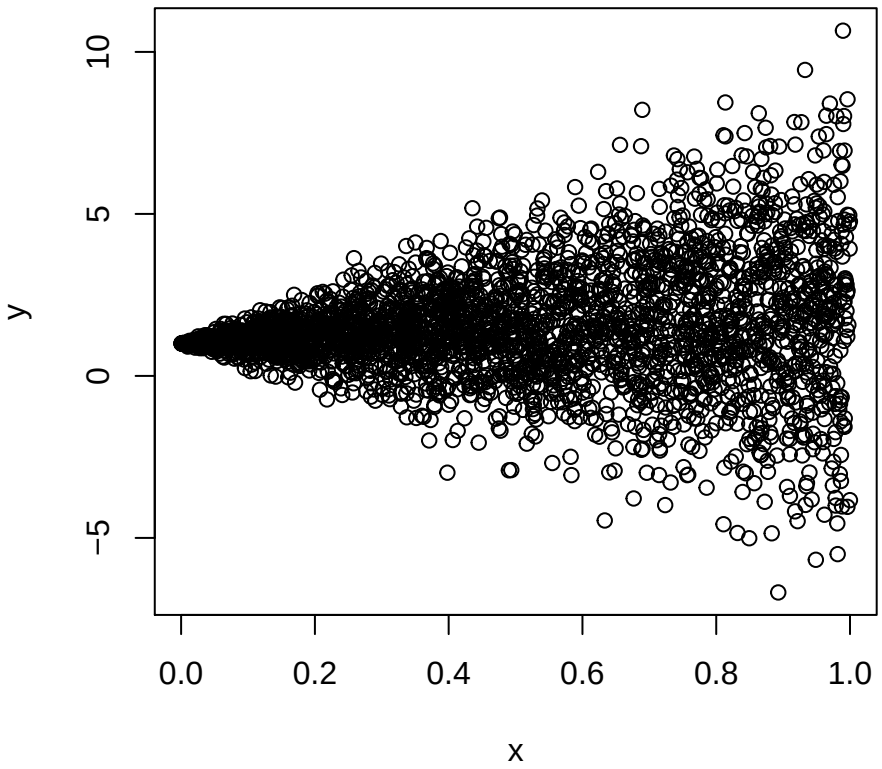
University of Edinburgh, School of Mathematics
Incomplete Data Analysis, 2020/2021
Workshop 5 – Solutions

Vanda Inácio

1. We start by simulating the data as instructed in the worksheet.

```
set.seed(1)
n <- 3000
x <- runif(n)
y <- rnorm(n, 1 + x, 3*x)
# Inspecting the scatterplot of the data
plot(x, y, xlab = expression(x), ylab = expression(y))
```



We will now randomly set 30% of the y values to be missing.

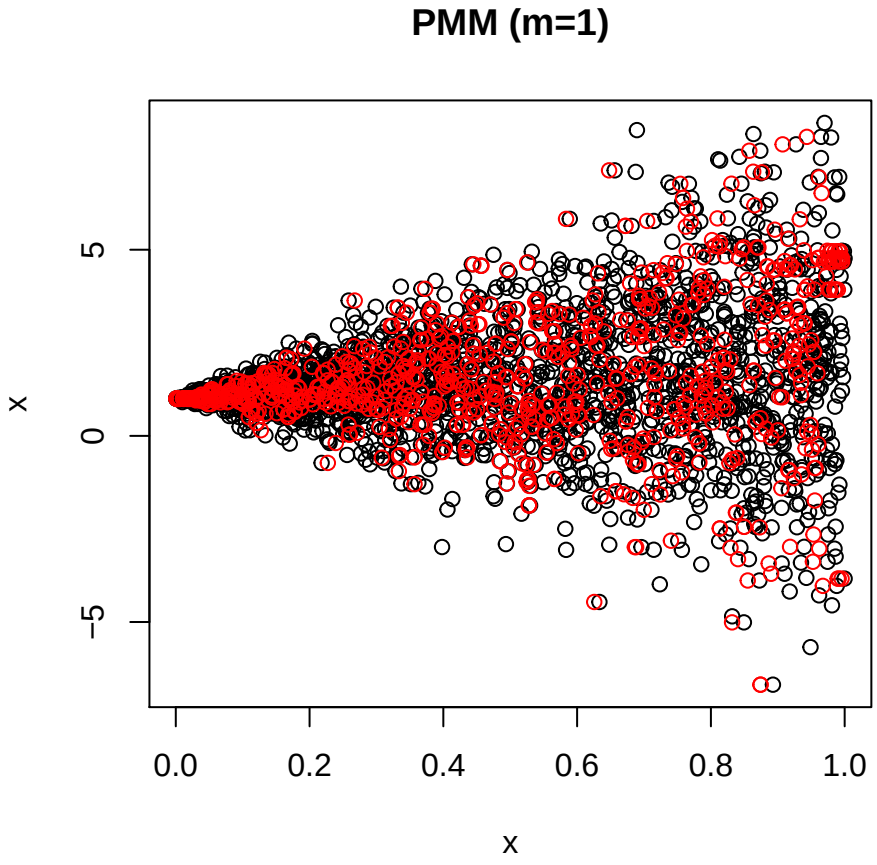
```
n_mis <- n*0.3
ind_mis <- sample(x = 1:n, size = n_mis, replace = FALSE)
y_mis <- c(y[-ind_mis], rep(NA, n_mis))
#all x values are observed but I am reordering them according to the corresponding y values
#(observed versus missing)
x_mis <- c(x[-ind_mis], x[ind_mis])
```

I will now impute the data using the predictive mean matching defaults used by `mice()`, i.e., Type 1 matching and 5 donors.

