are `logreg`. From the previous plots, `chol` and `HDL` may have strong relationships, but we need to be careful and check their relationship by scatter plot.

```r
require(VIM)
```

```
## Loading required package: VIM
```

```
## Loading required package: colorspace
```

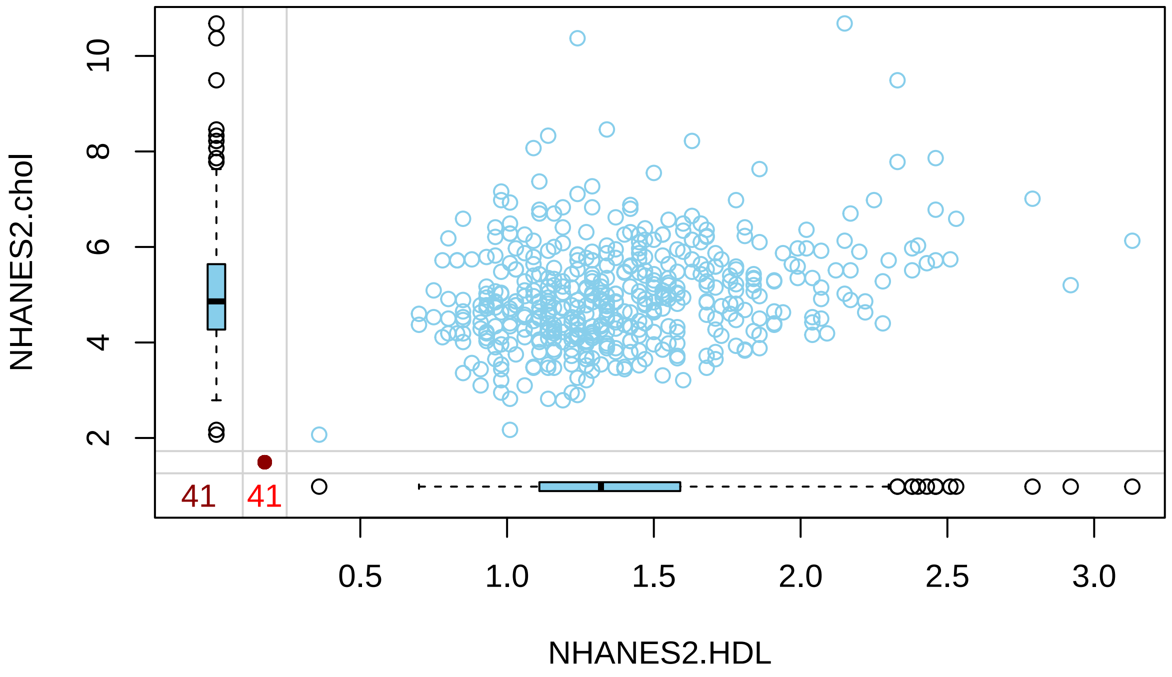
```
## Loading required package: grid
```

```
## VIM is ready to use.
```

```
## Suggestions and bug-reports can be submitted at: https://github.com/statistikat/VIM/issues
```

```
##
## Attaching package: 'VIM'

## The following object is masked from 'package:datasets':
##
##     sleep
```

```
marginplot(data.frame(NHANES2$HDL,NHANES2$chol))
```



Seems like although the missing values' number and locations seem same, but we can not find good relation between them. And although distribution of variable `hgt`, which represents height, looks like normal distribution, it may have relations between age and weight, hence we cannot impute it by "norm" roughly.

To be more conservative, I choose to set `maxit = 20`.

```
sum(is.na(NHANES2))/7
```

```
## [1] 30.57143
```

And due to the average missing value is incomplete variables(without `educ`, as it only has 1 missing value) is 30,I choose to set `M = 30`, running `mice()` function further.

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

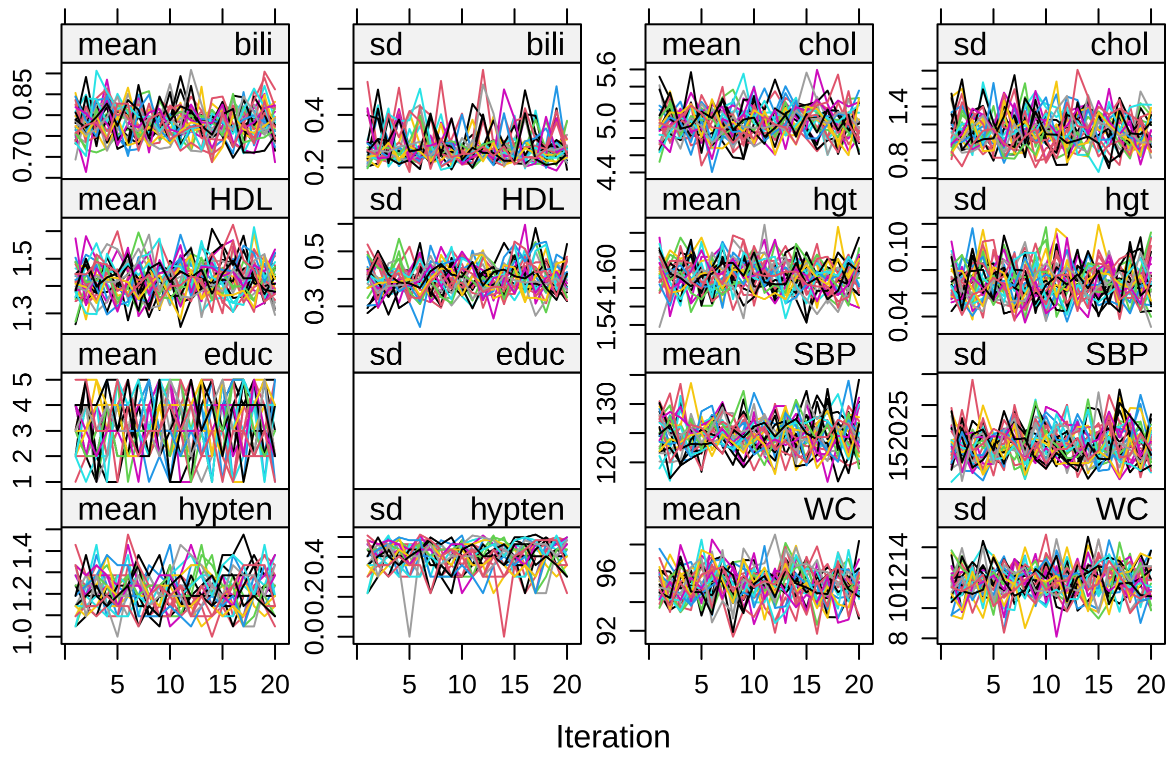
Checking the `loggedEvents` contained in our object imp allows us to know if `mice()` detected any problems during the imputation.

```
imp$loggedEvents
```

```
## NULL
```

The result is NULL, which means `mice()` does not detecte problems during the imputation. Then plotting our object and visualise the traceplots, checking whether the MICE algorithmhas converged.

