University of Edinburgh, School of Mathematics

Incomplete Data Analysis, 2020/2021

Assignment 3

Matilda Worlidge

All code included in this assignment is available in a GitHub repository at https://github.com/mworlidge/incomplete-data-analysis/blob/main/assignment3.Rmd. First I need to load all the necessary packages.

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

Question 1

Considering the `nhanes` dataset, we can use the functions `dim` and `str` to explore the data and to see what we're working with.

```r
dim(nhanes)
```

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

```r
str(nhanes)
```

```
## 'data.frame':    25 obs. of  4 variables:
##  $ age: num  1 2 1 3 1 3 1 1 2 2 ...
##  $ bmi: num  NA 22.7 NA NA 20.4 NA 22.5 30.1 22 NA ...
##  $ hyp: num  NA 1 1 NA 1 NA 1 1 1 NA ...
##  $ chl: num  NA 187 187 NA 113 184 118 187 238 NA ...
```

This shows that we have 25 cases of 4 variables, `age`, `bmi`, `hyp` and `chl`.

a)

The following code chunk shows that there are 13 complete cases and 12 incomplete cases. Therefore 48% of cases are incomplete.

```r
# number of complete cases
cc <- nrow(cc(nhanes))
# total number of cases
n <- nrow(nhanes)
# percentage of missing cases
(n - cc)/n * 100
```

```
## [1] 48
```

b)

Now we can impute the data using the package `mice` using the default `seed` = 1. In step 2 of the imputation (the `with()` function), we can predict `bmi` from `age`, `hyp` and `chl` by the normal linear regression model.

```r
imps <- mice(nhanes, printFlag = FALSE, seed = 1)
fits <- with(imps, lm(bmi ~ age + hyp + chl))
```

Now we can proceed to step 3 and pool the analyses.

```r
ests <- pool(fits)
ests$pooled[, c(1, 3, 10)]
```

```
##          term    estimate     lambda
## 1 (Intercept) 19.61789252 0.08938989
## 2         age -3.55287155 0.68640637
## 3         hyp  2.19701748 0.35043452
## 4         chl  0.05378081 0.30408063
```

The proportions of variance due to the missing data from each parameter is denoted by the column λ which is given by $\frac{B+\frac{B}{M}}{V^T}$. Therefore we have $\lambda_{\text{age}} = 0.6864$, $\lambda_{\text{hyp}} = 0.3504$, $\lambda_{\text{chl}} = 0.3040$. The parameter that appears to be most affected by the nonresponse is `age` since it has the highest value for the proportions of variance due to the missing data.

c)

We can now repeat this analysis with seeds 2, 3, 4, 5, and 6.

```r
#using the default M=6 but changing the seed
ests2 <- pool(with(mice(nhanes, printFlag = FALSE, seed = 2), lm(bmi ~ age + hyp + chl)))
ests3 <- pool(with(mice(nhanes, printFlag = FALSE, seed = 3), lm(bmi ~ age + hyp + chl)))
ests4 <- pool(with(mice(nhanes, printFlag = FALSE, seed = 4), lm(bmi ~ age + hyp + chl)))
ests5 <- pool(with(mice(nhanes, printFlag = FALSE, seed = 5), lm(bmi ~ age + hyp + chl)))
ests6 <- pool(with(mice(nhanes, printFlag = FALSE, seed = 6), lm(bmi ~ age + hyp + chl)))

ests2$pooled[, c(1, 3, 10)]
```

```
##          term   estimate    lambda
## 1 (Intercept) 19.9464142 0.4144454
## 2         age -4.0615093 0.4033924
## 3         hyp  1.5304762 0.1430995
## 4         chl  0.0628349 0.2959966
```

```
ests3$pooled[, c(1, 3, 10)]
```

```
##          term    estimate    lambda
## 1 (Intercept) 20.55844343 0.2772900
## 2         age -3.85753338 0.5895051
## 3         hyp  1.35281238 0.4101152
## 4         chl  0.05872834 0.5621346
```

```
ests4$pooled[, c(1, 3, 10)]
```

```
##          term    estimate    lambda
## 1 (Intercept) 19.39540373 0.1315114
## 2         age -3.50603350 0.2189333
## 3         hyp  2.75053046 0.1961083
## 4         chl  0.04920611 0.3305334
```

```
ests5$pooled[, c(1, 3, 10)]
```

```
##          term    estimate    lambda
## 1 (Intercept) 19.17135935 0.4855733
## 2         age -3.49672250 0.4511896
## 3         hyp  1.50954775 0.5942866
## 4         chl  0.06081272 0.2346065
```

```
ests6$pooled[, c(1, 3, 10)]
```

```
##          term    estimate    lambda
## 1 (Intercept) 20.52083805 0.4168136
## 2         age -2.92141353 0.6549523
## 3         hyp  1.22474596 0.2960364
## 4         chl  0.04949218 0.5196295
```

From these results we can see that λ changes significantly as the seed changes. The conclusions made for `seed`$= 1$ are not the same as the conclusions made for the other seeds. When `seed` $= 4$, the parameter that is most affected by the nonresponse is `chl`, and when `seed` $= 5$, the parameter that is most affected by the nonresponse is `hyp`. For `seed` $= 3$ and `seed` $= 3$, the parameter `age` is still the parameter most affected by the nonresponse. When `seed` $= 2$ the `Intercept` is the parameter most affeced by the nonresponse.

d)