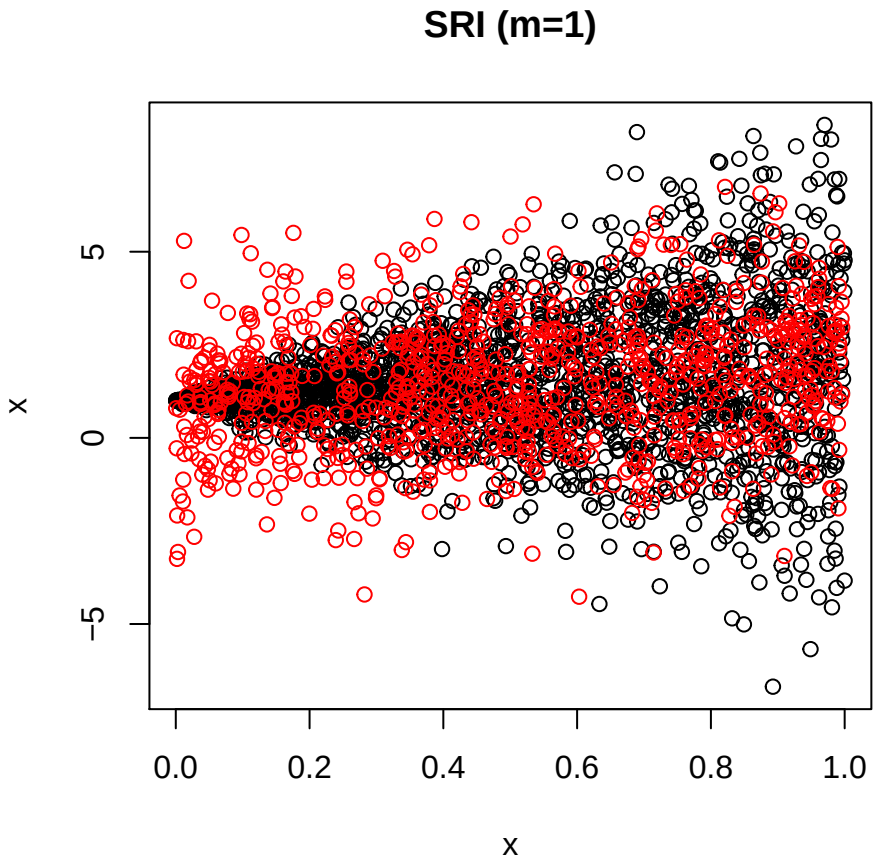
```
require(mice)
imp_pmm <- mice(data.frame("x" = x_mis, "y" = y_mis), m = 4,
                seed = 1, printFlag = FALSE)
imp_sri <- mice(data.frame("x" = x_mis, "y" = y_mis), m = 4,
                seed = 1, method = "norm.nob", printFlag = FALSE)
```

I will now construct the scatter plot of the observed (black circles) and imputed (red circles) data.

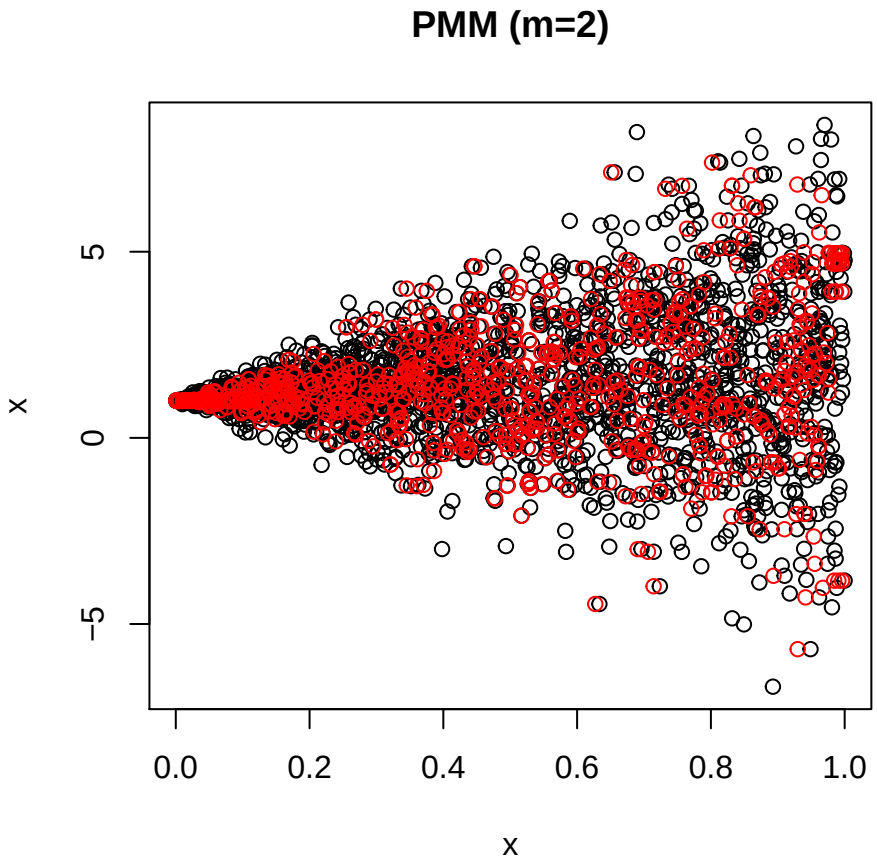
```
#extracting the completed datasets
comp_pmm_1 <- complete(imp_pmm, 1)
comp_sri_1 <- complete(imp_sri, 1)

comp_pmm_2 <- complete(imp_pmm, 2)
comp_sri_2 <- complete(imp_sri, 2)

comp_pmm_3 <- complete(imp_pmm, 3)
comp_sri_3 <- complete(imp_sri, 3)

comp_pmm_4 <- complete(imp_pmm, 4)
comp_sri_4 <- complete(imp_sri, 4)

par(mfrow = c(1,1))

plot(comp_pmm_1$x[1:(n-n_mis)],comp_pmm_1$y[1:(n-n_mis)],
     main = "PMM (m=1)", xlab = expression(x), ylab = expression(x)) #observed data
points(comp_pmm_1$x[(n-n_mis+1):n],comp_pmm_1$y[(n-n_mis+1):n], col = "red") #imputed data
```