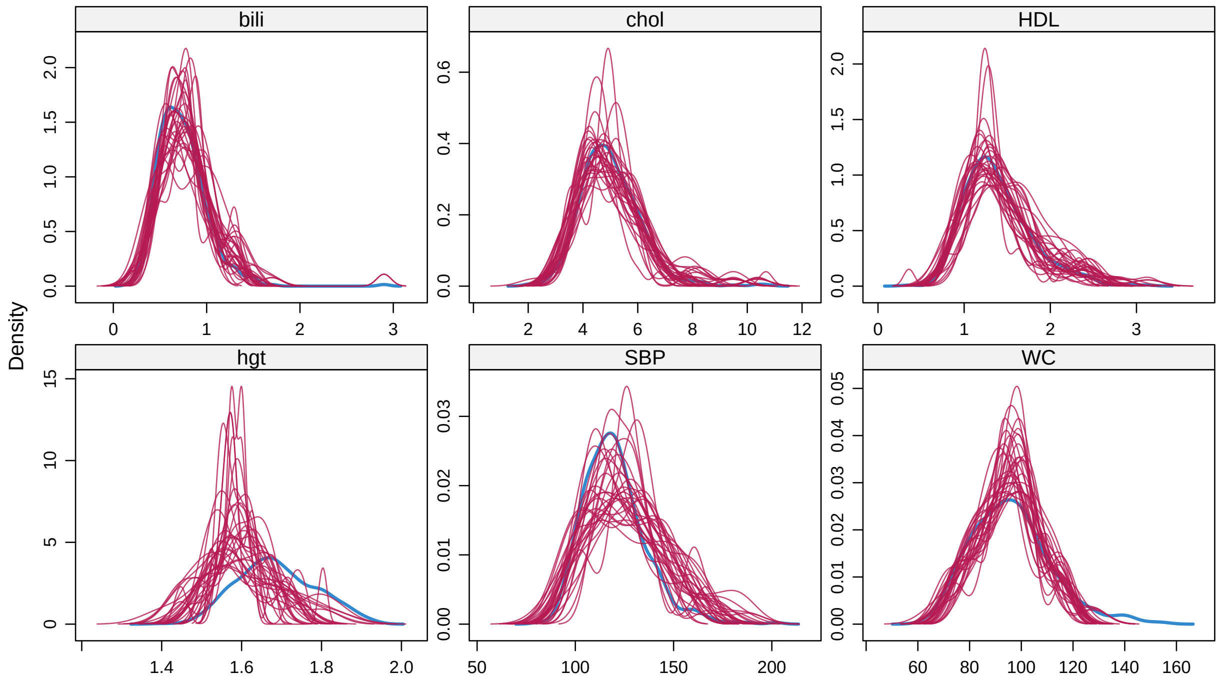
```
plot(imp, layout = c(4,4))
```



In each subplots of the figure above, some information on multiple imputation will be plotted. We can obtain information about each incomplete variables' standard error and mean value in 20 iteration. And each lines represent a m (m = 30). There is nothing plotted on the `educ` standard error sub-plot as it only have 1 missing value. We can find that basically, for each variables, the imputation are fluctuating in a reasonable converged. Although some variables, like `hypen` 's standard error, have 2 values very close to 0 and differ from other values, most of them appears to have converged.

Then, we can compare the distribution of the imputed values against the distribution of the observed values. We start doing that for the continuous variables.

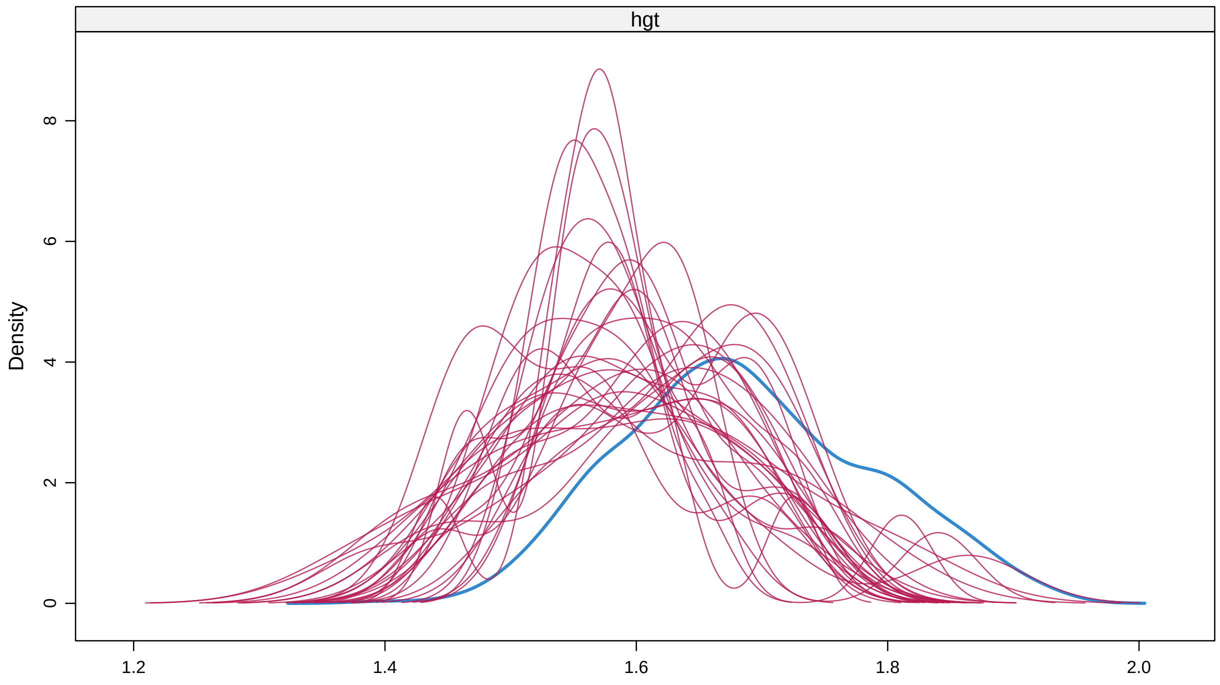
```
densityplot(imp)
```



The function `densityplot()` is used to draw the density estimates of each variable after multiple imputation, which is usually used to check the distribution and potential skewness of variables. In this plot, there is a density curve for each variable, where the red curve represents the original data and the blue curve represents the results after multiple imputation. If the blue curve and the red curve overlap closely, it can be considered that the multiple imputation has a good effect, and the distribution of the imputed data is similar to that of the original data. On the other hand, if the blue curve differs significantly from the red curve, it may be necessary to reconsider the imputation method or take measures such as increasing the sample size.

We can find that the imputation for `hgt` seems have an obvious difference from observation samples, and `SBP` follows. Consider about `hgt` looks like normal distribution, we can have a test.