The conclusions from the previous question suggest that we should increase the number of datasets M. Therefore we repeat the analysis with $M = 100$.

```
#using M=100 and changing the seed
ests1m100 <- pool(with(mice(nhanes, M=100, printFlag = FALSE, seed = 1),
                       lm(bmi ~ age + hyp + chl)))
ests2m100 <- pool(with(mice(nhanes, M=100, printFlag = FALSE, seed = 2),
                       lm(bmi ~ age + hyp + chl)))
ests3m100 <- pool(with(mice(nhanes, M=100, printFlag = FALSE, seed = 3),
                       lm(bmi ~ age + hyp + chl)))
ests4m100 <- pool(with(mice(nhanes, M=100, printFlag = FALSE, seed = 4),
                       lm(bmi ~ age + hyp + chl)))
ests5m100 <- pool(with(mice(nhanes, M=50, printFlag = FALSE, seed = 5),
                       lm(bmi ~ age + hyp + chl)))
ests6m100 <- pool(with(mice(nhanes, M=100, printFlag = FALSE, seed = 6),
                       lm(bmi ~ age + hyp + chl)))

ests1m100$pooled[, c(1, 3, 10)]
```

```
##          term    estimate     lambda
## 1 (Intercept) 19.61789252 0.08938989
## 2         age -3.55287155 0.68640637
## 3         hyp  2.19701748 0.35043452
## 4         chl  0.05378081 0.30408063
```

```
ests2m100$pooled[, c(1, 3, 10)]
```

```
##          term   estimate    lambda
## 1 (Intercept) 19.9464142 0.4144454
## 2         age -4.0615093 0.4033924
## 3         hyp  1.5304762 0.1430995
## 4         chl  0.0628349 0.2959966
```

```
ests3m100$pooled[, c(1, 3, 10)]
```

```
##          term    estimate    lambda
## 1 (Intercept) 20.55844343 0.2772900
## 2         age -3.85753338 0.5895051
## 3         hyp  1.35281238 0.4101152
## 4         chl  0.05872834 0.5621346
```

```
ests4m100$pooled[, c(1, 3, 10)]
```

```
##          term    estimate    lambda
## 1 (Intercept) 19.39540373 0.1315114
## 2         age -3.50603350 0.2189333
## 3         hyp  2.75053046 0.1961083
## 4         chl  0.04920611 0.3305334
```

```
ests5m100$pooled[, c(1, 3, 10)]
```

```
##          term    estimate    lambda
## 1 (Intercept) 19.17135935 0.4855733
## 2         age -3.49672250 0.4511896
## 3         hyp  1.50954775 0.5942866
## 4         chl  0.06081272 0.2346065
```

```
ests6m100$pooled[, c(1, 3, 10)]
```

```
##          term    estimate    lambda
## 1 (Intercept) 20.52083805 0.4168136
## 2         age -2.92141353 0.6549523
## 3         hyp  1.22474596 0.2960364
## 4         chl  0.04949218 0.5196295
```

The (pooled) estimates should get more stable as M increases so we can be more confident in any one specific run. Therefore, I would prefer these analyses over $M = 5$.

Question 2

The file `dataex2` consists of 100 datasets that were generated in the following way

$$y_i|x_i \overset{\text{iid}}{\sim} \text{N}(\beta_0 + \beta_1 x_i, 1) \quad x_i \overset{\text{iid}}{\sim} \text{Unif}(-1, 1) \quad \beta_0 = 1, \quad \beta_1 = 3$$

for $i = 1, ..., 100$. Additionally some of the responses have been set to missing from the datasets under the MAR mechanism.

```
# load the data
load("dataex2.Rdata")
# store the number of datasets as n
n <- dim(dataex2)[1]
```

Stochastic Regression Imputation

```
# initialize a counter
count <- 0
for (i in 1:n) {
  #impute values for the ith dataset using M=20
  imps_sri <- mice(dataex2[, , i], m = 20, method="norm.nob", printFlag = FALSE, seed = 1)
  fits_sri <- with(imps_sri, lm(Y ~ X)) #step 2
  ests_sri <- pool(fits_sri) # step 3
  summary_sri <- summary(ests_sri, conf.int = TRUE)
  if (summary_sri[2, c(7)] <= 3 & summary_sri[2, c(8)] >= 3) {
    count <- count + 1 #add to the counter if the the value of beta1 is contained in the
                       #confidence interval
  }
}

ecp_sri <- count/n
ecp_sri
```

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

Bootstrap version

```r
# initialize a counter
count <- 0
for (i in 1:n) {
  #impute values for the ith dataset, using m=20
  imps_boot <- mice(dataex2[,,i], m = 20, method="norm.boot", printFlag = FALSE, seed = 1)
  fits_sri <- with(imps_boot, lm(Y ~ X)) #step 2
  ests_boot <- pool(fits_sri) # step 3
  summary_boot <- summary(ests_boot, conf.int = TRUE)
  if (summary_boot[2, c(7)] <= 3 & summary_boot[2, c(8)] >= 3) {
    count = count + 1 #add to the counter if the true value of beta1 is contained in the
                      #confidence interval
  }
}

ecp_boot <- count/n
ecp_boot
```

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

Using stochastic regression imputation we get an empirical coverage probability of 88%. However when using the Bootstrap method we get an empirical coverage probability of 95%. The bootstrap method implements (normal linear) stochastic regression but takes parameter uncertainty into account. However the stochastic regression method implements (normal linear) stochastic regression for each copy m of the dataset we have made $m = 1, ..., 20$. Therefore parameter uncertainty is not taken into account since the same estimates are used for imputing all 20 copies of the dataset. Therefore, the confidence intervals may be too narrow and affect the coverage of the intervals.

