# Incomplete Data Analysis

## Multiple imputation and multivariate missingness: the `mice` package

School of Mathematics, University of Edinburgh

V. Inácio de Carvalho & M. de Carvalho

To illustrate how to perform multiple imputation with `mice` when several variables have missing values we will use a subset of the *National Health and Nutrition Examination Survey* (NHANES), whose goal is to assess the health and nutritional status of adults and children in the United States and to track changes over time.

The dataset is available in the file `NHANES.Rdata` in Learn. There are 18 variables available:

- `race`: 5 unordered categories,
- `DBP`: diastolic blood pressure in mmHg,
- `bili`: bilirubin concentration in mg/dL,
- `smoke`: smoking status; 3 ordered categories,
- `DM`: diabetes mellitus status; binary,
- `gender`: `male` vs `female`,
- `chol`: total serum cholesterol in mg/dL,
- `alc`: weekly alcohol consumption; 5 ordered categories,
- `SBP`: systolic blood pressure in mmHg,
- `wgt`: weight in kg,
- `hypten`: hypertensive; binary,
- `occup`: occupational status; 3 unordered categories,
- `age`: in years,
- `albu`: albumin concentration in g/dL,
- `creat`: creatinine concentration in mg/dL,
- `uricacid`: uric acid concentration in mg/dL,
- `BMI`: body mass index in kg/m$^2$,
- `hgt`: height in metres.

Depending on the research question, for the NHANES data we might use

- linear regression using the `lm()` function,
- logistic regression using the `glm()` function with `family = binomial()`.

For instance, we might be interested in the following linear regression model

$$\text{SBP} = \beta_0 + \beta_1\text{DM} + \beta_2\text{age} + \beta_3\text{BMI} + \beta_4\text{hypten} + \varepsilon, \quad \varepsilon \sim \text{N}(0, \sigma^2). \tag{1}$$

We start by taking a first look at the data, and by using the command `dim` we see that there are 1000 rows, which in this particular example represent individuals, and 18 variables.

```r
load("NHANES.RData")
dim(NHANES)
```

```
## [1] 1000   18
```

We can further inspect the nature of our variables and check they are correctly coded.

```
str(NHANES)
```

```
## 'data.frame':    1000 obs. of  18 variables:
##  $ race    : Factor w/ 5 levels "Mexican American",..: 2 5 3 3 3 3 2 3 1 3 ...
##  $ DBP     : num  56.7 81.3 70 66 69.3 ...
##  $ bili    : num  0.5 0.9 0.7 0.4 0.9 NA NA 0.9 0.6 0.9 ...
##  $ smoke   : Ord.factor w/ 3 levels "never"<"former"<..: 1 1 3 1 3 2 1 2 2 3 ...
##  $ DM      : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 2 1 1 1 ...
##  $ gender  : Factor w/ 2 levels "male","female": 2 1 1 2 1 2 1 1 1 1 ...
##  $ chol    : num  4.29 4.27 4.22 3.96 4.97 NA NA 5.2 5.56 4.86 ...
##  $ alc     : Ord.factor w/ 5 levels "0"<"<=1"<"1-3"<..: 2 NA NA 2 4 1 1 5 3 4 ...
##  $ SBP     : num  105 125 133 141 117 ...
##  $ wgt     : num  46.3 63.5 62.1 113.9 102.1 ...
##  $ hypten  : Factor w/ 2 levels "no","yes": 1 1 1 2 1 2 2 1 1 1 ...
##  $ occup   : Factor w/ 3 levels "working","looking for work",..: 1 1 1 1 1 3 3 1 1 1 ...
##  $ age     : num  22 39 51 45 31 75 73 52 29 40 ...
##  $ albu    : num  3.8 4.3 4.3 3.6 4.9 NA NA 3.9 4.9 4.3 ...
##  $ creat   : num  0.61 0.87 0.89 0.61 0.83 NA NA 0.91 0.93 0.94 ...
##  $ uricacid: num  3.6 5.4 6.2 4.3 6.1 NA NA 7.8 3.9 4.9 ...
##  $ BMI     : num  19.3 19.5 22.1 41.8 32.3 ...
##  $ hgt     : num  1.55 1.8 1.68 1.65 1.78 ...
```

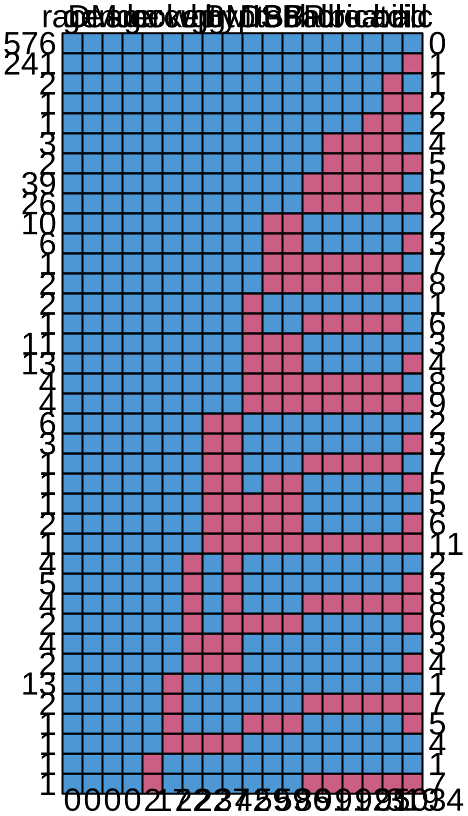
Also, by using the command summary we can have a quick idea about min/max/mean/quantiles of the observed data in each variable along with the number of missing values.

```
summary(NHANES)
```

```
##                   race          DBP              bili            smoke
##  Mexican American  :112   Min.   : 12.67   Min.   :0.2000   never  :608
##  Other Hispanic    :110   1st Qu.: 64.00   1st Qu.:0.6000   former :191
##  Non-Hispanic White:350   Median : 70.67   Median :0.7000   current:199
##  Non-Hispanic Black:229   Mean   : 70.80   Mean   :0.7527   NA's   :  2
##  other             :199   3rd Qu.: 77.33   3rd Qu.:0.9000
##                           Max.   :130.00   Max.   :2.9000
##                           NA's   :59       NA's   :95
##    DM         gender         chol          alc           SBP
##  no :853   male  :493   Min.   : 2.070   0   :113   Min.   : 81.33
##  yes:147   female:507   1st Qu.: 4.270   <=1 :281   1st Qu.:109.33
##                         Median : 4.910   1-3 :105   Median :118.00
##                         Mean   : 4.998   3-7 : 81   Mean   :120.15
##                         3rd Qu.: 5.610   >7  :101   3rd Qu.:128.67
##                         Max.   :10.680   NA's:319   Max.   :202.00
##                         NA's   :86                  NA's   :59
##      wgt          hypten                 occup          age
##  Min.   : 39.01   no  :693   working         :544   Min.   :20.00
##  1st Qu.: 63.50   yes :265   looking for work: 46   1st Qu.:31.00
##  Median : 76.88   NA's: 42   not working     :393   Median :43.00
##  Mean   : 78.35              NA's            : 17   Mean   :45.23
##  3rd Qu.: 89.13                                     3rd Qu.:59.00
##  Max.   :167.83                                     Max.   :79.00
##  NA's   :22
##       albu           creat           uricacid          BMI
##  Min.   :3.000   Min.   :0.4400   Min.   :2.300   Min.   :15.34
##  1st Qu.:4.100   1st Qu.:0.6900   1st Qu.:4.400   1st Qu.:23.18
##  Median :4.300   Median :0.8200   Median :5.300   Median :26.58
##  Mean   :4.289   Mean   :0.8525   Mean   :5.356   Mean   :27.49
##  3rd Qu.:4.500   3rd Qu.:0.9500   3rd Qu.:6.200   3rd Qu.:30.73
##  Max.   :5.400   Max.   :7.4600   Max.   :9.900   Max.   :60.54
##  NA's   :91      NA's   :91       NA's   :92      NA's   :37
##       hgt
##  Min.   :1.397
##  1st Qu.:1.626
##  Median :1.676
##  Mean   :1.685
##  3rd Qu.:1.753
##  Max.   :1.981
##  NA's   :22
```

We should also inspect the missing data patterns. We can use, for instance, the `md.pattern` function from `mice`, although in this case due to the large number of variables, it becomes difficult to extract meaningful information from it.