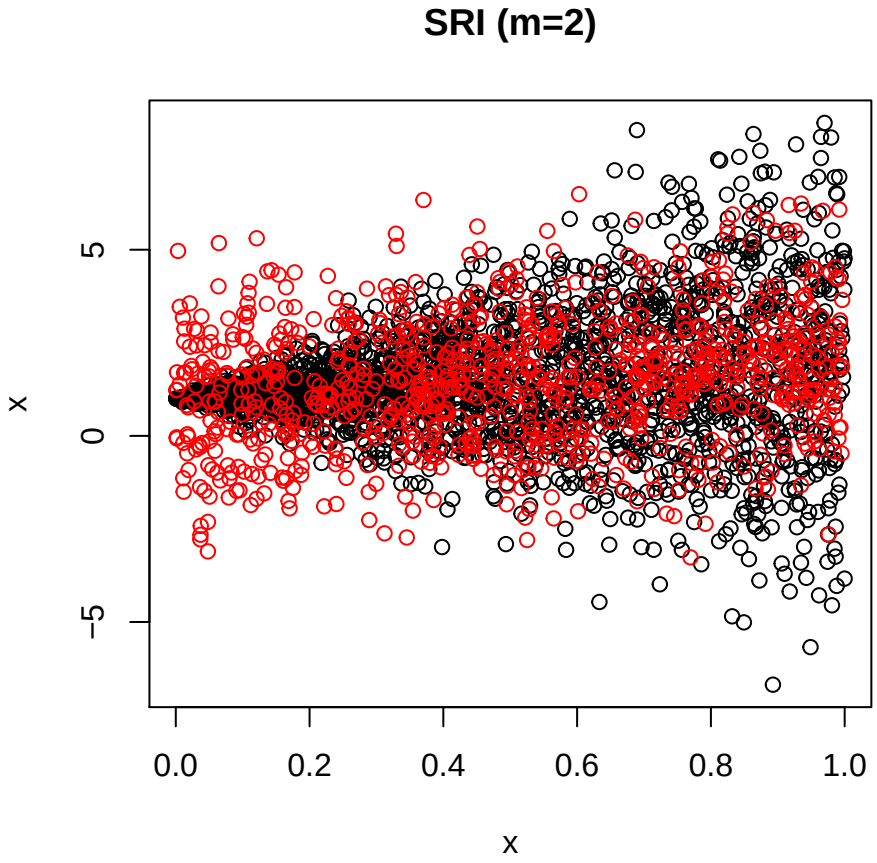

```
plot(comp_sri_1$x[1:(n-n_mis)],comp_sri_1$y[1:(n-n_mis)],
     main = "SRI (m=1)", xlab = expression(x), ylab = expression(x))
points(comp_sri_1$x[(n-n_mis+1):n],comp_sri_1$y[(n-n_mis+1):n], col = "red")
```



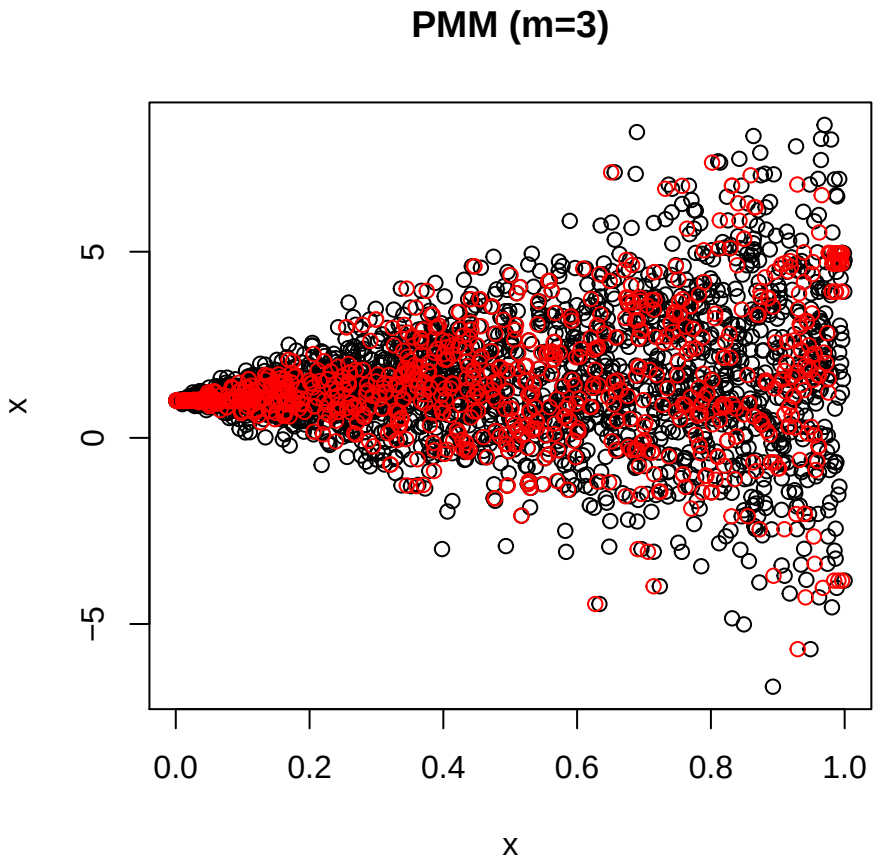

```
plot(comp_pmm_2$x[1:(n-n_mis)],comp_pmm_2$y[1:(n-n_mis)],
     main = "PMM (m=2)", xlab = expression(x), ylab = expression(x)) #observed data
points(comp_pmm_2$x[(n-n_mis+1):n],comp_pmm_2$y[(n-n_mis+1):n], col = "red") #imputed data
```