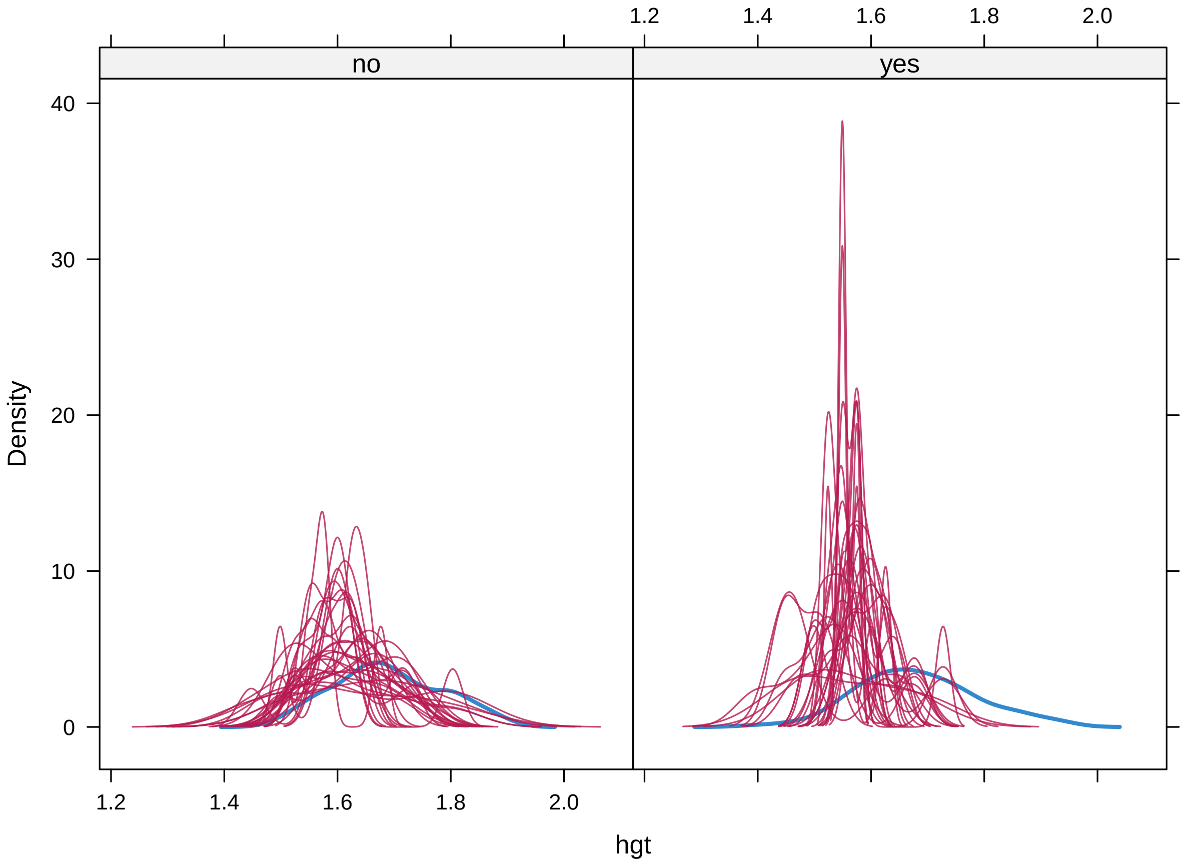
```
meth <- imp0$method
meth["hgt"] <- "norm"
imp.test <- mice(NHANES2, maxit = 20, m = 30, seed = 1,
                 method = meth, printFlag = FALSE)
densityplot(imp.test)[4]
```



The result is still not good, therefore, we can do further analyse to check variables relationships among these 2 values and other incomplete factor variables. More specifically, consider the `gender` and `hypten`.

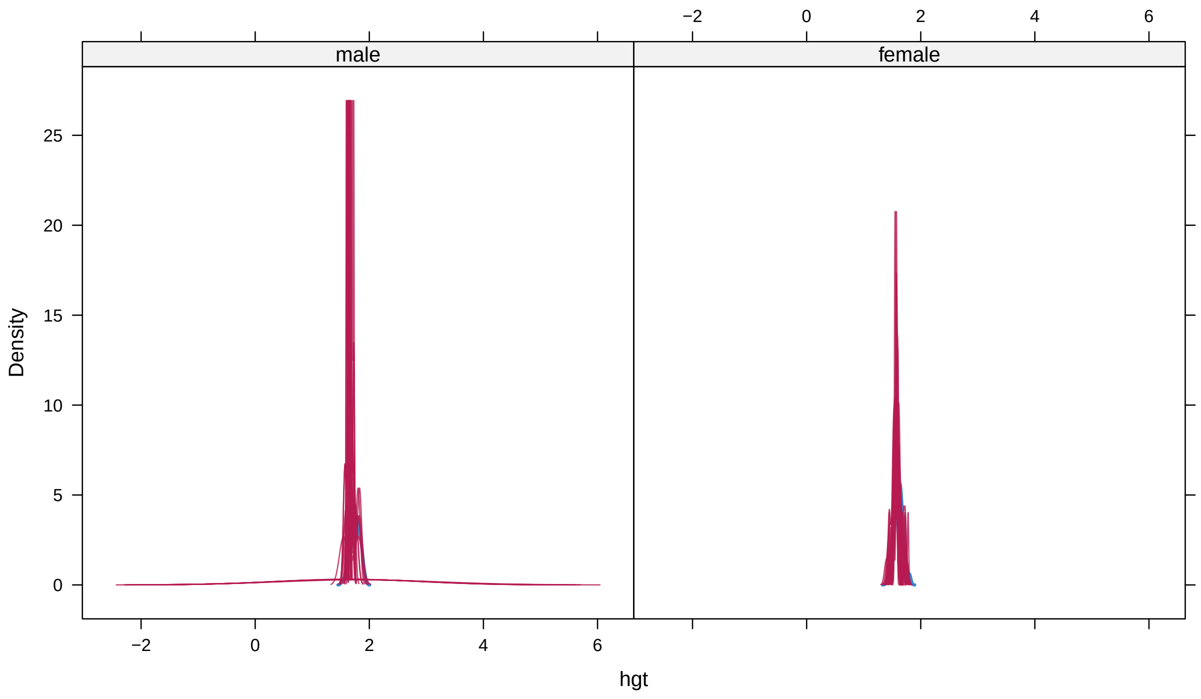
Check `hgt` and `hypten`.

```
densityplot(imp, ~hgt|hypten)
```



Check `hgt` and `gender`.