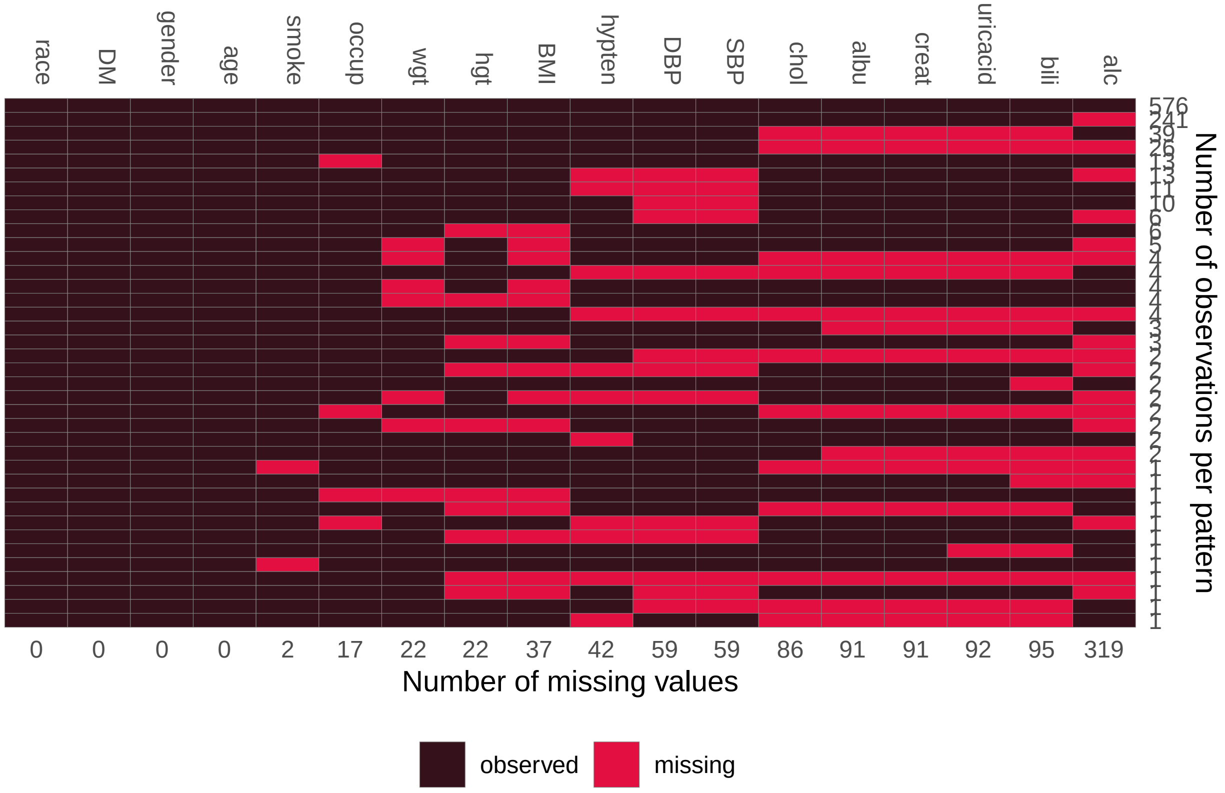
```
require(mice)
mdpat_mice <- md.pattern(NHANES)
```



```
mdpat_mice
```

We have also seen the packages `VIM` and `naniar` in week 2. I will also introduce now the `JointAI` package, which performs (Bayesian) multiple imputation and has also very useful visualisation functions.

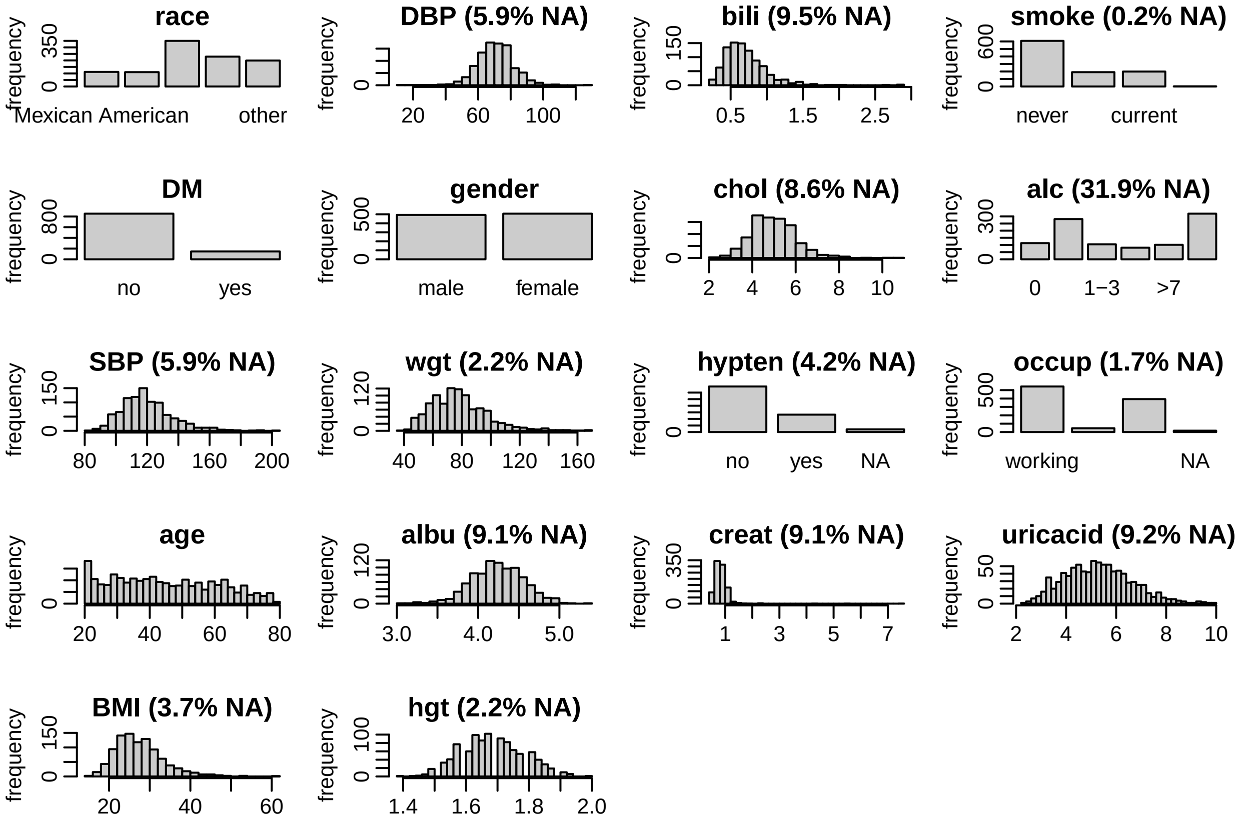
```
require(JointAI)
md_pattern(NHANES, pattern = FALSE, color = c('#34111b', '#e30f41'))
```



We can conclude, for instance, that there are 576 observations with observed values on all 18 variables. Also, 241 observations for which only the weekly alcohol consumption is missing, etc. As a further check, we can also look at the correlations between the different variables.

We know that predictive mean matching is the default of `mice` for continuous variables. In case we instead want for some reason to use a normal linear regression model for imputing the missing values, we should (informally) inspect whether the normality assumption is roughly met. The package `JointAI` allows us to visualise how the observed parts of the incomplete variables are distributed.

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



Having inspected the data, we are ready to start our imputation procedure. We will start by doing a setup or 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 our particular dataset.

```
imp0 <- mice(NHANES, maxit = 0)
imp0
```

```
## Class: mids
## Number of multiple imputations:  5
## Imputation methods:
##      race       DBP      bili     smoke        DM    gender      chol       alc
##        ""     "pmm"     "pmm"    "polr"        ""        ""     "pmm"    "polr"
##       SBP       wgt    hypten     occup       age      albu     creat  uricacid
##     "pmm"     "pmm"  "logreg" "polyreg"        ""     "pmm"     "pmm"     "pmm"
##       BMI       hgt
##     "pmm"     "pmm"
## PredictorMatrix:
##        race DBP bili smoke DM gender chol alc SBP wgt hypten occup age albu
## race      0   1    1     1  1      1    1   1   1   1      1     1   1    1
## DBP       1   0    1     1  1      1    1   1   1   1      1     1   1    1
## bili      1   1    0     1  1      1    1   1   1   1      1     1   1    1
## smoke     1   1    1     0  1      1    1   1   1   1      1     1   1    1
## DM        1   1    1     1  0      1    1   1   1   1      1     1   1    1
## gender    1   1    1     1  1      0    1   1   1   1      1     1   1    1
##        creat uricacid BMI hgt
## race       1        1   1   1
## DBP        1        1   1   1
## bili       1        1   1   1
## smoke      1        1   1   1
## DM         1        1   1   1
## gender     1        1   1   1
```

As a matter of example, in the previous plot depicting the distribution of the observed data for the different variables, we could appreciate that using a normal distribution for the DBP variable is possibly not a completely unreasonable idea (the same would apply, e.g., to the albumin variable). Let us then change the default imputation method from `pmm` to `norm` for the variable DBP.