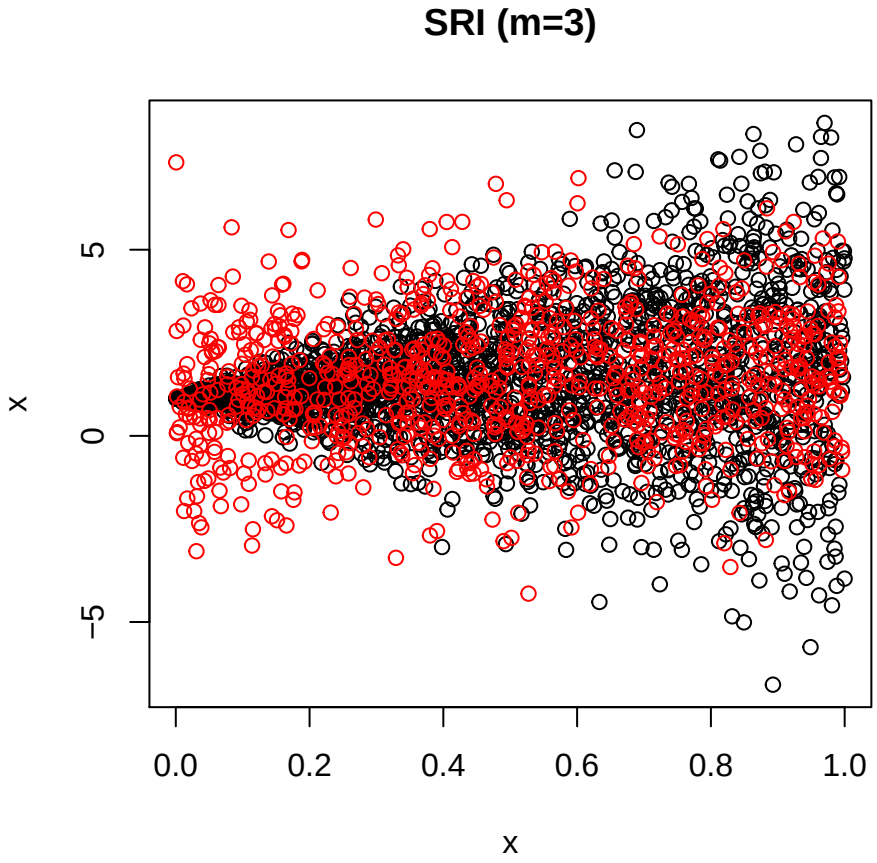

```
plot(comp_sri_2$x[1:(n-n_mis)],comp_sri_2$y[1:(n-n_mis)],
     main = "SRI (m=2)", xlab = expression(x), ylab = expression(x))
points(comp_sri_2$x[(n-n_mis+1):n],comp_sri_2$y[(n-n_mis+1):n], col = "red")
```



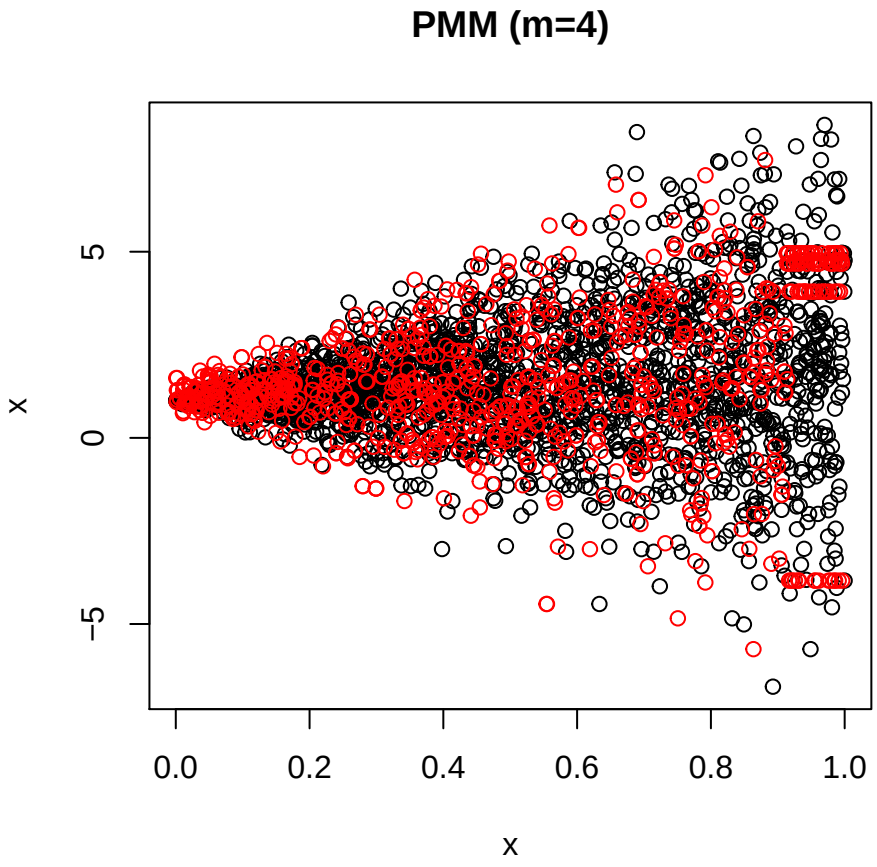

```
plot(comp_pmm_3$x[1:(n-n_mis)],comp_pmm_3$y[1:(n-n_mis)],
     main = "PMM (m=3)", xlab = expression(x), ylab = expression(x)) #observed data
points(comp_pmm_2$x[(n-n_mis+1):n],comp_pmm_2$y[(n-n_mis+1):n], col = "red") #imputed data
```