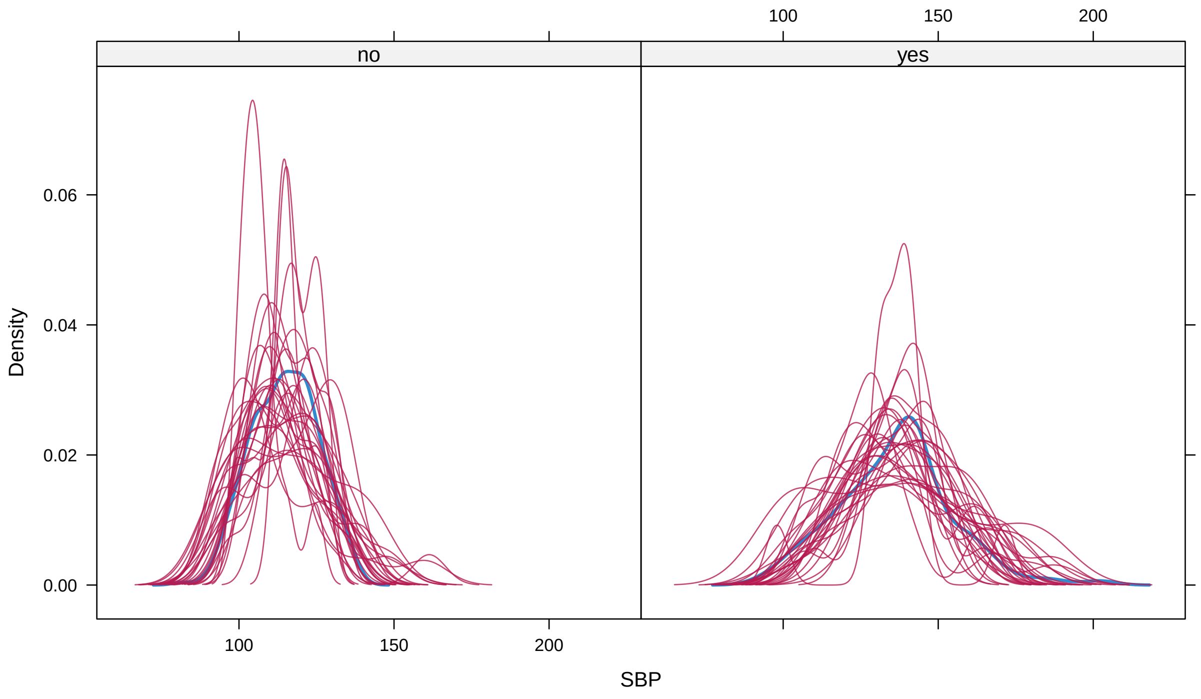
```
densityplot(imp, ~hgt|gender)
```



It is clear that, height has no relation with **hypten** and may have relations with gender, but as the height values are too concentrated, the relation is not very clear.

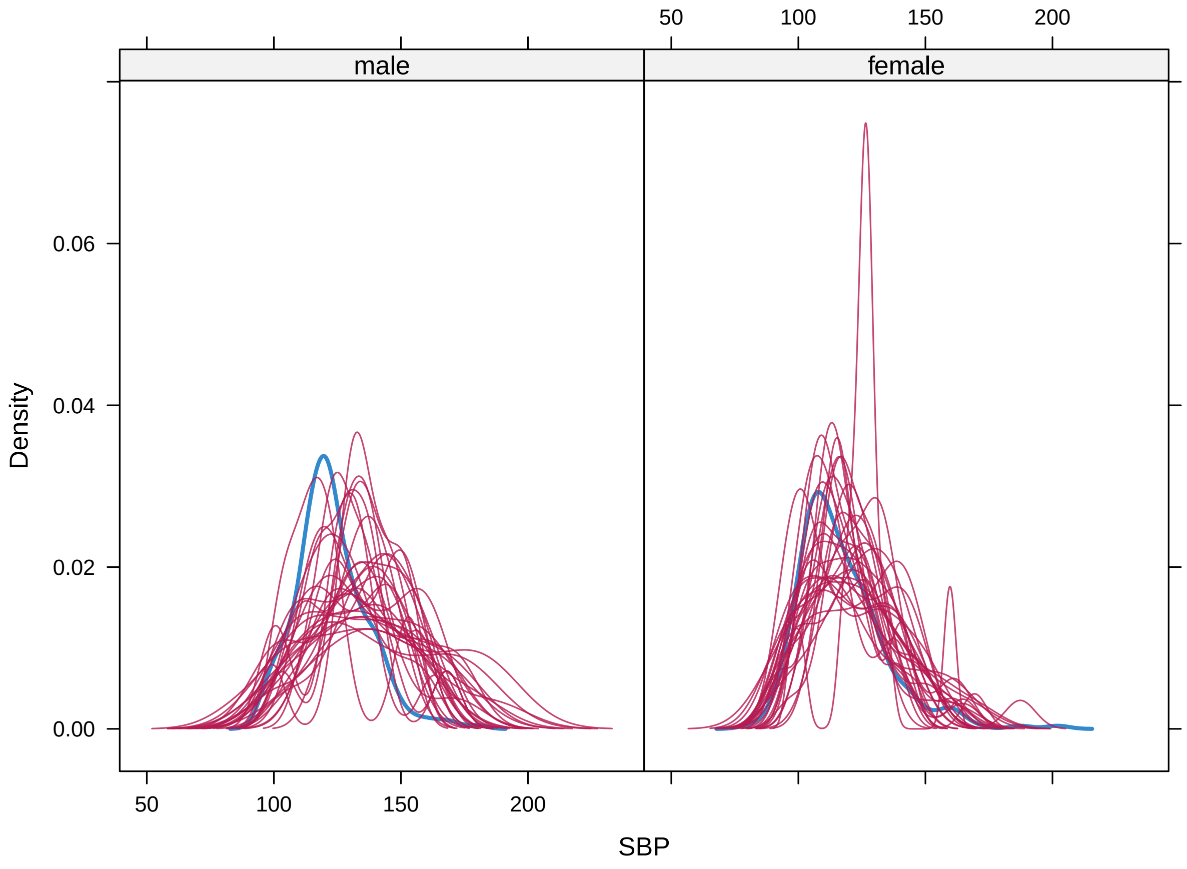
Check `SBP` and **hypten**.

```
densityplot(imp, ~SBP|hypten)
```



Check SBP and gender.

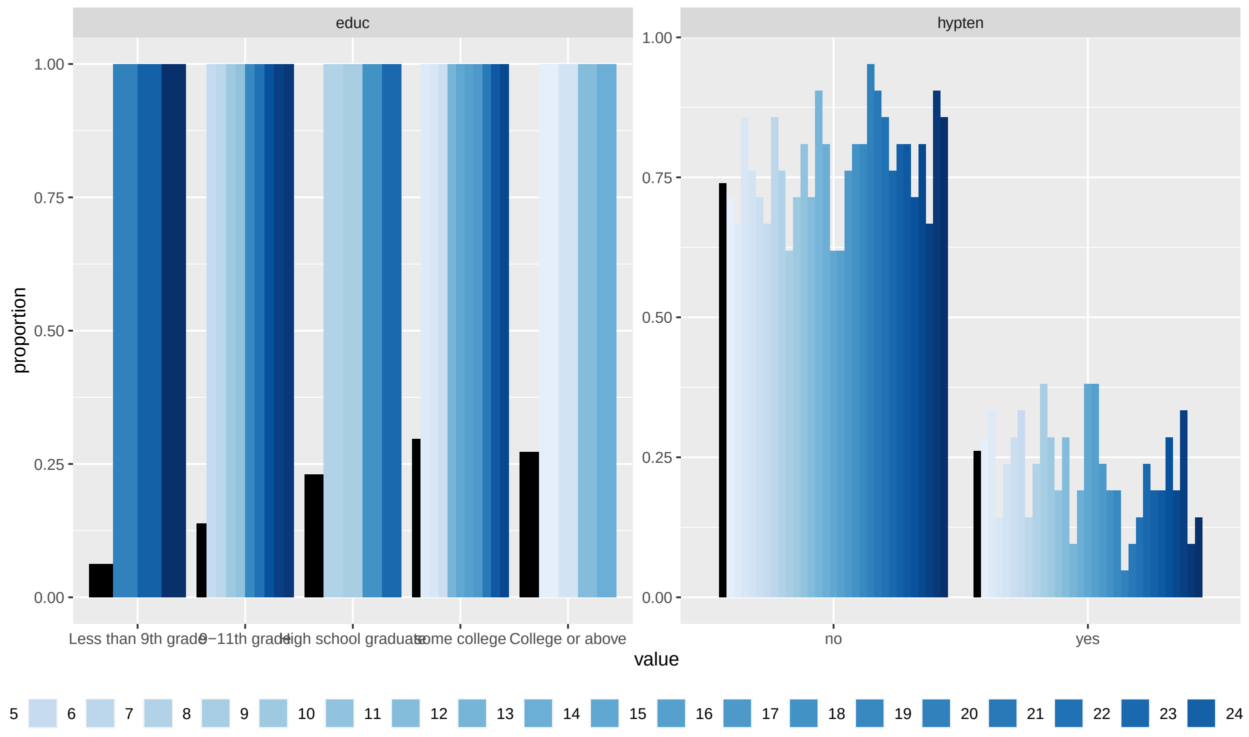
```
densityplot(imp, ~SBP|gender)
```



Gender and hypten status appear to have a moderate effect on the discrepancies observed between the imputed and observed values of SBP.