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

```
##      race       DBP      bili     smoke        DM    gender      chol       alc
##        ""    "norm"     "pmm"    "polr"        ""        ""     "pmm"    "polr"
##       SBP       wgt    hypten     occup       age      albu     creat  uricacid
##     "pmm"     "pmm"  "logreg" "polyreg"        ""     "pmm"     "pmm"     "pmm"
##       BMI       hgt
##     "pmm"     "pmm"
```

However, we need to be careful, because we do not want to risk imputing a negative value for the diastolic blood pressure! Fortunately, the function `mice()` has the argument `post()` that can be used to specify functions that modify the imputed values. With the below syntax all imputed values of DBP that are outside the interval $(0, 180)$ will be set to those limiting values. Note that I am unsure whether the DBP can be in the 10–40 range or above 160, but I just want to illustrate how to create bounds for the imputed values.

```
post <- imp0$post
post["DBP"] <- "imp[[j]][,i] <- squeeze(imp[[j]][,i], c(0, 180))"
```

Further, we should note that the BMI is a deterministic function of the height and weight, BMI $=$ weight/`height`$^2$ and the three variables are in our dataset. If we impute the BMI directly, its values may be inconsistent with the imputed values of height and weight (none of the three variables is fully observed). Also, because there are cases where only one of these two variables is missing, to possibly gain some precision, we want to impute the height and the weight separately and BMI should then be calculated from the (imputed) values of these two variables. If BMI is not a relevant predictor in any of the other imputation models, we could just exclude BMI from the imputation and calculate it afterwards. To use BMI as a predictor in the other imputation models, it has to be calculated in each iteration of the algorithm, which in `mice()` is possible through the so-called passive imputation strategy, which requires us to specify a formula to calculate BMI through the `I()` operator.

```
meth["BMI"] <- "~I(wgt/(hgt^2))"
meth
```

```
##                race               DBP              bili             smoke
##                  ""            "norm"             "pmm"            "polr"
##                  DM            gender              chol               alc
##                  ""                ""             "pmm"            "polr"
##                 SBP               wgt            hypten             occup
##               "pmm"             "pmm"          "logreg"         "polyreg"
##                 age              albu             creat          uricacid
##                  ""             "pmm"             "pmm"             "pmm"
##                 BMI               hgt
## "~I(wgt/(hgt^2))"             "pmm"
```

To prevent feedback from BMI in the imputation of height and weight, the predictor matrix needs to be modified.

```
pred <- imp0$predictorMatrix
# BMI will not be used as predictor of height and weight
```

```
pred[c("hgt", "wgt"), "BMI"] <- 0
```

That is, BMI will not act as a predictor of in the height and weight imputation models. To avoid multicollinearity, which may lead to problems during imputation, imputation models should not include all the three variables as predictor variables. In this example, we will use BMI to impute the other variables and so we do the following change in the `predictorMatrix`

```
pred[, c("hgt", "wgt")] <- 0
```

However, we are not using the BMI to impute the height and the weight and so we want them to be included in the imputation model of each other.

```
pred["hgt", "wgt"] <- 1
pred["wgt", "hgt"] <- 1
pred
```

```
##          race DBP bili smoke DM gender chol alc SBP wgt hypten occup age albu
## race        0   1    1     1  1      1    1   1   1   0      1     1   1    1
## DBP         1   0    1     1  1      1    1   1   1   0      1     1   1    1
## bili        1   1    0     1  1      1    1   1   1   0      1     1   1    1
## smoke       1   1    1     0  1      1    1   1   1   0      1     1   1    1
## DM          1   1    1     1  0      1    1   1   1   0      1     1   1    1
## gender      1   1    1     1  1      0    1   1   1   0      1     1   1    1
## chol        1   1    1     1  1      1    0   1   1   0      1     1   1    1
## alc         1   1    1     1  1      1    1   0   1   0      1     1   1    1
## SBP         1   1    1     1  1      1    1   1   0   0      1     1   1    1
## wgt         1   1    1     1  1      1    1   1   1   0      1     1   1    1
## hypten      1   1    1     1  1      1    1   1   1   0      0     1   1    1
## occup       1   1    1     1  1      1    1   1   1   0      1     0   1    1
## age         1   1    1     1  1      1    1   1   1   0      1     1   0    1
## albu        1   1    1     1  1      1    1   1   1   0      1     1   1    0
## creat       1   1    1     1  1      1    1   1   1   0      1     1   1    1
## uricacid    1   1    1     1  1      1    1   1   1   0      1     1   1    1
## BMI         1   1    1     1  1      1    1   1   1   0      1     1   1    1
## hgt         1   1    1     1  1      1    1   1   1   1      1     1   1    1
##          creat uricacid BMI hgt
## race         1        1   1   0
## DBP          1        1   1   0
## bili         1        1   1   0
## smoke        1        1   1   0
## DM           1        1   1   0
## gender       1        1   1   0
## chol         1        1   1   0
## alc          1        1   1   0
## SBP          1        1   1   0
## wgt          1        1   0   1
## hypten       1        1   1   0
## occup        1        1   1   0
## age          1        1   1   0
## albu         1        1   1   0
## creat        0        1   1   0
## uricacid     1        0   1   0
## BMI          1        1   0   0
## hgt          1        1   0   0
```

Note that passive imputation overrules the selection of variables specified in the `predictorMatrix` argument.

Now, the sequence in which weight, height, and BMI values are imputed is important. By default, `mice()` imputes incomplete columns in the data from left to right. To be sure that the imputed values of BMI match the imputed values of height and weight at each iteration, BMI needs to be imputed after height and weight. Because BMI in our dataset appears before than height, we know that this is not the case and should be changed. Rather then reordering the dataset itself, it is more convenient to change the visiting scheme of the algorithm by the `visitSequence` argument of the function `mice()`.

```
visSeq <- imp0$visitSequence
visSeq
```

```
##  [1] "race"     "DBP"      "bili"     "smoke"    "DM"       "gender"
##  [7] "chol"     "alc"      "SBP"      "wgt"      "hypten"   "occup"
## [13] "age"      "albu"     "creat"    "uricacid" "BMI"      "hgt"
```

```
which_BMI <- match("BMI", visSeq)
visSeq <- c(visSeq[-which_BMI], visSeq[which_BMI])
visSeq
```

```
##  [1] "race"     "DBP"      "bili"     "smoke"    "DM"       "gender"
##  [7] "chol"     "alc"      "SBP"      "wgt"      "hypten"   "occup"
## [13] "age"      "albu"     "creat"    "uricacid" "hgt"      "BMI"
```

We are now good to go. I will be using $M = 30$ but sensitivity of our estimates/conclusions to this value should be checked. We also need to specify `maxit`, which is the number of iterations (we should not forget that this is an iterative procedure!). By default `mice()` uses `maxit=5`. I will be on the conservative side and set `maxit=20` (this will make step1 to take a while longer to be executed), which also facilitates checking convergence of the chains of imputed values (or better stated, their mean and standard deviation).

```
imp <- mice(NHANES, method = meth, predictorMatrix = pred, visitSequence = visSeq,
            maxit = 20, m = 30, seed = 1, printFlag = FALSE)
```

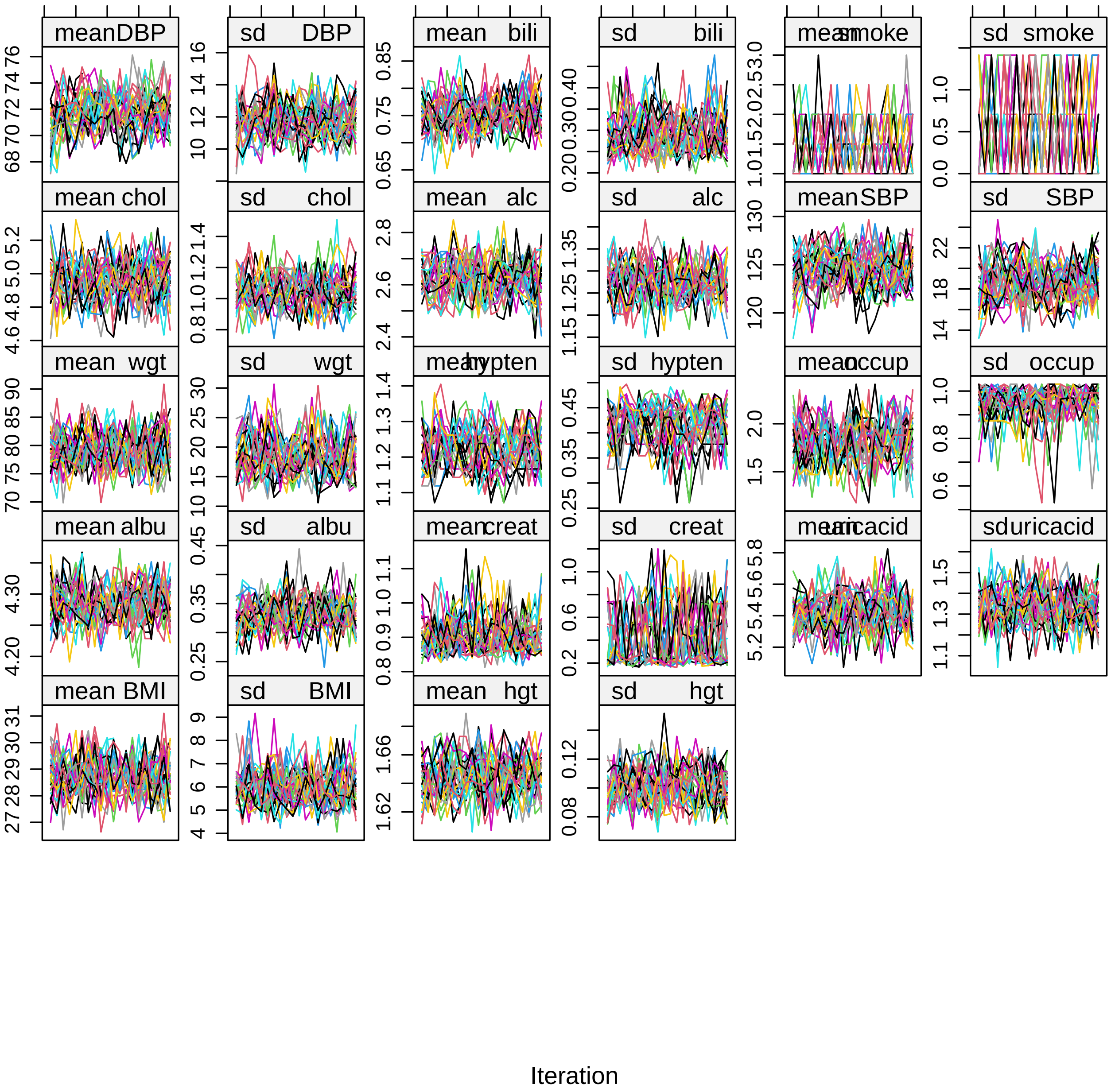
We should be aware that `mice()` does some pre-processing and will remove incomplete variables that are not imputed but act as predictors in other imputation models, it will also remove constant variables and variables that are collinear. Checking the `loggedEvents` contained in our object `imp` allows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

```
imp$loggedEvents
```

```
## NULL
```