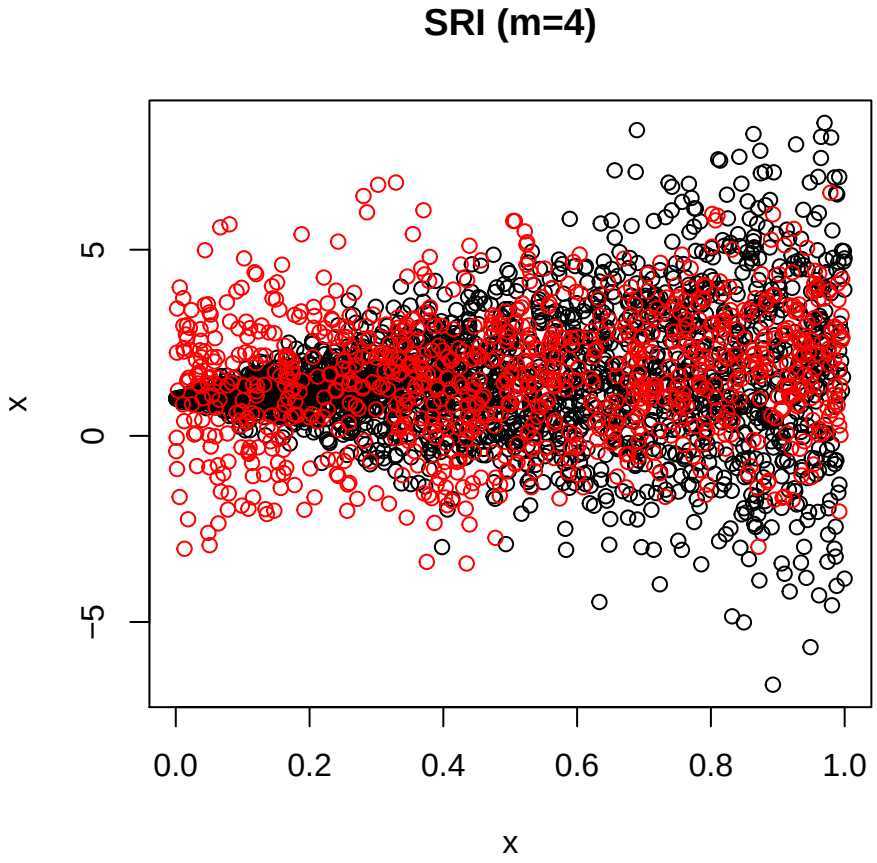

```
plot(comp_sri_3$x[1:(n-n_mis)],comp_sri_3$y[1:(n-n_mis)],
     main = "SRI (m=3)", xlab = expression(x), ylab = expression(x))
points(comp_sri_3$x[(n-n_mis+1):n],comp_sri_3$y[(n-n_mis+1):n], col = "red")
```




```
plot(comp_pmm_4$x[1:(n-n_mis)],comp_pmm_4$y[1:(n-n_mis)],
     main = "PMM (m=4)", xlab = expression(x), ylab = expression(x)) #observed data
points(comp_pmm_4$x[(n-n_mis+1):n],comp_pmm_4$y[(n-n_mis+1):n], col = "red") #imputed data
```



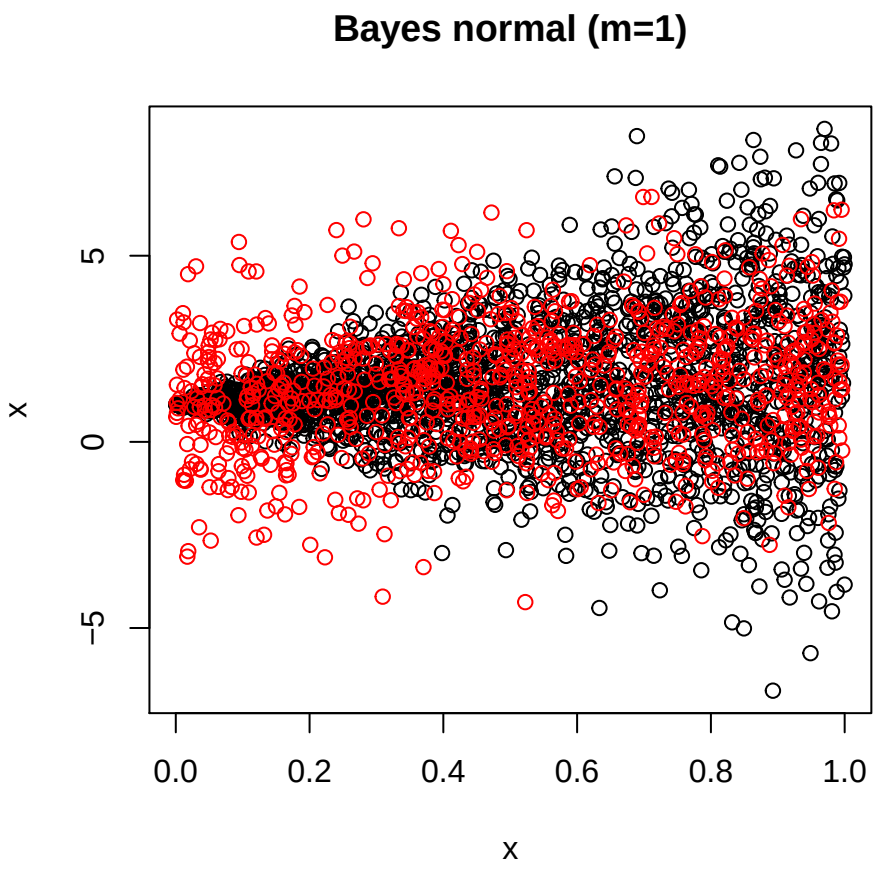

```
plot(comp_sri_4$x[1:(n-n_mis)],comp_sri_4$y[1:(n-n_mis)],
     main = "SRI (m=4)", xlab = expression(x), ylab = expression(x))
points(comp_sri_4$x[(n-n_mis+1):n],comp_sri_4$y[(n-n_mis+1):n], col = "red")
```