After comparing the distributions of the imputed values for the continuous variables, check for binary/categorical variables. Here we use function implemented by Nicole Erler.

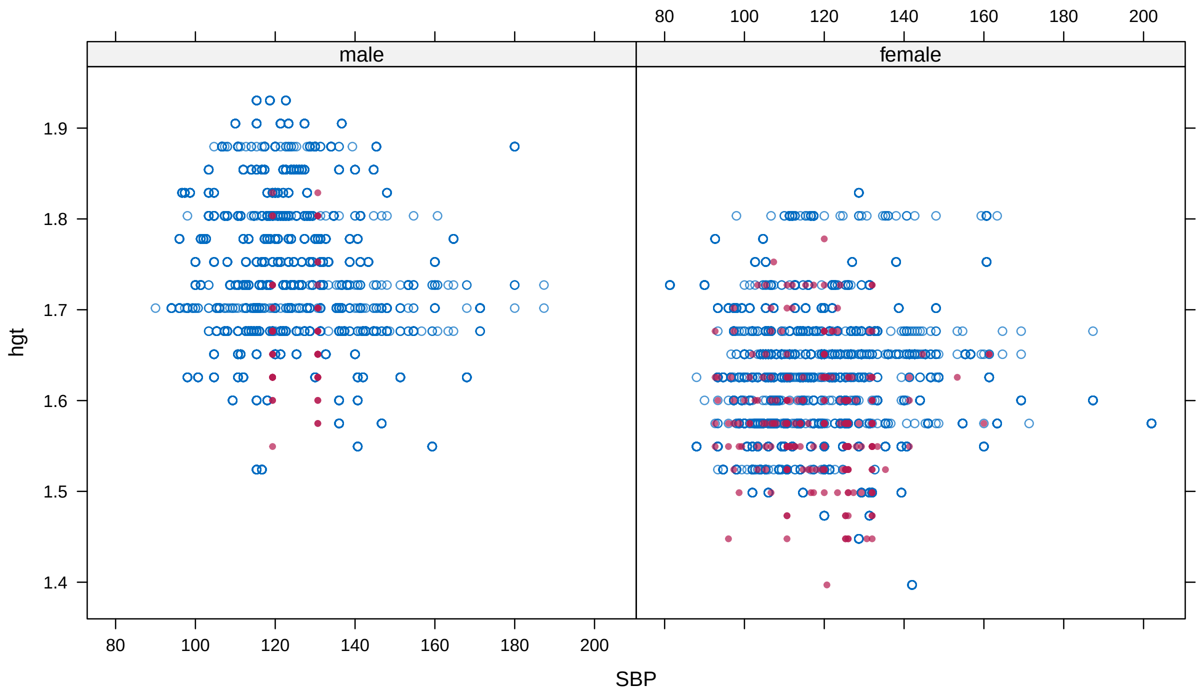
```
require(devtools)
require(reshape2)
require(RColorBrewer)
require(ggplot2)
source_url("https://gist.githubusercontent.com/NErler/0d00375da460dd33839b98faeee2fdab/raw/c6f537ecf80e

propplot(imp)
```



For categories variables, we notice a significant difference between the distribution of the observed data and the imputed data for the variable of educational status. However, since only one value is missing out of a total of 500, there is no need to be overly concerned about this discrepancy.

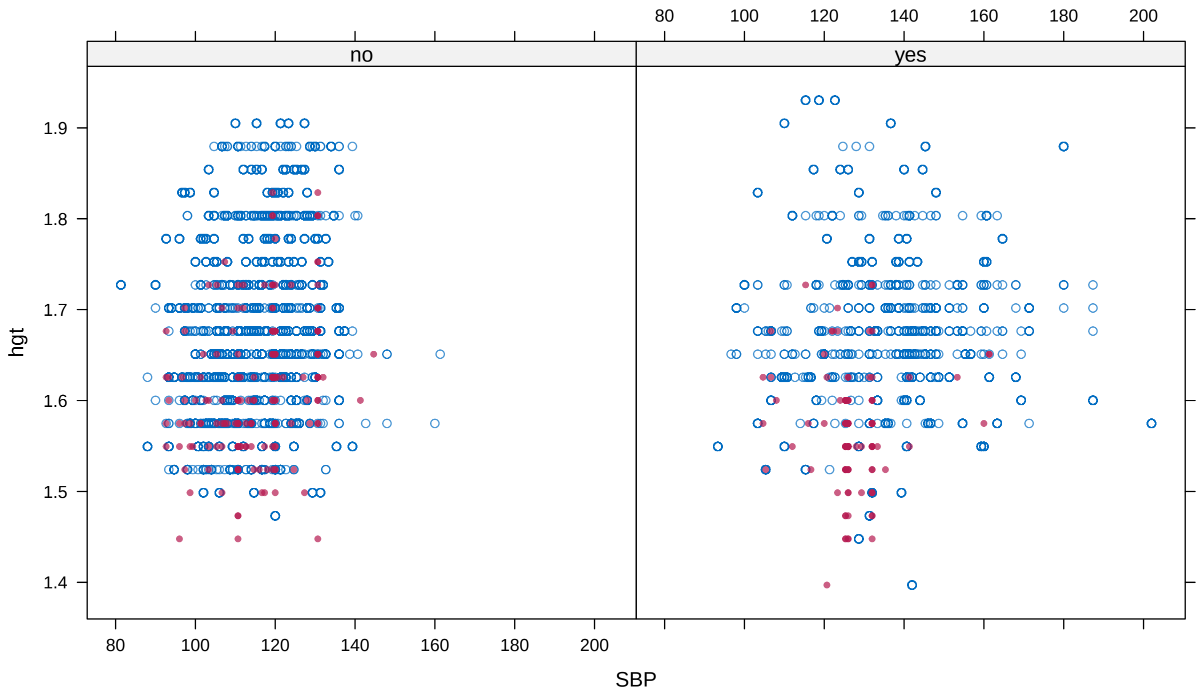
Besides, using function `xyplot()` allows to visualize scatterplots of the imputed and observed values for pairs of variables.

```
xyplot(imp, hgt ~ SBP | gender, pch = c(1, 20))
```



From the figure above, we can find that the `hgt` of male is higher for male averagely, and the incomplete cases for female's `hgt` and `SBP` is much more than male's.

```
xyplot(imp, hgt ~ SBP | hypten, pch = c(1, 20))
```



From the figure above, we can find that the samples `SBP` for `hypten` equal to no is more concentrated, and the incomplete cases for `hypten` equal to no is higher that `hypten` equal to yes.