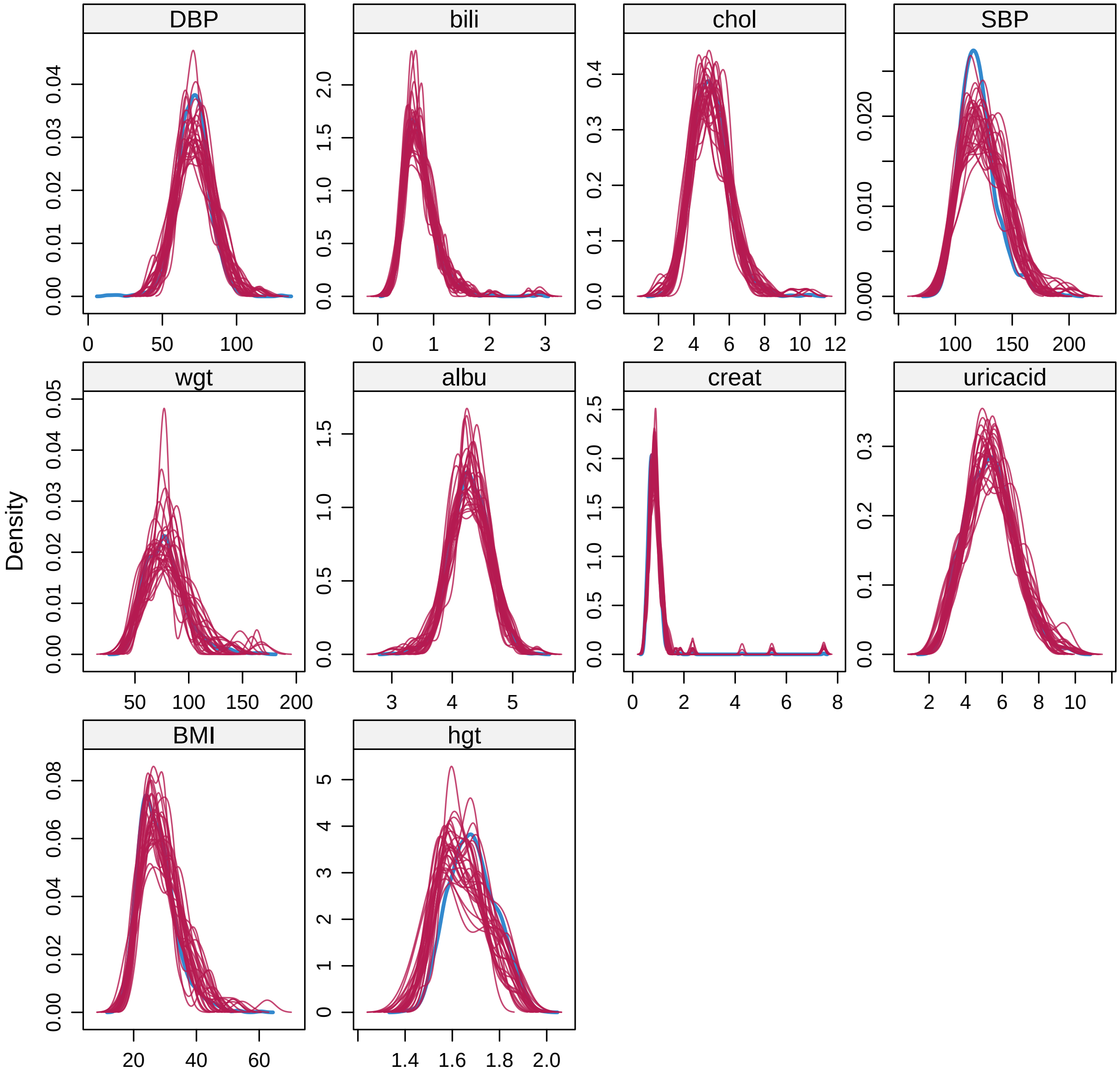
We need to check whether the MICE algorithm has converged. as without convergence there is no guarantee that the results we are obtaining are correct. The mean and variance of the imputed values per iteration and variable are stored in the elements `chainMean` and `chainVar` of the `mids` object `imp`. We can just plot our object and visualise the traceplots.

```
plot(imp, layout = c(6,6))
```



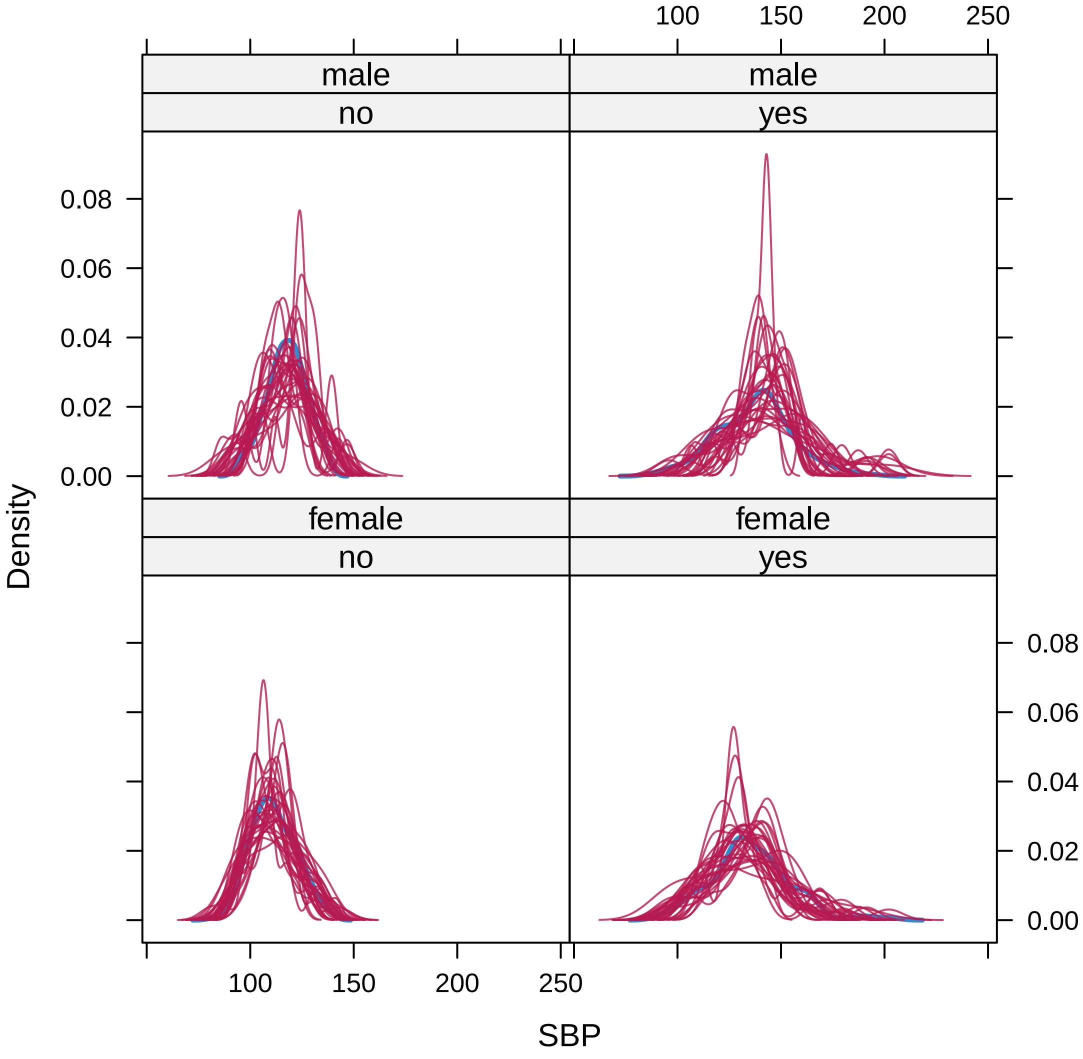
Now that we know that the iterative algorithm appears to have converged for all variables that were imputed, we can compare the distribution of the imputed values against the distribution of the observed values. We start doing that for the continuous variables.

```
densityplot(imp)
```