Question 3

Given a dataset $\{y_i, x_{1i}, ..., x_{pi}\}$. We can consider a linear (in the coefficients) regression model

$$y_i = \beta_0 + \beta_1 x_{1i} + ... + \beta_p x_{pi} + \epsilon_i, \qquad \epsilon_i \sim \mathrm{N}(0, \sigma^2).$$

For strategy (i), we first compute the predicted values from each fitted model in step 2,

$$\hat{y}_i^{(m)} = \hat{\beta}_0^{(m)} + \hat{\beta}_1^{(m)} x_{1i} + ... + \hat{\beta}_p^{(m)} x_{pi},$$

where $\hat{\beta}_j^{(m)}$ is the estimate of β_j, $j = (0, ..., p)$, for the m^{th} dataset. Then pool them according to Rubin's rule for point estimates,

$$\begin{aligned}
\bar{y}_i &= \frac{1}{M} \sum_{i=1}^{M} \hat{y}_i^{(m)} \\
&= \frac{1}{M} \hat{\beta}_0^{(m)} + \hat{\beta}_1^{(m)} x_{1i} + ... + \hat{\beta}_p^{(m)} x_{pi} \\
&= \frac{1}{M} \sum_{i=1}^{M} \hat{\beta}_0^{(m)} + \frac{1}{M} \sum_{i=1}^{M} \hat{\beta}_1^{(m)} x_{1i} + ... + \frac{1}{M} \sum_{i=1}^{M} \hat{\beta}_p^{(m)} x_{pi} \\
&= \bar{\beta}_0 + \bar{\beta}_1 x_{1i} + ... + \bar{\beta}_p x_{pi}
\end{aligned}$$

Using strategy (ii), first we need to pool the regression coefficients from each fitted model in step 2 using Rubin's rule for point estimates,

$$\begin{aligned}
\bar{\beta}_0 &= \frac{1}{M}\sum_{m=1}^{M}\hat{\beta}_0^{(m)} \\
&\vdots \\
\bar{\beta}_p &= \frac{1}{M}\sum_{m=1}^{M}\hat{\beta}_p^{(m)}.
\end{aligned}$$

Then we can compute the predicted values,

$$\bar{y}_i = \bar{\beta}_0 + \bar{\beta}_1 x_{1i} + ... + \bar{\beta}_p x_{pi}$$

These two equations show that computing the predicted values from each fitted model in step 2 and then pooling them according to Rubin's rule for point estimates is the mathematically equivalent to pooling the regression coefficients from each fitted model in step 2 using Rubin's rule for point estimates and then computing the predicted values afterwards.

Question 4

The model used to generate the data, which corresponds to our model of interest in step 2, was the following one:

$$\begin{gathered}
y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 x_{1i}x_{2i} + \epsilon_i, \\
x_{1i} \overset{\text{iid}}{\sim} \text{N}(0,1), \qquad x_{2i} \overset{\text{iid}}{\sim} \text{N}(1.5,1), \qquad \epsilon_i \overset{\text{iid}}{\sim} \text{N}(0,1),
\end{gathered}$$

for $i = 1, ..., 1000, \beta_0 = 1.5, \beta_1 = 1, \beta_2 = 2, \beta_3 = 1$. Missingness has been imposed on y and x_1.

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

a) Impute, then transform

By only imputing the y and $x1$ variables in step 1, we can calculate estimates of β_1, β_2 and β_3, with the 95% confidence intervals. In this approach the interaction variable is left outside the imputation process and calculated afterwards in the analysis model.

```
imps1 <- mice(dataex4, m=50, seed=1, printFlag = FALSE)
fits1 <- with(imps1, lm(y ~ x1 + x2 + x1*x2))
ests1 <- pool(fits1)
summary(ests, conf.int = TRUE)[, c(1,2,3,7,8)]
```

```
##          term    estimate  std.error        2.5 %      97.5 %
## 1 (Intercept) 19.61789252 3.41003531 12.421281424 26.81450361
## 2         age -3.55287155 1.54113006 -8.067205920  0.96146282
## 3         hyp  2.19701748 2.10797844 -2.568707058  6.96274202
## 4         chl  0.05378081 0.02031792  0.008646559  0.09891506
```

This shows that the estimate of β_2 is approximately true. However the estimates of β_1 and β_3 are not very good and the true values do not lie in the 95% confidence intervals.

b) Passive imputation

Another method is passive imputation. We start by calculating the interaction variable $x_{3i} = x_{1i}x_{2i}$ in the incomplete data and appending it as a variable to the dataset. Then we will use passive imputation to impute the interaction variable.

```r
x1 <- dataex4$x1; x2 <- dataex4$x2; dataex4$x3 <- x1*x2
imp0 <- mice(dataex4, maxit=0)

meth <- imp0$method
meth["x3"] <- "~I(x1*x2)"

pred <- imp0$predictorMatrix
pred[c("x1", "x2"), "x3"] <- 0

visSeq <- imp0$visitSequence
visSeq
```

```
## [1] "y"  "x1" "x2" "x3"
```

```r
imps2 <- mice(dataex4, method = meth, predictorMatrix = pred, visitSequence = visSeq,
              m = 50, seed = 1, printFlag = FALSE)

ests2 <- pool(with(imps2, lm(y ~ x1 + x2 + x1*x2)))

summary(ests2, conf.int=TRUE)[,c(1,2,3,7,8)]
```

```
##          term  estimate  std.error     2.5 %    97.5 %
## 1 (Intercept) 1.5534782 0.08842211 1.3788626 1.7280939
## 2          x1 1.1926170 0.09584345 1.0034980 1.3817360
## 3          x2 1.9964402 0.04936582 1.8989468 2.0939336
## 4       x1:x2 0.8740573 0.05678521 0.7615712 0.9865434
```