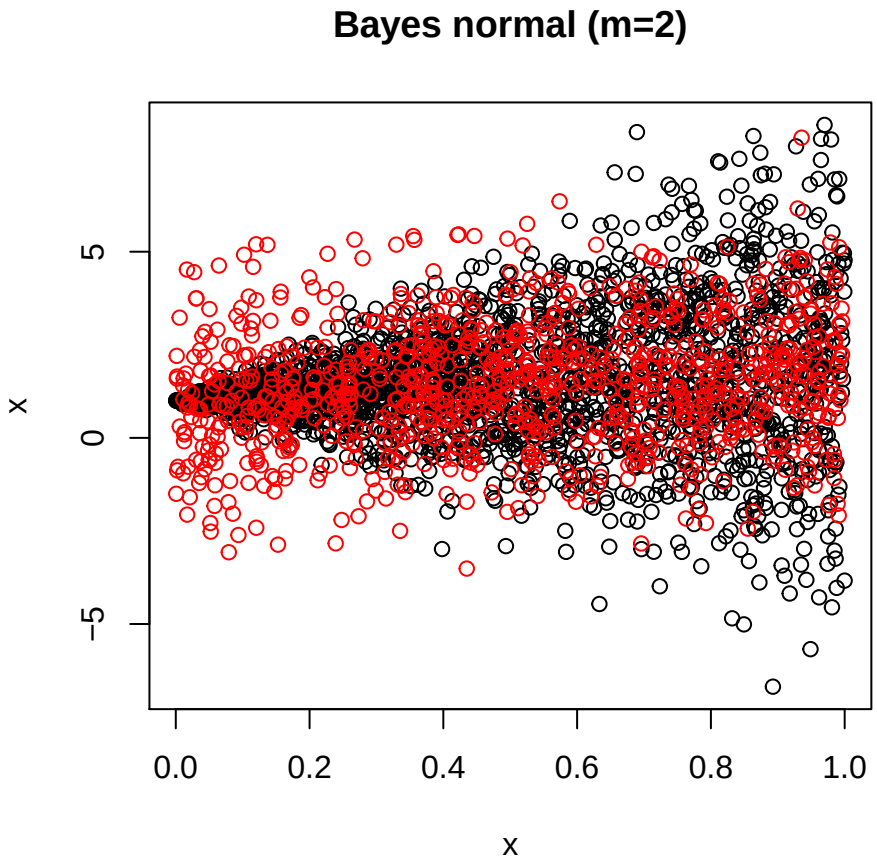
From the four plots we have constructed it is clear that stochastic regression imputation overestimates the variance for smaller values of x and underestimates the variance for larger x values. On the other hand, predictive mean matching reflects the structure of the observed data much better.

I will now impute the data under the Bayesian normal linear method and check how well it recovers the structure of the data. However, I do not expect its performance to be better than stochastic regression imputation, as the main difference between both approaches is that in the Bayesian counterpart uncertainty in the parameters estimates is taken into account