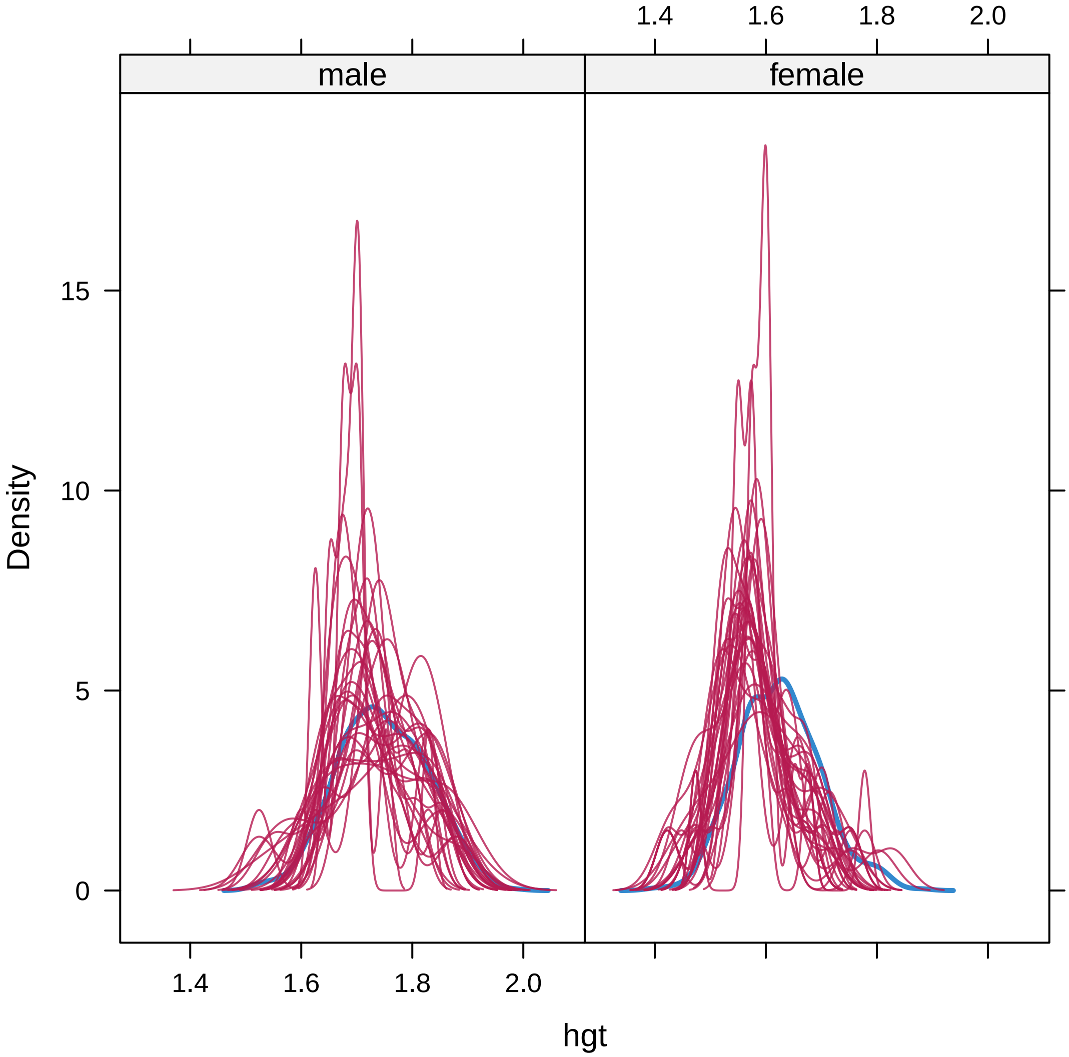


First note that we are using $M = 30$ and because the density of the observed data (the one in blue) is possibly plotted first, we can barely see it. The most outstanding plots are the ones from SBP and height (which, by consequence, also affects the BMI). Although there is nothing here that we should be too worried about, let us investigate if such differences in the two distributions (observed versus imputed), can be explained by other variables. Specifically, let us check SBP conditional on the gender and hypertensive status and height conditional on gender.

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



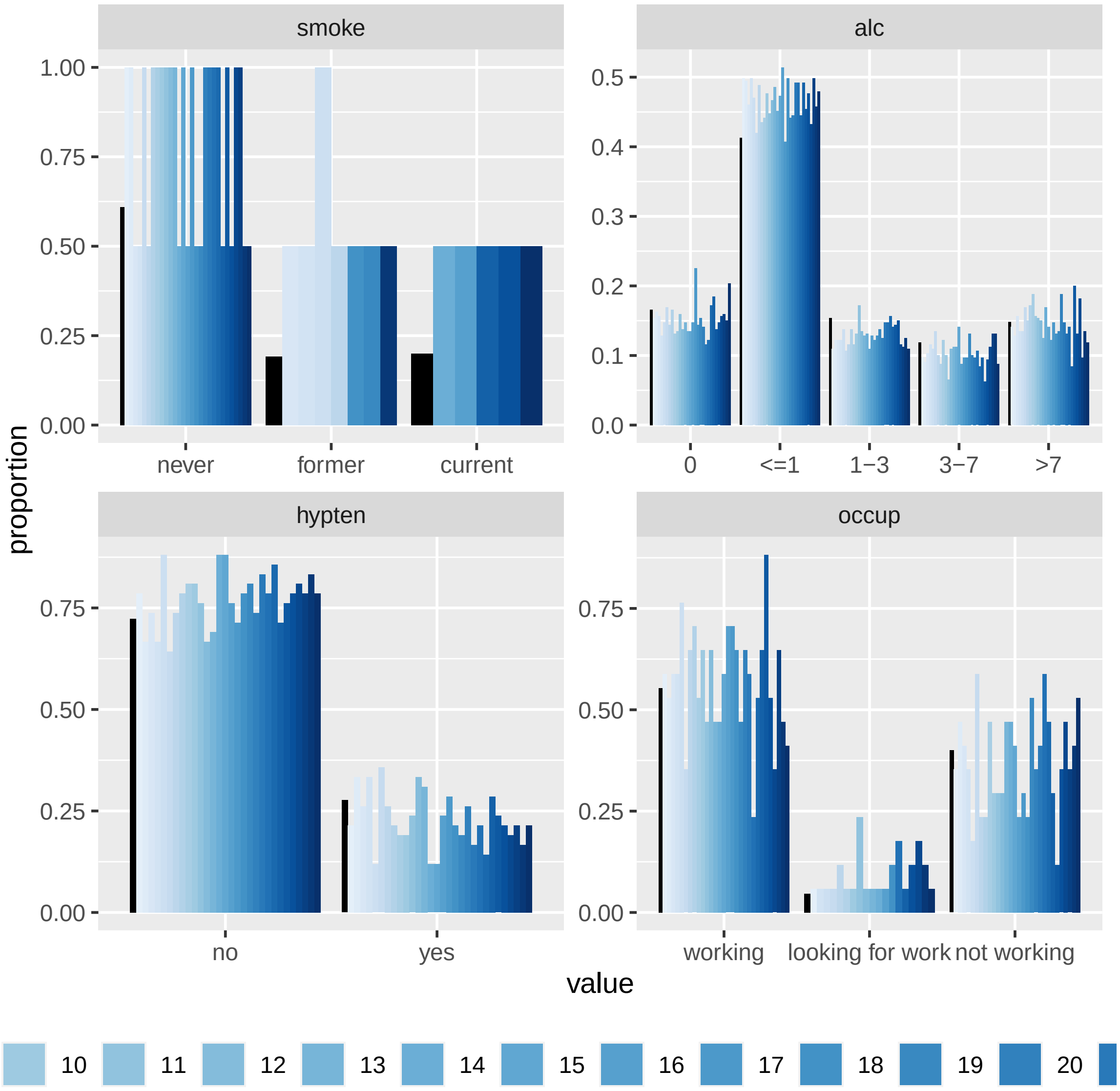
```
densityplot(imp, ~hgt|gender)
```



It seems that gender and hypertensive status, to a certain extent, explain the differences between the observed and imputed values for SBP. That seems to be less the case for the variable height.

With regard to binary/categorical variables, we can compare the proportion of values in each category. `mice` does not provide a function to do this, but there is a nice one, `propplot`, implemented by Nicole Erler and available on her github.