The estimates have improved slightly for β_1, β_2 and β_3, however the true values of β_1 and β_3 still do not lie in the 95% confidence intervals.

c) Just another variable

Additionally, now that the interaction variable has been appended to the dataset, we can impute it just like another variable and use this variable for the interaction term in step 2.

```r
imps3 <- mice(dataex4, m = 50, seed = 1, printFlag = FALSE)
fits3 <- with(imps3, lm(y ~ x1 + x2 + x3))
ests3 <- pool(fits3)
summary(ests3, conf.int=TRUE)[, c(1,2,3,7,8)]
```

```
##          term estimate  std.error     2.5 %   97.5 %
## 1 (Intercept) 1.499714 0.07821436 1.3452011 1.654227
## 2          x1 1.003930 0.08228372 0.8414967 1.166363
## 3          x2 2.026180 0.04371605 1.9398113 2.112548
## 4          x3 1.017793 0.04428071 0.9303479 1.105238
```

This method has improved the estimates greatly. All the estimates are approximately equal to the true values of the parameters and they all lie in the 95% confidence intervals.

d)

The most obvious conceptual drawback of the *just another variable* approach for imputing interactions is that because x_3 is the last column it is imputed using the x_1 and x_2 from the original dataset.

Question 5

The dataset `NHANES2.Rdata` is a subset of data from the *National Health and Nutrition Examination Survey* (NHANES), where the goal is to assess the health and nutrition status of adults and children in the United States. The variables in the dataset are the following:

- `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; binary,
- `smoke`: smoking status; 3 ordered categories,
- `DM`: diabetes mellitus status; binary,
- `WC`: waist circumference in cm.

The analysis of interest is the linear model:

$$\text{wgt} = \beta_0 + \beta_1\text{gender} + \beta_2\text{age} + \beta_3\text{hgt} + \beta_4\text{WC} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2). \tag{1}$$

We can start by looking at the data, and by using the command `dim`, we see there are 500 rows (individuals), and 12 variables.

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

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

Using the command `str` we can inspect the nature of our variables and check they are coded correctly.

```r
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 ...
```

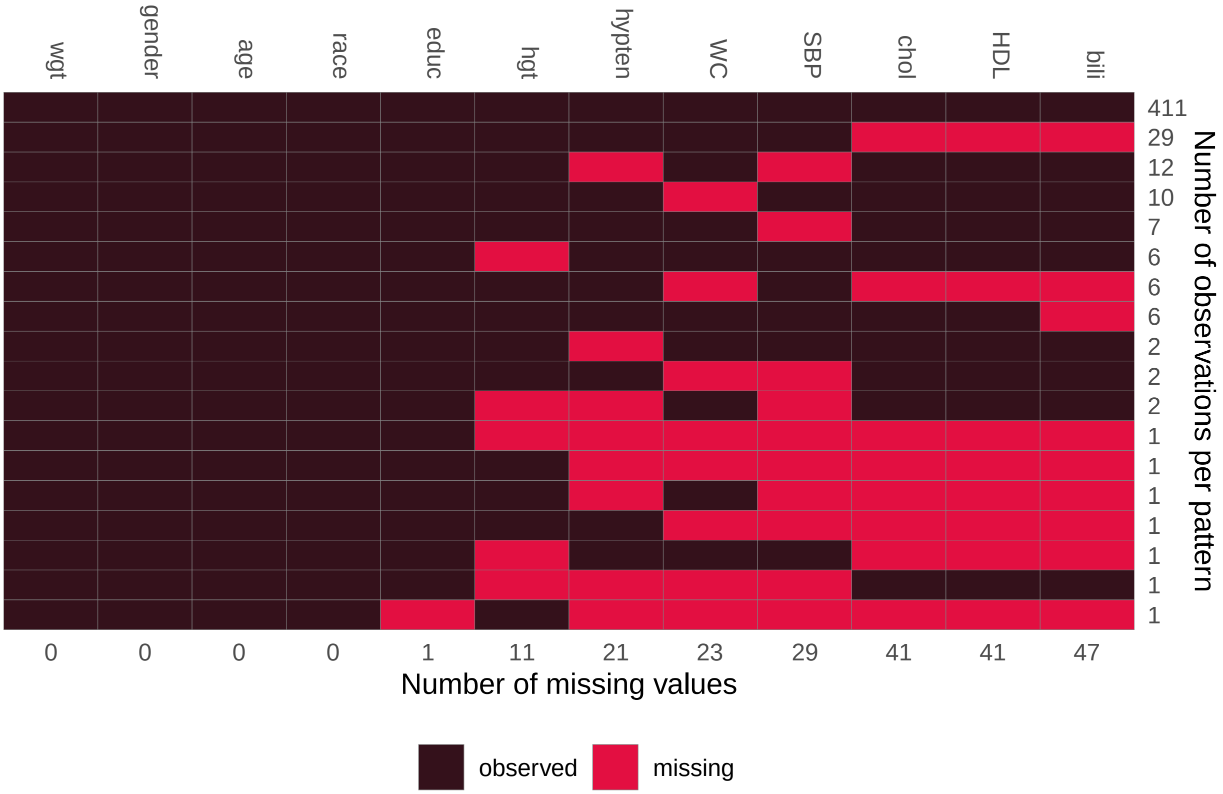
Additionally, by using the `summary` command we can get an idea of the min/max/mean/quantiles of the observed data in each variable, as well as the number of missing values.

```
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
```