### Analysis of the Imputed Data

Now that we have verified the success of our imputation process, we can move forward with analyzing the imputed data. The primary model we are interested in is:

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

Based on the above model, we can move forward with analyzing the imputed data.

```
fits <-  with(imp, lm(wgt ~ gender + age + hgt + WC))
```

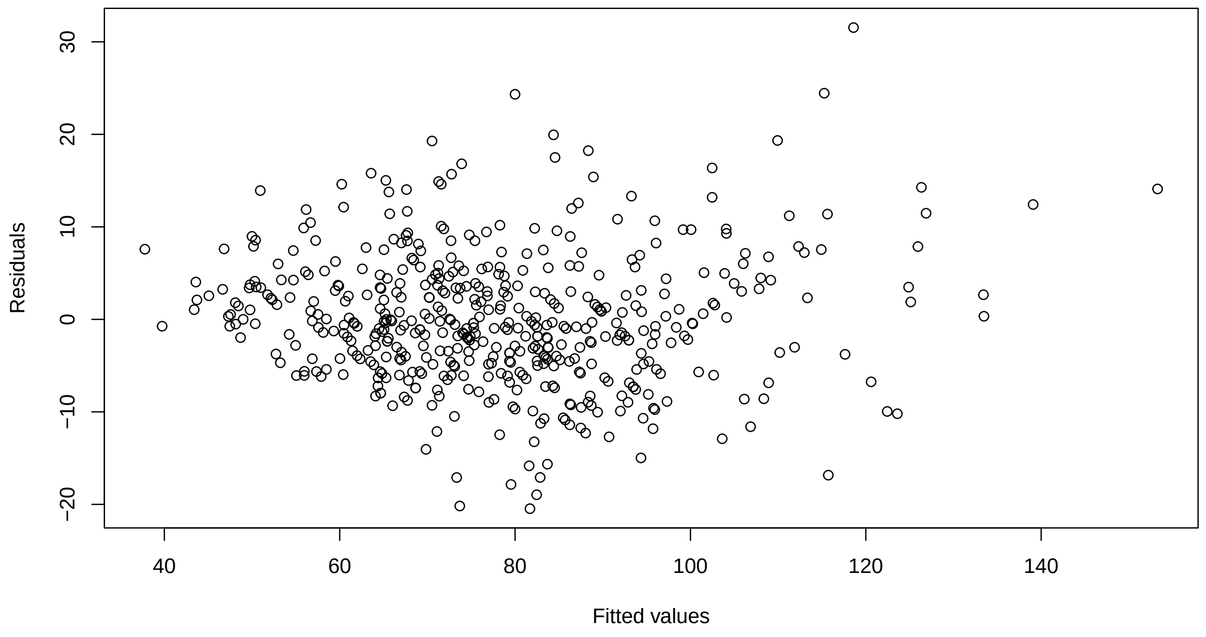
We can obtain additional information available in the object `fits` that we can investigate. First, we can look at the summary of the fitted model in the first imputed dataset.

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

```
##
## Call:
## lm(formula = wgt ~ gender + age + hgt + WC)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -20.4638  -4.5537  -0.4955   3.8854  31.5403
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -100.51035    7.50652 -13.390  < 2e-16 ***
## genderfemale   -1.26815    0.81952  -1.547    0.122
## age            -0.15827    0.02085  -7.590  1.6e-13 ***
## hgt            52.15392    4.29615  12.140  < 2e-16 ***
## WC              1.02795    0.02213  46.452  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.179 on 495 degrees of freedom
## Multiple R-squared:  0.8575, Adjusted R-squared:  0.8563
## F-statistic: 744.6 on 4 and 495 DF,  p-value: < 2.2e-16
```

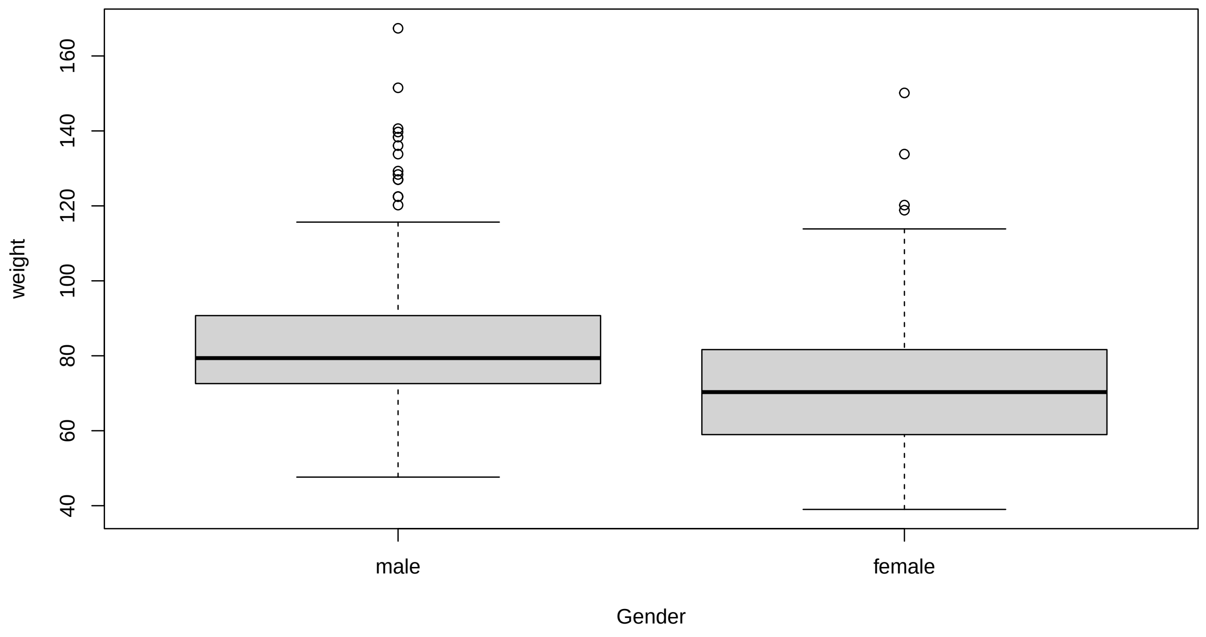
In addition, for model specification and validation purposes, such as applying transformations, we have the option of examining either the complete cases or utilizing one of the imputed/completed datasets. It is important to ensure that any transformations are applied consistently across all datasets and not specific to just one. If we determine that transformations are necessary, we may need to revisit the imputation models and fit them using the transformed variables.

```
comp1 <- complete(imp, 1)
plot(fits$analyses[[1]]$fitted.values,
     residuals(fits$analyses[[1]]),
     xlab = "Fitted values", ylab = "Residuals")
```