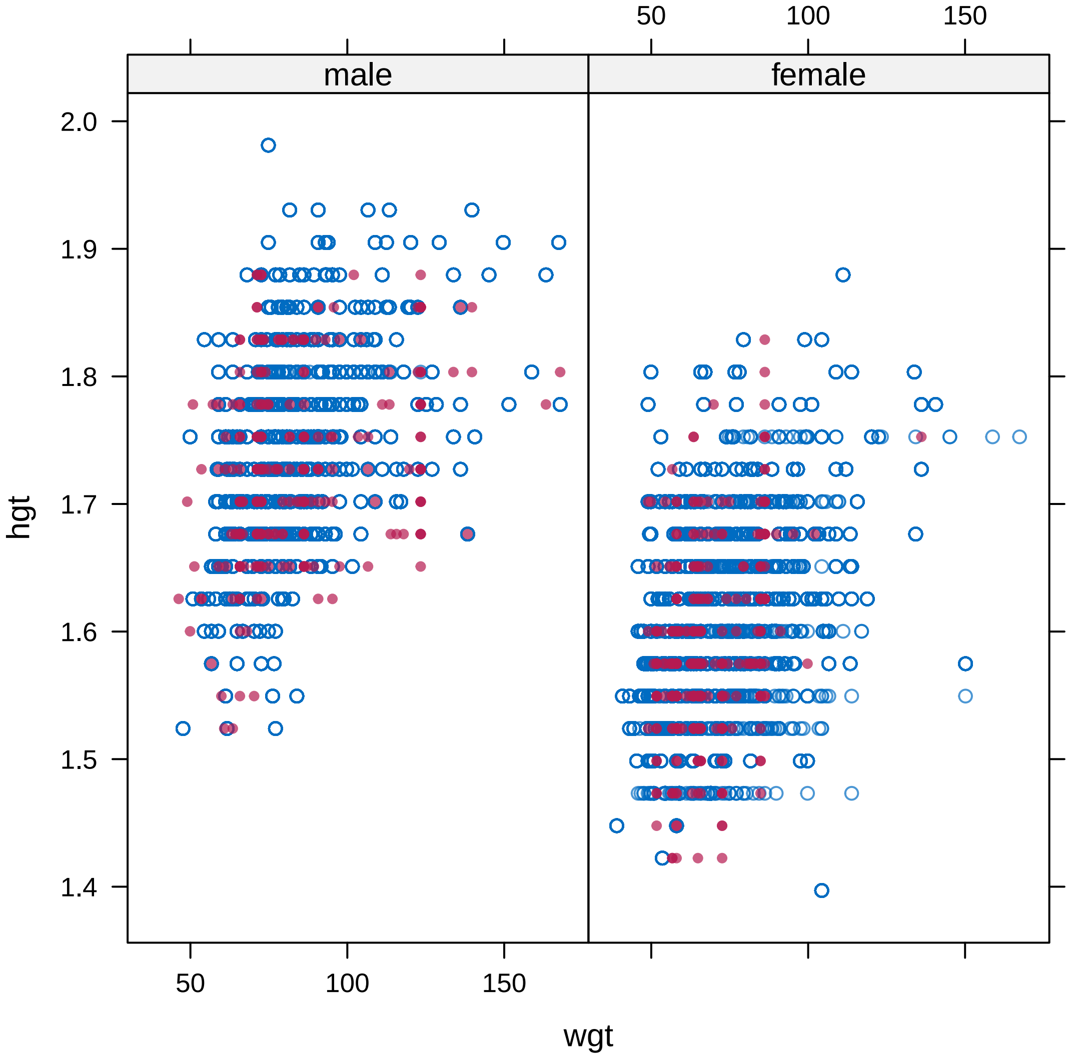
```
require(devtools)
require(reshape2)
require(RColorBrewer)
require(ggplot2)
source_url("https://gist.githubusercontent.com/NErler/0d00375da460dd33839b98faeee2fdab/raw/c6f537ecf80ee

propplot(imp)
```



We observe a large discrepancy between the observed and imputed data distributions for the smoke and occupational staus variable, but because the smoking status variable only has 2 missing values and the occupational status has 17 (out of 1000), we should not be too worried about this.

The function `xyplot()` allows to visualise scatterplots of the imputed and observed values for pairs of variables.

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



Having confirmed that our imputation step was successful, we can proceed to the analysis of the imputed data. For the sake of illustration we will assume that our substantive model of interest in the one in (1).

```
fit <-  with(imp, lm(SBP ~ DM + age + BMI + hypten))
```

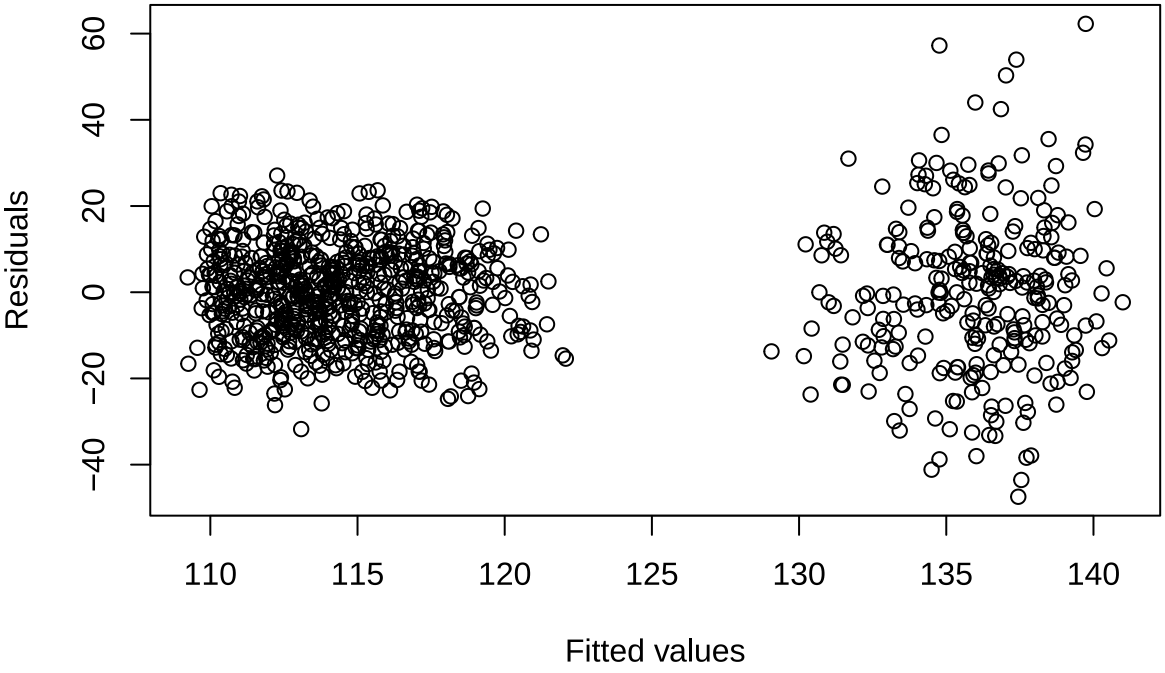
We can further explore the information contained in the object `fit`, which we already know is of class `mira`. For instance, we can look at the summary of the fitted model in the first imputed dataset.

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

```
##
## Call:
## lm(formula = SBP ~ DM + age + BMI + hypten)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -47.445  -8.958   0.388   8.236  62.262
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 103.42948    2.31786  44.623  < 2e-16 ***
## DMyes         0.60854    1.28151   0.475   0.6350
## age            0.16956    0.02984    5.682 1.74e-08 ***
## BMI            0.14603    0.07314    1.996   0.0462 *
## hyptenyes     18.31166    1.07642   17.012  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.23 on 995 degrees of freedom
## Multiple R-squared:  0.3656, Adjusted R-squared:  0.363
## F-statistic: 143.3 on 4 and 995 DF,  p-value: < 2.2e-16
```

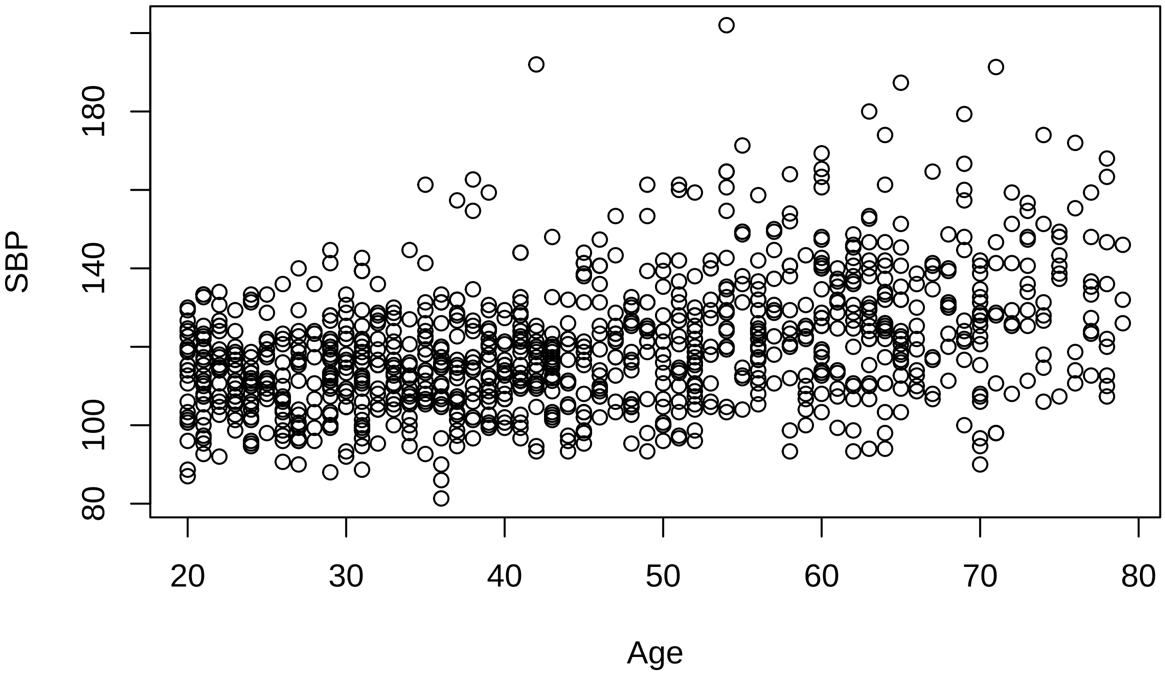
Also, to do model specification/validation, e.g., transformations, we can either look at the complete cases or use one of the completed/imputed datasets. Any transformations will have to apply to all the datasets, so we should not be too dataset-specific in our checks. If we decide transformations are needed, we might reconsider the imputation models too and fit them with transformed values.