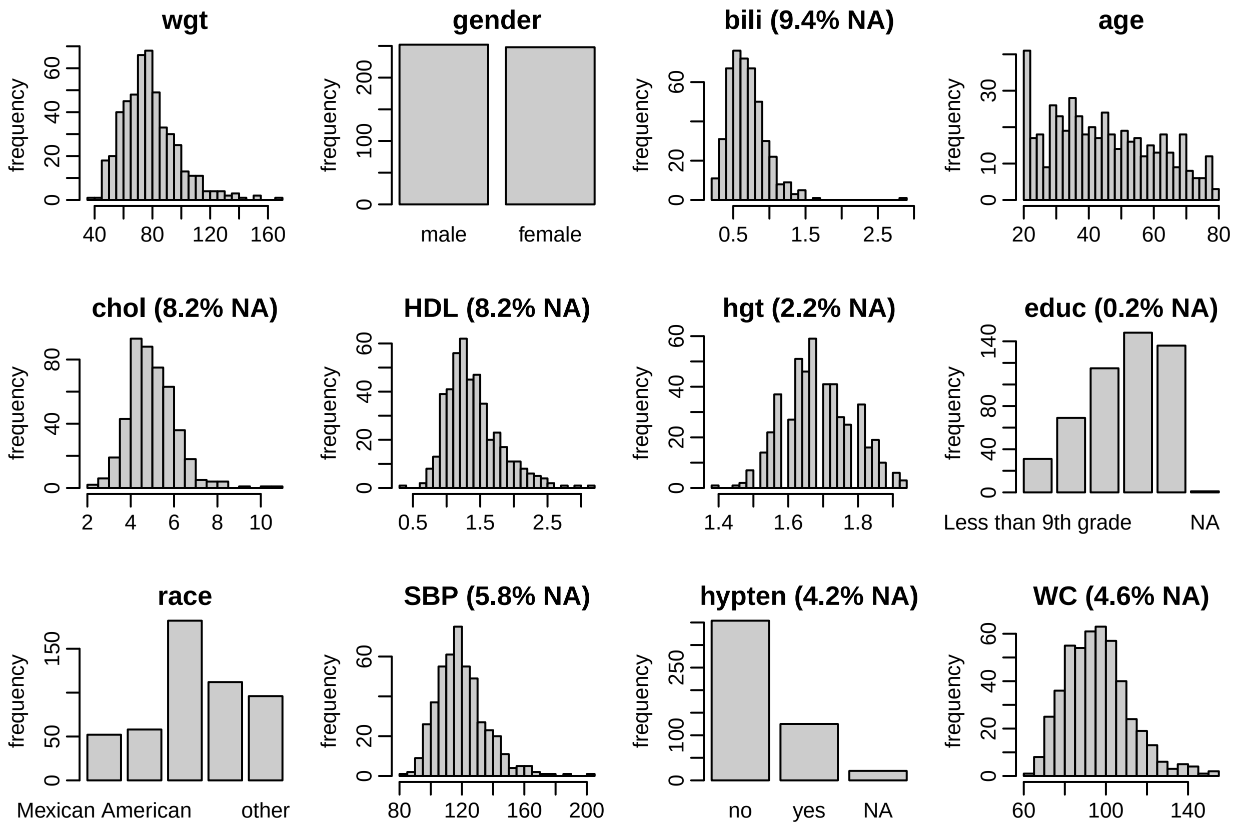
I have used the `JointAI` package to inspect the missing data patterns. `JointAI` performs (Bayesian) multiple imputation and has useful visualisation functions.

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



Predictive mean matching is the default of `mice` for continuous variables. Since we want to use a normal linear regression model for imputing the missing values, we need to inspect whether the normality assumption is approximately met. The package `JointAI` allows us to visualise 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 are now ready to start the imputation procedure. I will start by doing a dry run of `mice()`, without any iterations, which will create the default versions of everything that needs to be specified. These default settings can then be adapted to this dataset.

```
imp0 <- mice(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
```

By assessing the plots, using a normal distribution for the `hgt` is not an unreasonable idea. Let us change the default imputation method from `pmm` to `norm` for the variable `hgt`.

```
meth <- imp0$method
meth["hgt"] <- "norm"
meth
```

```
##      wgt   gender     bili      age     chol      HDL      hgt     educ 
##       ""       ""    "pmm"       ""    "pmm"    "pmm"   "norm"   "polr" 
##     race      SBP   hypten       WC 
##       ""    "pmm" "logreg"    "pmm"
```

We do not want to impute negative values of height so we can use the function `post()` to specify functions that modify the imputed values. With the below syntax all imputed values of `hgt` that are outside the interval $(1.35, 1.95)$ will be set to those limiting values.

```
post <- imp0$post
post["hgt"] <- "imp[[j]][,i] <- squeeze(imp[[j]][,i], c(1.35, 1.95))"
```

We can now begin the imputation.