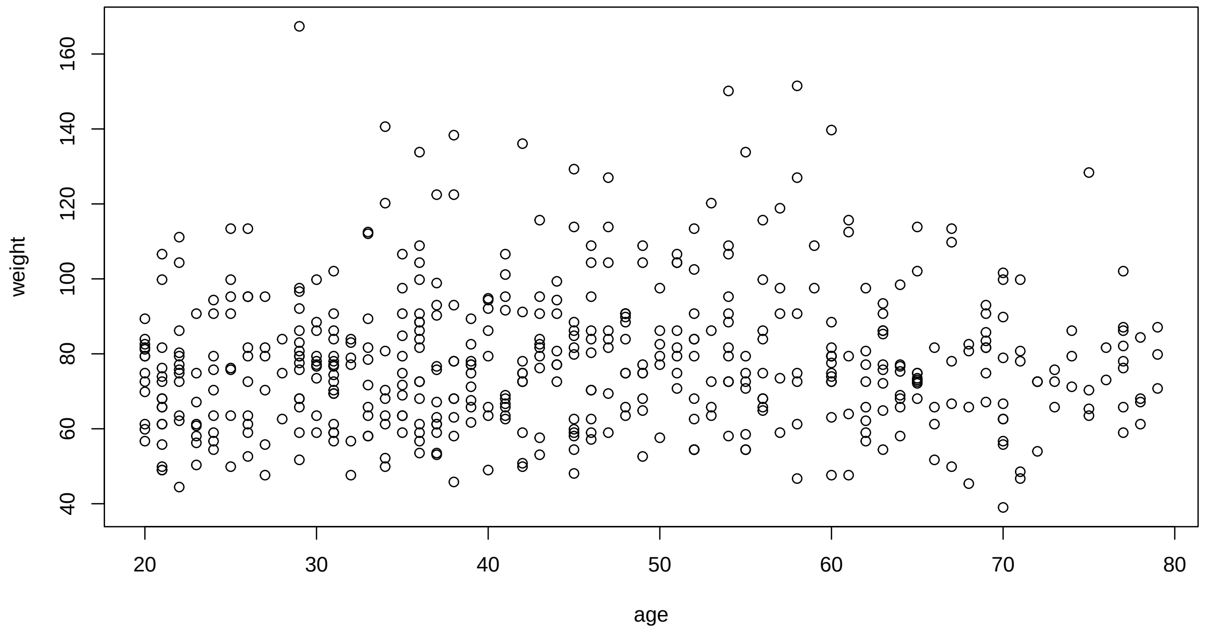


In the above plot, the x-axis represents the fitted values from the regression model, which are the predicted values, and the y-axis represents the difference between the predicted values and the actual observed values, which are the residuals. The fitted values are mostly fluctuating around zero residuals.

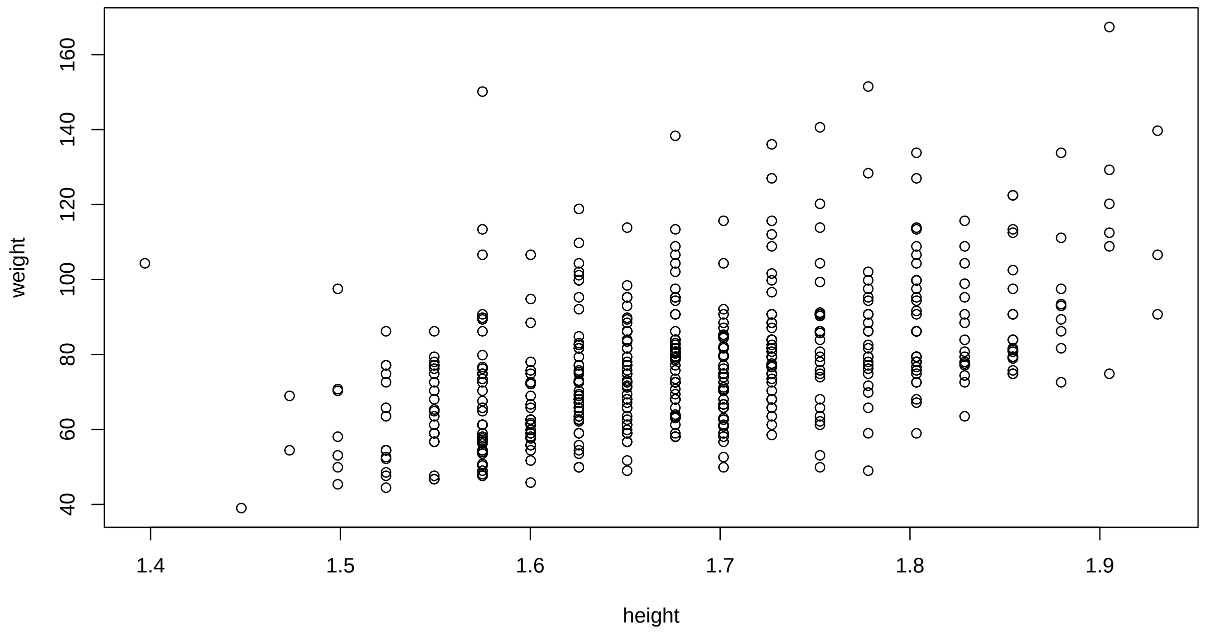
```
plot(comp1$wgt ~ comp1$gender, xlab = "Gender", ylab = "weight")
```



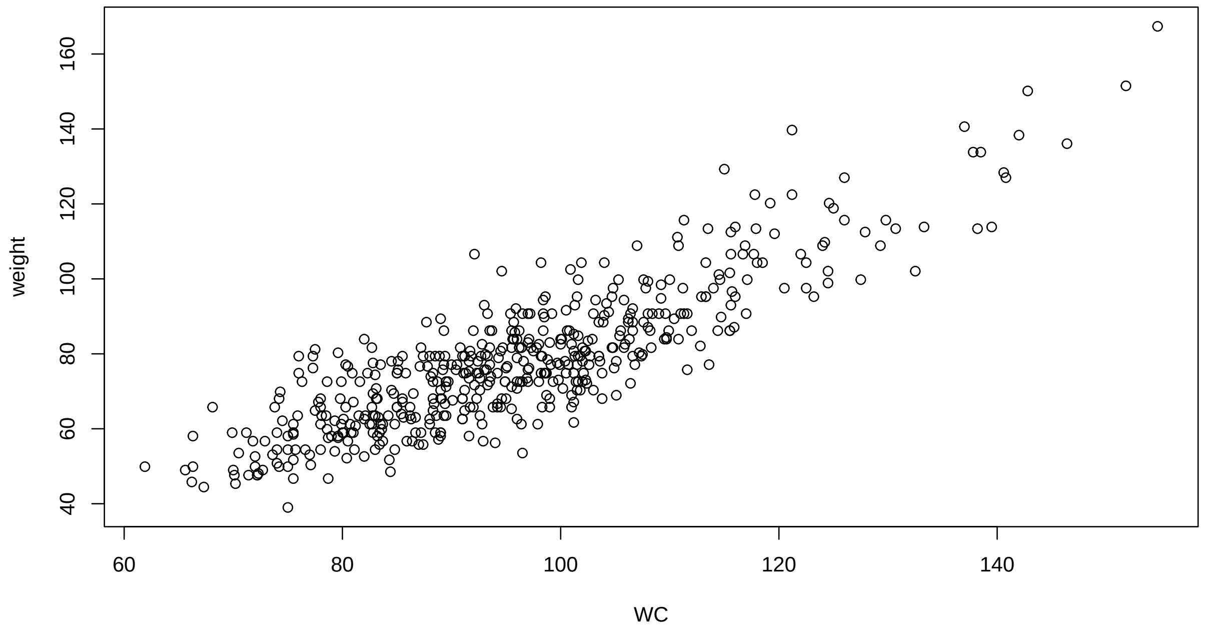
```r
plot(comp1$wgt ~ comp1$age, xlab = "age", ylab = "weight")
```



```r
plot(comp1$wgt ~ comp1$hgt, xlab = "height", ylab = "weight")
```