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

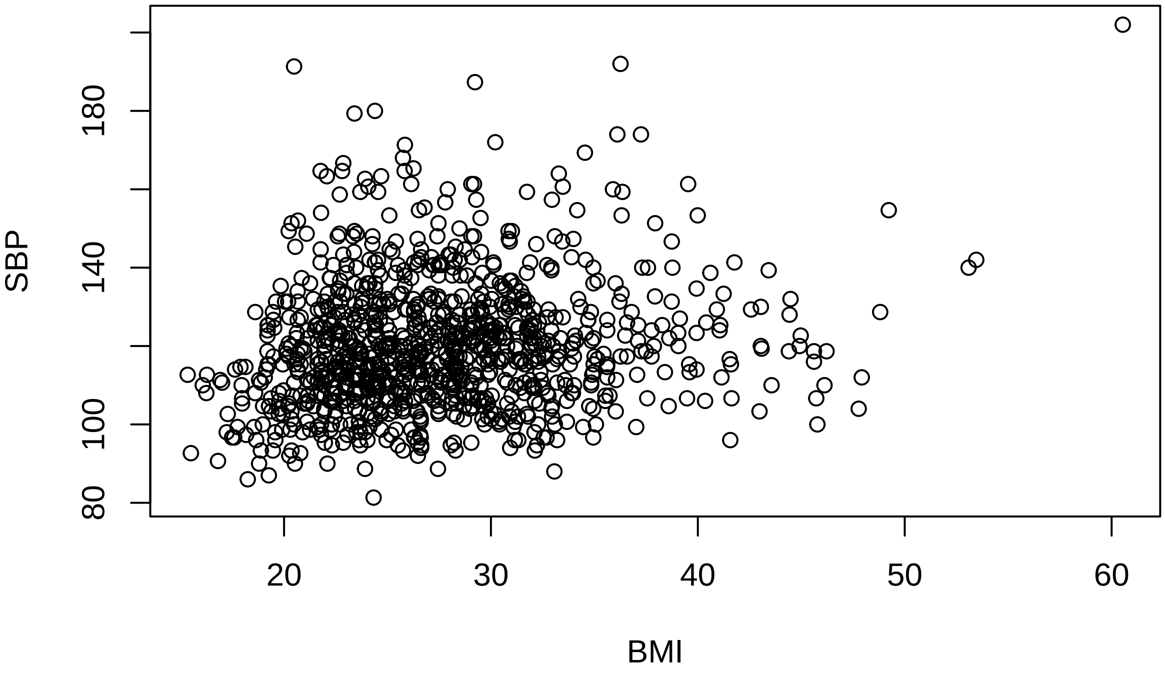


In this fitted values versus residuals plot we can observe two clusters. The residuals are symmetric about zero, and we can suspect that maybe the homoscedastic assumption is slightly violated, but nothing that we cannot live with. I was however curious and decided to plot the response variable, SBP, against the other variables. They seem to indicate that the major cause of the two clusters is the hypertensive status.

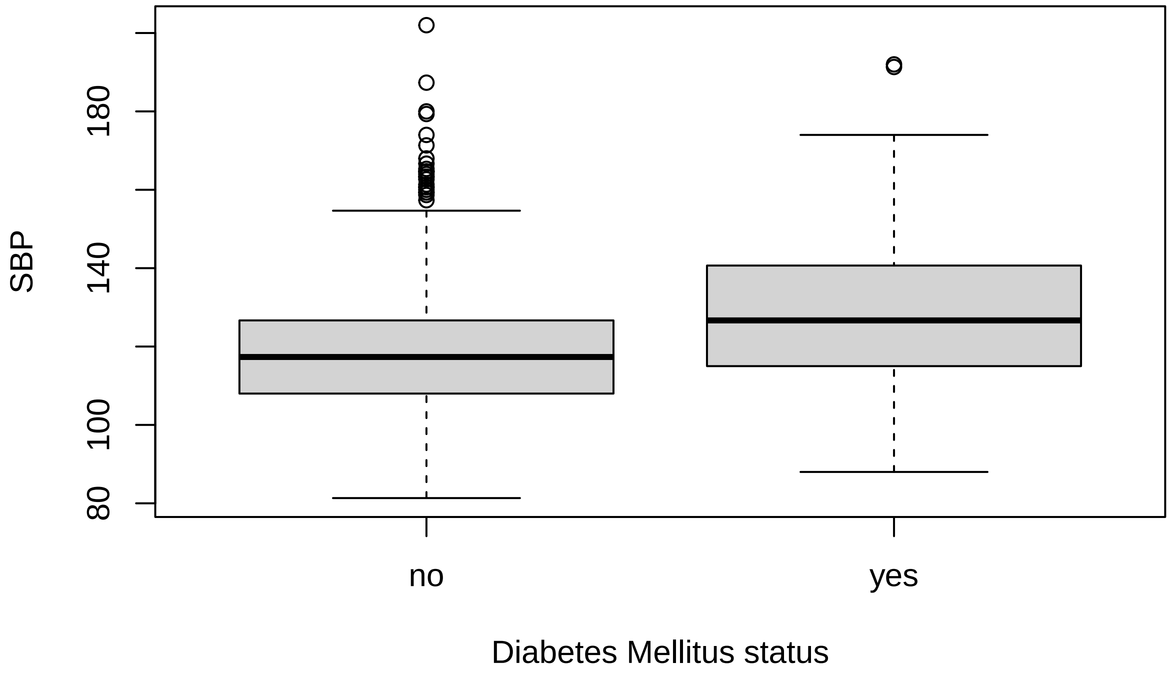
```
plot(comp1$SBP ~ comp1$age, xlab = "Age", ylab = "SBP")
```



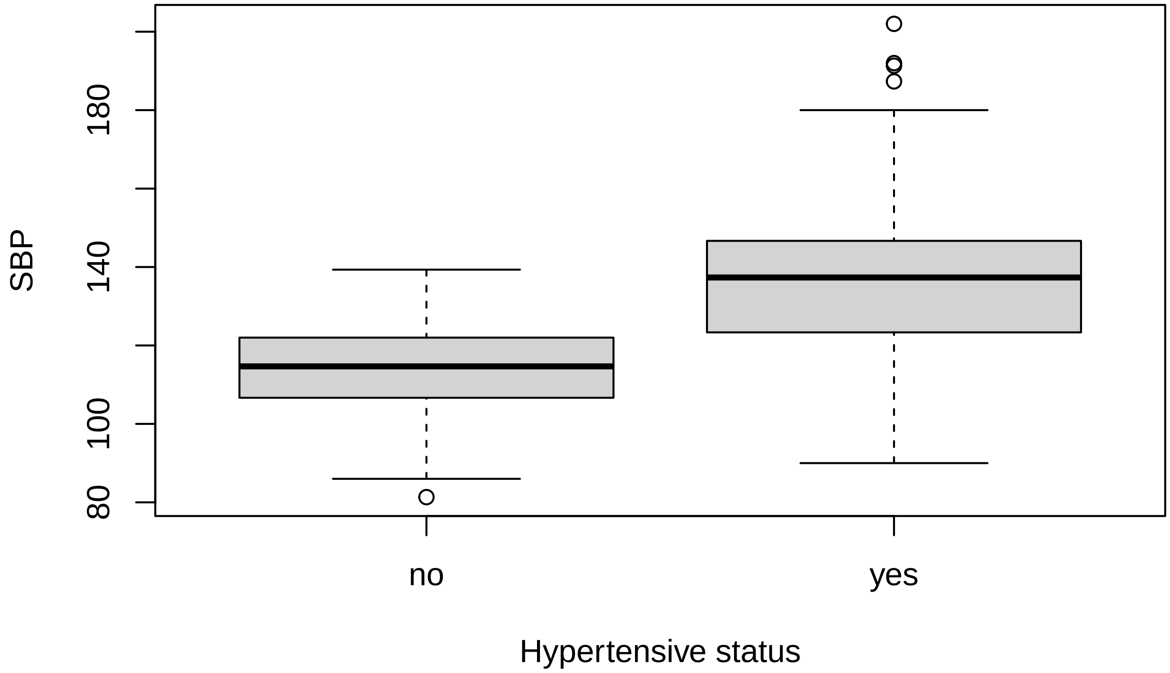
```
plot(comp1$SBP ~ comp1$BMI, xlab = "BMI", ylab = "SBP")
```



```
boxplot(comp1$SBP ~ comp1$DM, xlab = "Diabetes Mellitus status", ylab = "SBP")
```