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

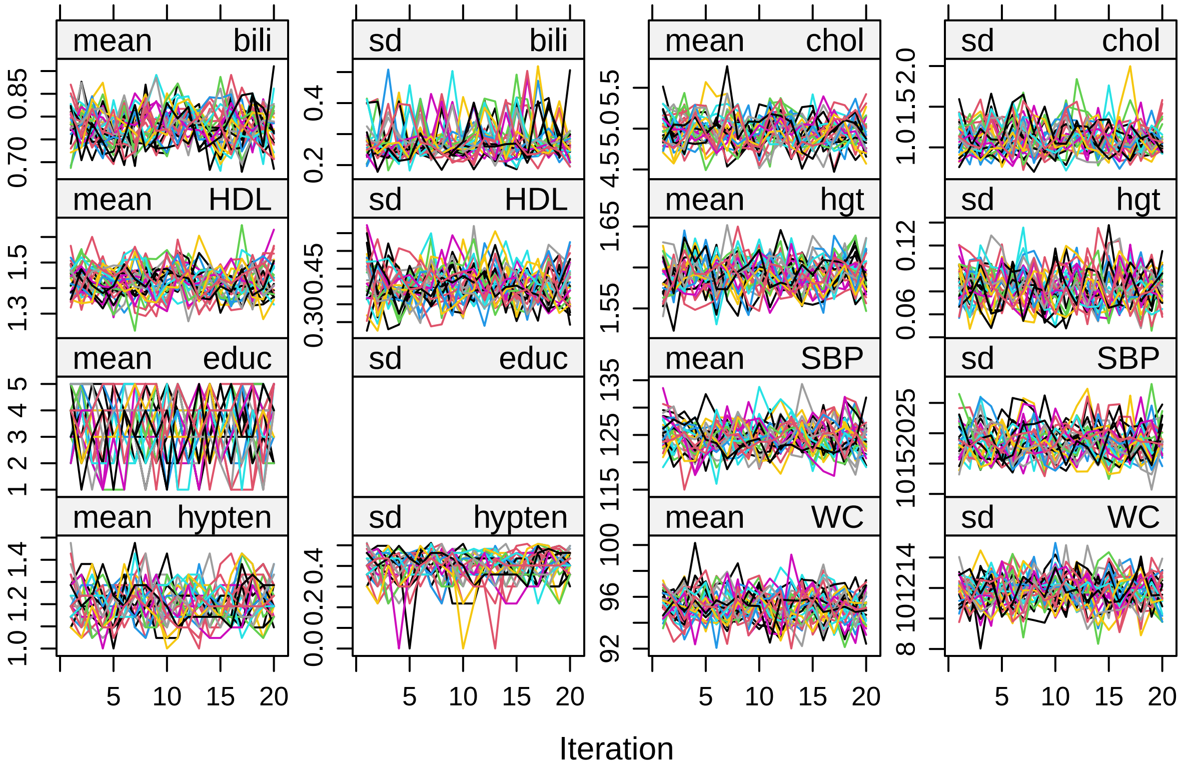
`mice()` does some pre-processing and removes incomplete variables that are not imputed but act as predictors in other imputation models, it also removes constant variables and variable that are collinear. Checking the `loggedEvents` contained in our object `imp` allows us to know if `mice()` detected any problems during the imputation.

```
imp$loggedEvents
```

```
## NULL
```

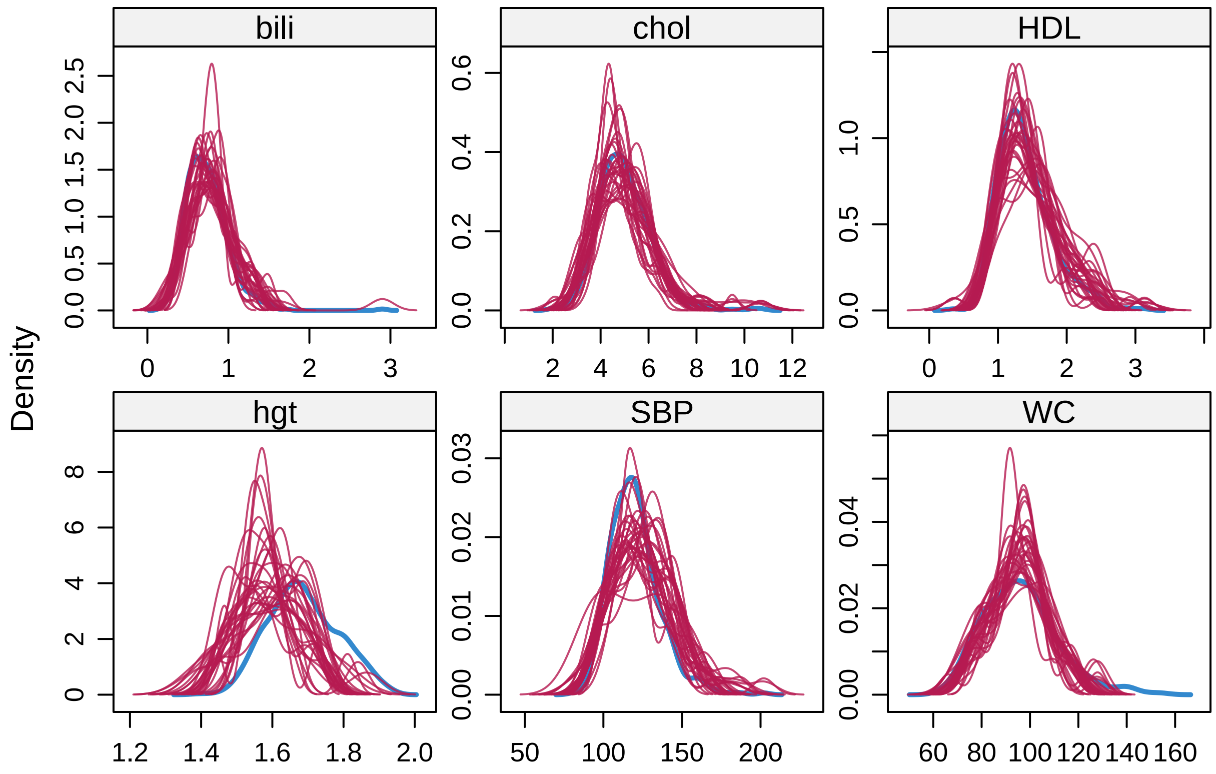
The mean and variance of the imputed values and variable are stored in the elements `chainMean` and `chainVar` of the `mids` object `imp`. To check that the `mice()` algorithm has converged, we can plot our object and visualize the trace plots. If the algorithm hasn't converged then there is no guarantee that the results obtained are correct.