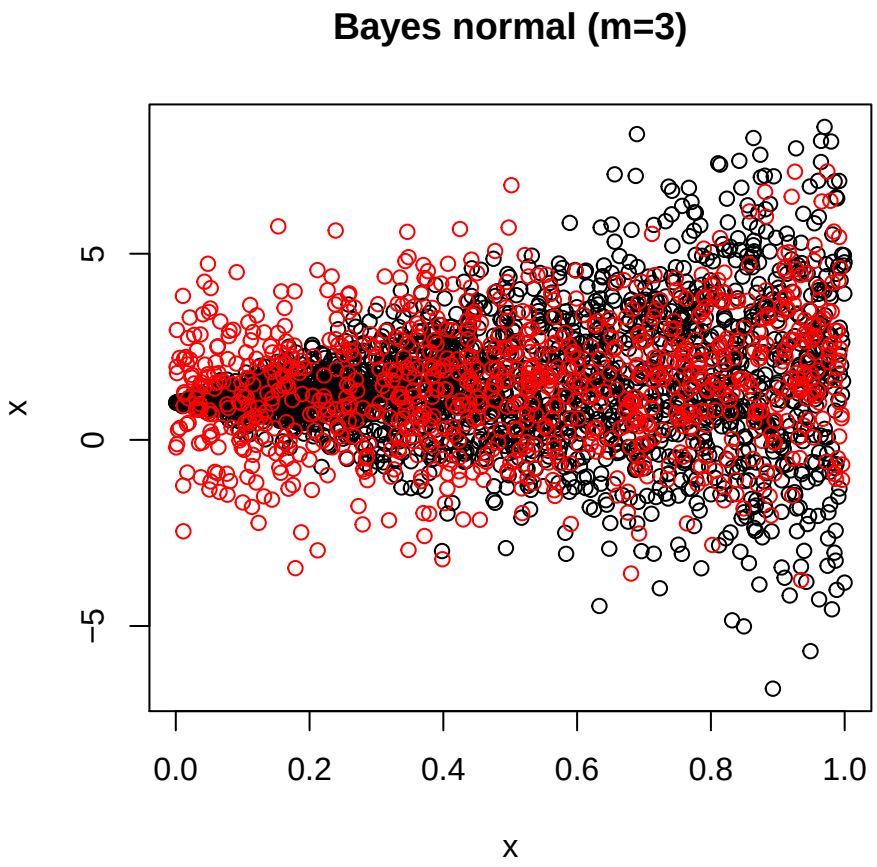
```
require(mice)
imp_norm <- mice(data.frame("x" = x_mis, "y" = y_mis), m = 4,
                 seed = 1, method = "norm", printFlag = FALSE)

comp_norm_1 <- complete(imp_norm, 1)
comp_norm_2 <- complete(imp_norm, 2)
comp_norm_3 <- complete(imp_norm, 3)
comp_norm_4 <- complete(imp_norm, 4)

plot(comp_norm_1$x[1:(n-n_mis)],comp_norm_1$y[1:(n-n_mis)],
     main = "Bayes normal (m=1)", xlab = expression(x), ylab = expression(x))
points(comp_norm_1$x[(n-n_mis+1):n],comp_norm_1$y[(n-n_mis+1):n], col = "red")
```



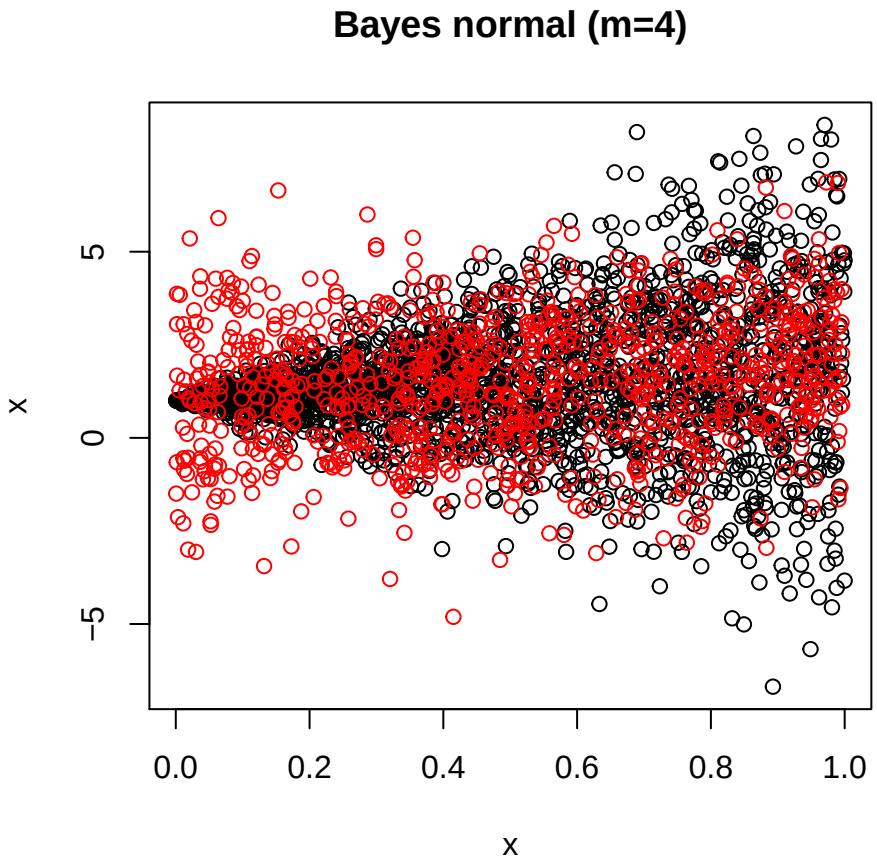

```
plot(comp_norm_2$x[1:(n-n_mis)],comp_norm_2$y[1:(n-n_mis)],
     main = "Bayes normal (m=2)", xlab = expression(x), ylab = expression(x))
points(comp_norm_2$x[(n-n_mis+1):n],comp_norm_2$y[(n-n_mis+1):n], col = "red")
```




```
plot(comp_norm_3$x[1:(n-n_mis)],comp_norm_3$y[1:(n-n_mis)],
     main = "Bayes normal (m=3)", xlab = expression(x), ylab = expression(x))
points(comp_norm_3$x[(n-n_mis+1):n],comp_norm_3$y[(n-n_mis+1):n], col = "red")
```




```
plot(comp_norm_4$x[1:(n-n_mis)],comp_norm_4$y[1:(n-n_mis)],
     main = "Bayes normal (m=4)", xlab = expression(x), ylab = expression(x))
points(comp_norm_4$x[(n-n_mis+1):n],comp_norm_4$y[(n-n_mis+1):n], col = "red")
```



As expected, the Bayesian (normal linear) approach also fails to recover the heteroscedastic structure of the data.