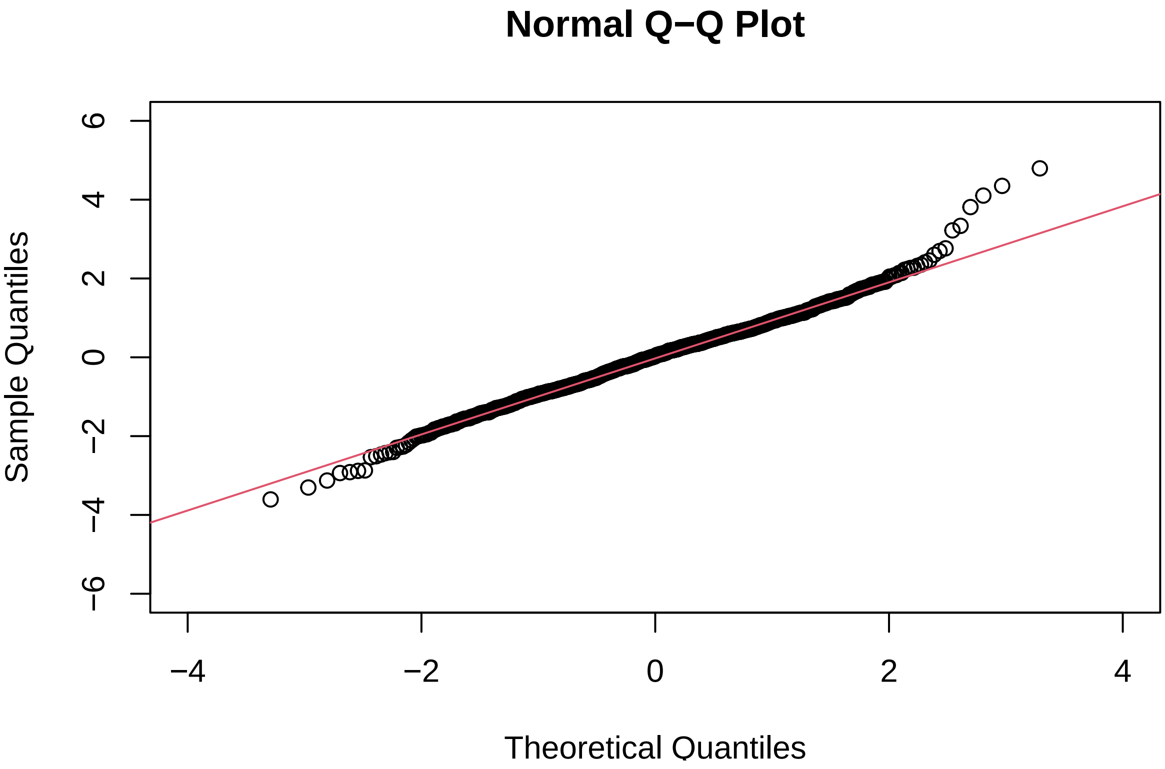


```
boxplot(comp1$SBP ~ comp1$hypten, xlab = "Hypertensive status", ylab = "SBP")
```



We can also do a QQplot and nothing looks suspicious.

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



Pooling the results is no different from what we have done last week.

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

```
##          term    estimate  std.error    statistic       df      p.value
## 1 (Intercept) 103.76513101 2.41329310 42.997318033 771.0537 0.000000e+00
## 2       DMyes   0.00924365 1.34865947  0.006853954 682.2390 9.945334e-01
## 3         age   0.17370089 0.03114938  5.576384226 707.3350 3.496302e-08
## 4         BMI   0.13097600 0.07705484  1.699776460 678.1499 8.963154e-02
## 5   hyptenyes  18.93341630 1.12973510 16.759164396 689.9535 0.000000e+00
##         2.5 %      97.5 %
## 1 99.02772709 108.5025349
## 2 -2.63877807   2.6572654
## 3  0.11254459   0.2348572
## 4 -0.02031873   0.2822707
## 5 16.71528510  21.1515475
```

`mice` has some functions for evaluating model fit or to perform model comparison. Last week we have already seen the function `pool.r.squared` that calculates the pooled (adjusted) $R^2$.

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

```
##                est     lo 95     hi 95       fmi
## adj R^2 0.3748671 0.3247665 0.4243171 0.0937342
```

To compare nested models `mice` has the functions `D1()` and `D3()`, which implement a multivariate Wald test and a likelihood-ratio test statistic, respectively. The function `D2()` allows to pool test statistics when no variance-covariance matrix is available. It is out of the scope of this course to go into the details of these tests but more information can be found in the book by van Buuren (2018, Section 5.3). Just for the sake of illustrating their syntax, suppose that we want to test whether the diabetes mellitus variable has a relevant contribution for the SBP model. We should fit the model without DM and compare the two models.

```
fit_no_DM <- with(imp, lm(SBP ~ age + BMI + hypten))
D1(fit, fit_no_DM)
```

```
##    test    statistic df1      df2 dfcom   p.value       riv
##  1 ~~ 2 4.697669e-05   1 681.6054   995 0.9945334 0.1114138
```

In this case the Wald test statistic is not significant, and therefore the DM has no relevant contribution to the SBP model. Let us now try check the contribution of the hypertensive status variable. Our common sense would dictate that this would be a relevant variable, and our former investigations already show us that this variable is the main reason behind the two clusters observed in the residuals plot.

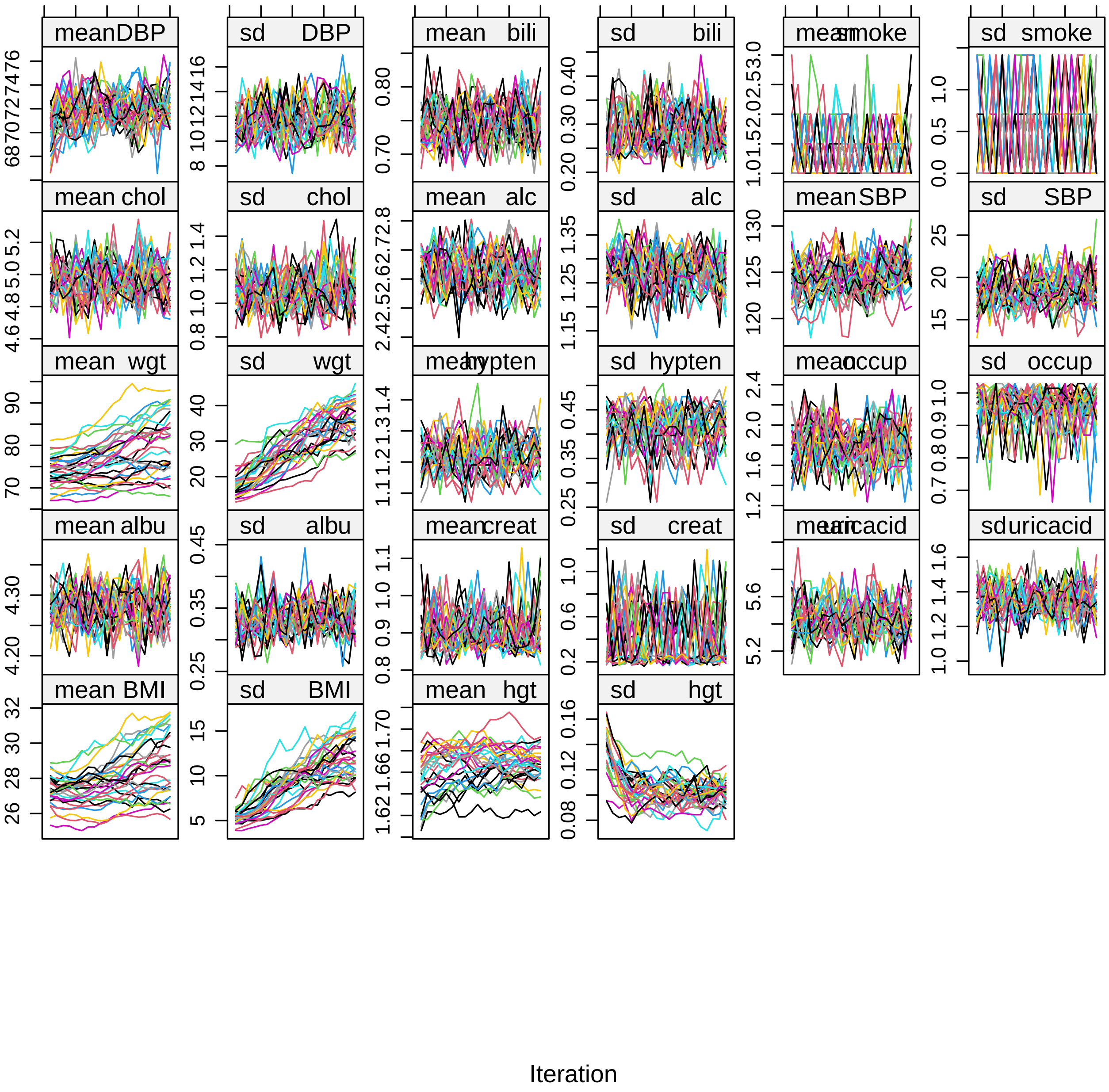
```
fit_no_hypten <- with(imp, lm(SBP ~ DM + age + BMI))
D1(fit, fit_no_hypten)
```

```
##    test statistic df1      df2 dfcom      p.value       riv
##  1 ~~ 2  280.8696   1 690.1973   995 3.804788e-53 0.1089154
```

We now have a significant Wald test and therefore we should keep this variable in our model. Of course, we should check if we increase $M = 50$, say, if our conclusions remain the same.

To conclude and quite unrelated but because I would like to illustrate it, let us look at the trace plots we would have obtained in case we have not adjusted our `predictorMatrix` and BMI would be used to impute both height and weight.

```
imp_0_naive <- mice(NHANES, maxit = 0)
meth_naive <- imp_0_naive$method
meth_naive["BMI"] <- "~I(wgt/(hgt^2))"

pred_naive <- imp_0_naive$predictorMatrix
pred_naive[, c("hgt", "wgt")] <- 0
pred_naive["hgt", "wgt"] <- 1
pred_naive["wgt", "hgt"] <- 1

visSeq_naive <- imp_0_naive$visitSequence
which_BMI <- match("BMI", visSeq_naive)
visSeq_naive <- c(visSeq_naive[-which_BMI], visSeq_naive[which_BMI])

imp_naive <- mice(NHANES, method = meth_naive, predictorMatrix = pred_naive,
                  visitSequence = visSeq_naive, maxit = 20, m = 30,
                  seed = 1, printFlag = FALSE)

plot(imp_naive, layout = c(6,6))
```



We can clearly see that the chains for BMI, height, and weight are not mixing well. Note that the only thing that we have changed is that now BMI is used as a predictor for height and weight.