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



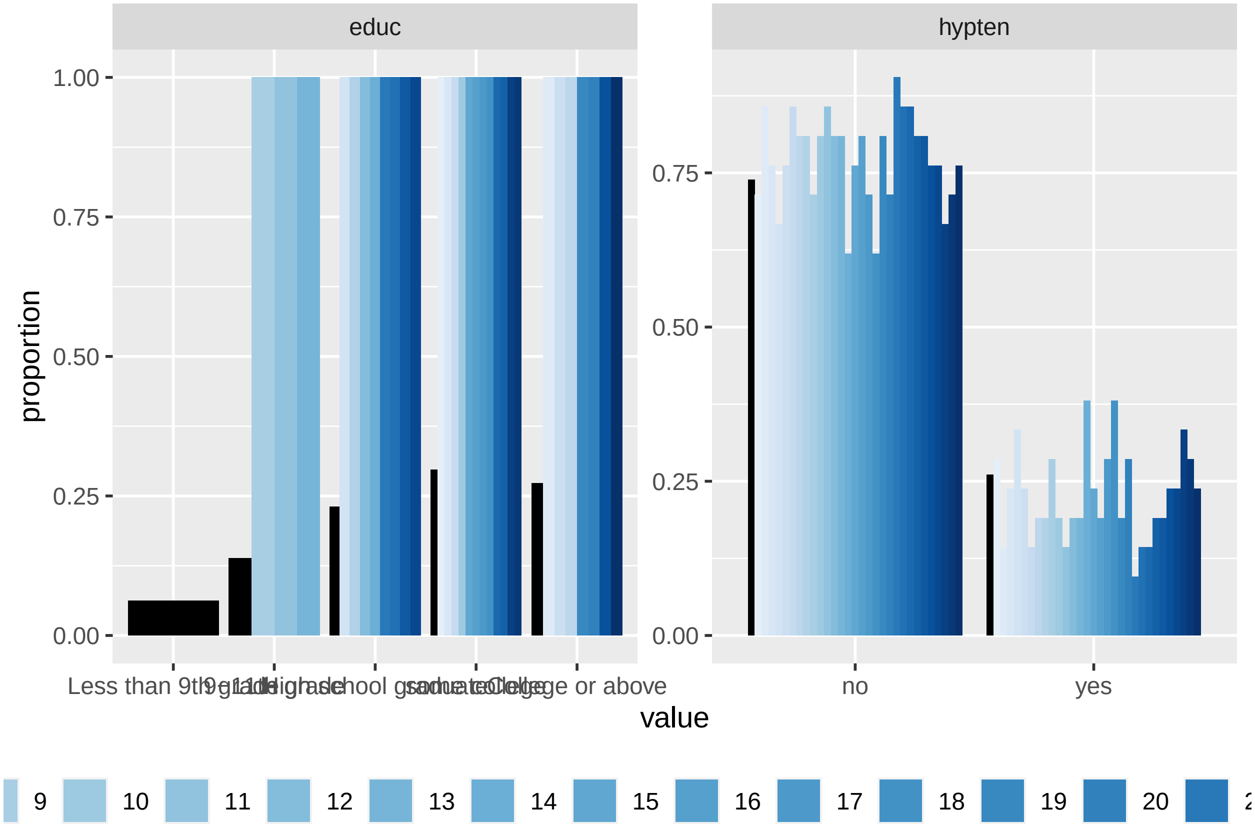
We can see that the iterative algorithm appears to have converged for all variables that were imputed. We can compare the distribution of the imputed values against the distribution of the observed values. For the continuous variables we can use the command `densityplot()`.

```
densityplot(imp)
```



We used $M = 30$, and since the density of the observed values are plotted first (blue line), it is difficult to see. The plot which are most different are the ones for `SBP` and `hgt`. With regard to categorical variables, we can compare the proportion of values in each category. We can use `propplot`, implemented by Nicole Erler and available on her github.

```
propplot(imp)
```