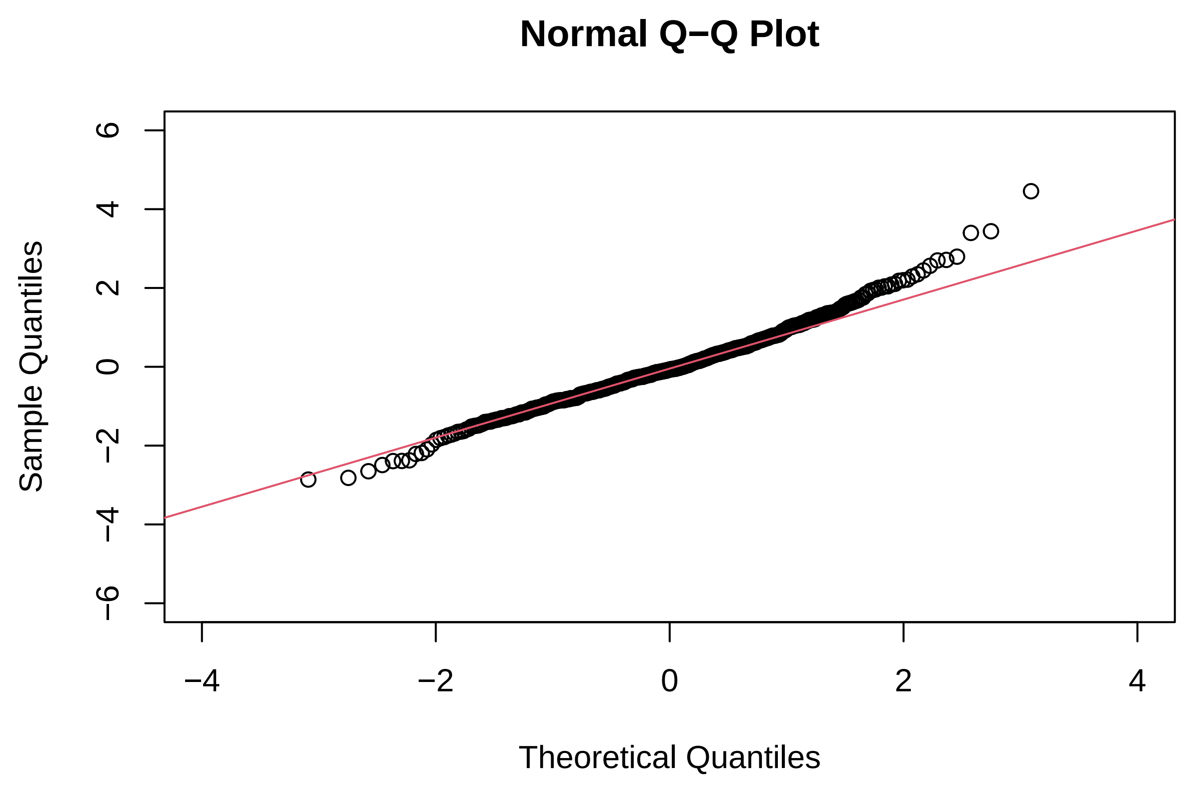
```r
plot(comp1$wgt ~ comp1$WC, xlab = "WC", ylab = "weight")
```



Using `plot()` function, we can view the data based on the model we are interested in after imputing in the 4 figures above. We can find weight a little bit differ from genders, uniform shape for weight with different ages, the increasing weights with increasing heights and positive linear relationship between weight and WC.

Besides, we can also do a QQplot and nothing looks suspicious.

```r
qqnorm(rstandard(fits$analyses[[1]]), xlim = c(-4, 4), ylim = c(-6, 6))
qqline(rstandard(fits$analyses[[1]]), col = 2)
```



### Pooling the Results

We can pooling the results by the follow codes and look the summary result.

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

```
##           term     estimate  std.error  statistic       df       p.value
## 1  (Intercept) -100.8520702 7.67289217 -13.143945 449.6454  1.273238e-33
## 2 genderfemale   -1.3850796 0.83448095  -1.659810 469.9080  9.761988e-02
## 3          age   -0.1576829 0.02141424  -7.363458 451.6777  8.561184e-13
## 4          hgt   52.4292200 4.39636603  11.925581 444.3719  1.228343e-28
## 5           WC    1.0260613 0.02232811  45.953795 481.3369 3.869359e-178
##          2.5 %      97.5 %
## 1 -115.9312510 -85.7728894
## 2   -3.0248557   0.2546965
## 3   -0.1997668  -0.1155990
## 4   43.7889680  61.0694720
## 5    0.9821887   1.0699340
```

After pooling the result, using `mice` package's function evaluating model fit, first calculates the pooled $R^2$.

```
pool.r.squared(pooled_ests, adjusted = TRUE)
```

```
##                est     lo 95     hi 95        fmi
## adj R^2 0.8559941 0.8304596 0.8779613 0.02126541
```

The result is a point estimate of the R-squared value for the model, along with 95% confidence intervals and the fractional multiple imputation (FMI) factor. The adjusted R-squared value of 0.856 suggests that the model explains a large proportion of the variance in the data, after accounting for the number of predictors. The confidence intervals indicate that this estimate is precise, and the FMI factor suggests that the variance due to imputation is small relative to the total variance. Overall, these results indicate that the model is a good fit for the data.

Then, using function `D1()` and `D2()` to compare nested models. Mice provides several functions for comparing nested models: `D1()`, `D2()`, and `D3()`. These functions implement different tests, such as multivariate Wald test and likelihood-ratio test statistic, to compare the models. If no variance-covariance matrix is available, the D2() function can be used to pool test statistics.

Based on the model we are interested in, I want to divide the model's variables into 2 parts. First, test for "personal" 's part, including `gender` and `age`.

```
fit.personal <- with(imp, lm(wgt ~ age + gender))
D1(fits,fit.personal)$result
```

```
##       F.value df1      df2          P(>F)        RIV
## [1,] 1289.297   2 486.6161 3.390217e-195 0.03734877
```

The result is statistically significant, with a very low p-value (3.39e-195) and a moderate RIV (0.037).

Then, test for 'body" s part, including `WC` and `hgt`.

```
fit.body <- with(imp, lm(wgt ~ WC + hgt))
D1(fits,fit.body)$result
```

```
##       F.value df1      df2        P(>F)       RIV
## [1,] 27.76323   2 484.8734 3.840751e-12 0.0424763
```

The output shows that the new model fit (fit.body) significantly improves upon the MI model fits (fits), as indicated by the low p-value (P(>F)) and relatively large RIV. The result may provide additional insights or interpretation of the output.