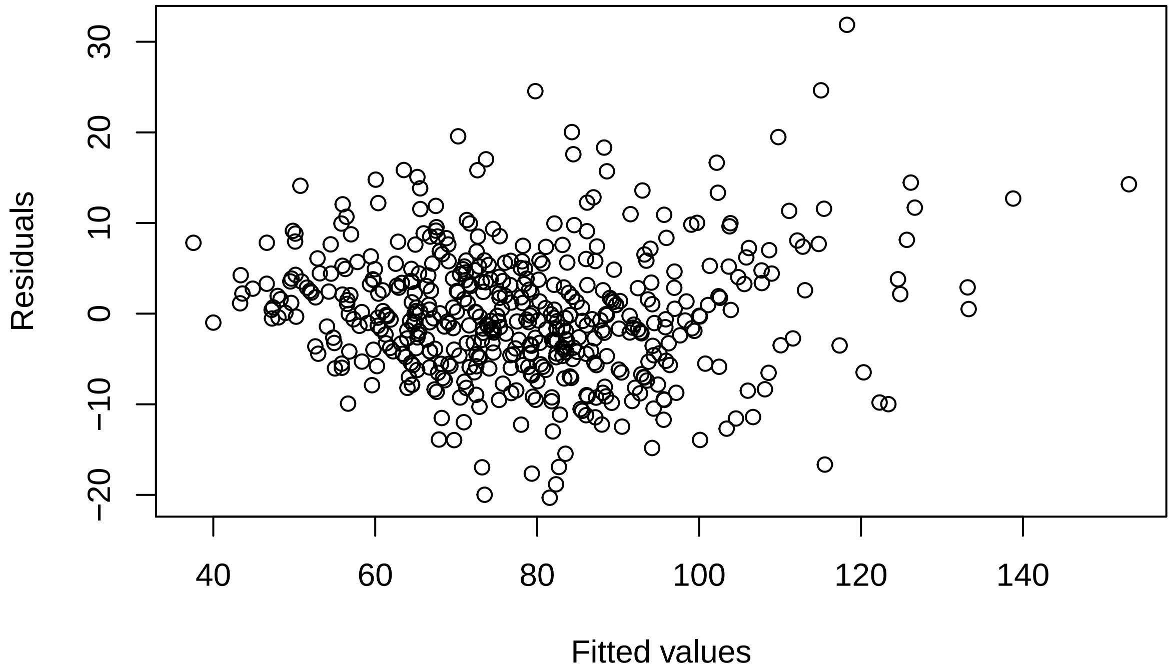


This shows a large discrepancy between the observed and imputed data distributions for the educ variables. However there is only 0.2% of data missing for this variable.

The imputation step was successful, so we can continue to the analysis of the imputed data. The model of interest is (1).

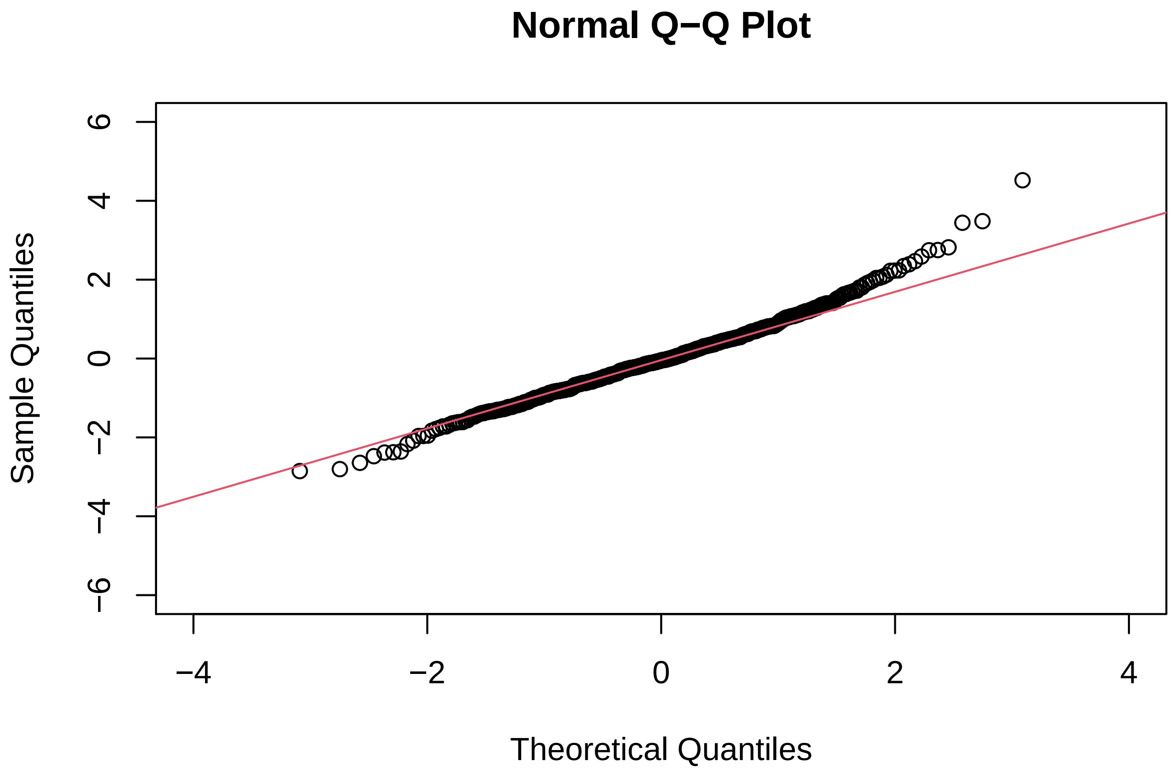
```r
fit <- with(imp, lm(wgt ~ gender + age + hgt + WC))
```

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



Doing a QQplot, we can see that nothing looks suspicious.

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



Now we can pool the results.

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

```
##           term     estimate  std.error  statistic       df      p.value
## 1  (Intercept) -100.6455031 7.61471595 -13.217237 460.5355 0.000000e+00
## 2 genderfemale   -1.3873860 0.83978725  -1.652068 458.3258 9.920548e-02
## 3          age   -0.1576518 0.02159588  -7.300087 433.7302 1.384448e-12
## 4          hgt   52.3207670 4.37203661  11.967138 450.7310 0.000000e+00
## 5           WC    1.0258588 0.02237817  45.841943 478.8476 0.000000e+00
##          2.5 %      97.5 %
## 1 -115.6093979 -85.6816083
## 2   -3.0376968   0.2629248
## 3   -0.2000974  -0.1152062
## 4   43.7286611  60.9128729
## 5    0.9818873   1.0698304
```

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

```
##                est     lo 95     hi 95 fmi
## adj R^2 0.8557788 0.8300365 0.8779047 NaN
```