2. Let us start by inspecting the data.

```
load("ncds.Rdata")
dim(ncds)
```

```
## [1] 17631     7
```

```
str(ncds)
```

```
## 'data.frame':    17631 obs. of  7 variables:
##  $ noqual2 : Factor w/ 2 levels "at least 1","none": 1 2 NA 1 1 NA 1 1 1 1 ...
##  $ care    : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ soch7   : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 2 1 ...
##  $ mo_age  : int  8 2 -2 9 4 -3 -1 0 -5 -3 ...
##  $ readtest: int  19 23 NA NA 32 NA NA 26 NA 22 ...
##  $ bsag    : int  NA 18 NA 2 NA NA 10 3 NA NA ...
##  $ sex     : Factor w/ 2 levels "boy","girl": 1 2 2 2 2 1 2 1 2 2 ...
```

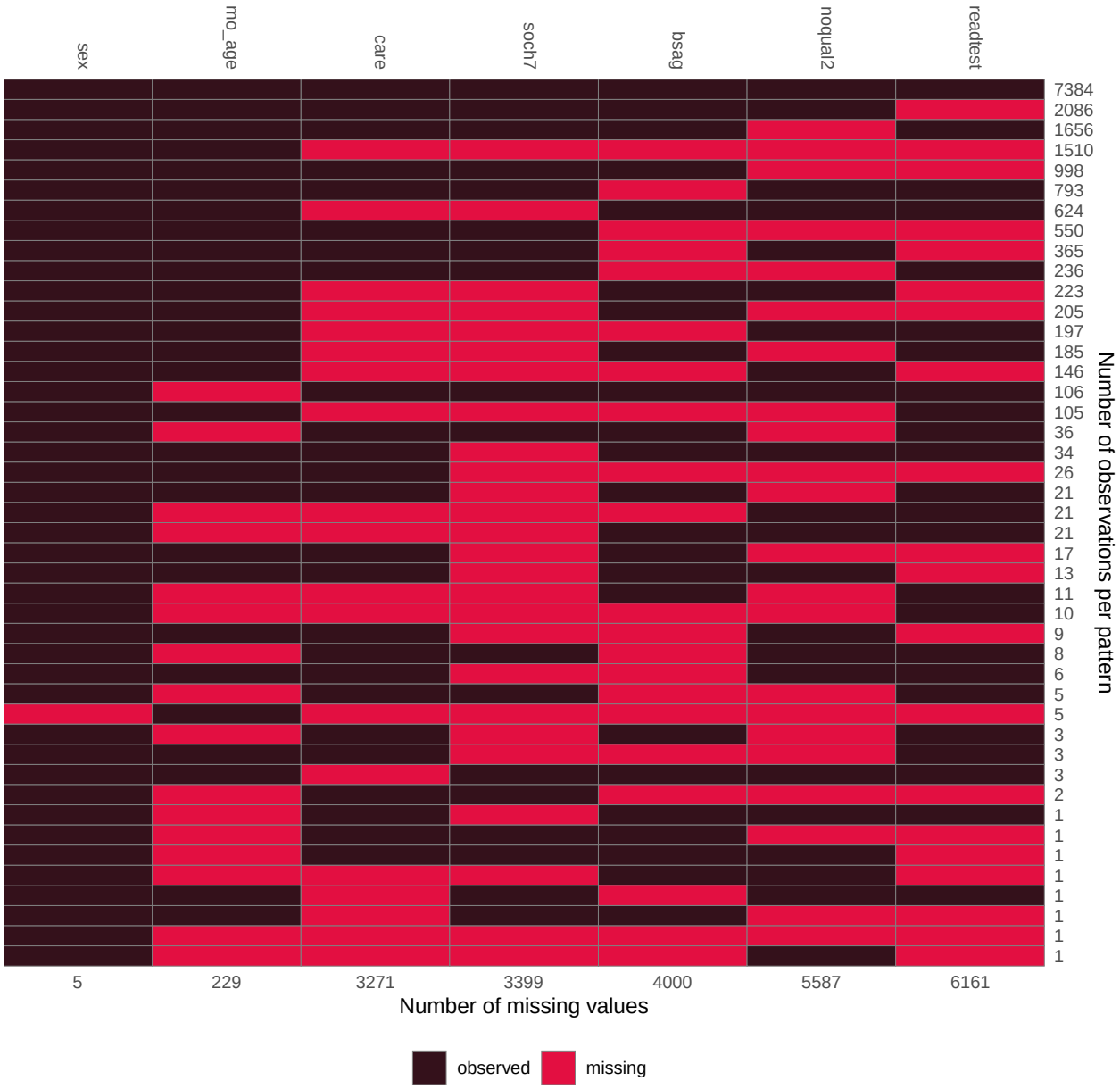
```
summary(ncds)
```

```
##        noqual2       care         soch7          mo_age           readtest    
##  at least 1:8694   no  :14044   no  :8521   Min.   :-13.0000   Min.   : 0.00  
##  none      :3350   yes :  316   yes :5711   1st Qu.: -4.0000   1st Qu.:21.00  
##  NA's      :5587   NA's: 3271   NA's:3399   Median :  0.0000   Median :27.00  
##                                             Mean   :  0.4585   Mean   :25.44  
##                                             3rd Qu.:  4.0000   3rd Qu.:31.00  
##                                             Max.   : 21.0000   Max.   :35.00
```

```
##                                             NA's   :229        NA's   :6161
##       bsag          sex
##  Min.   : 0.000   boy :9108
##  1st Qu.: 2.000   girl:8518
##  Median : 5.000   NA's:   5
##  Mean   : 8.459
##  3rd Qu.:13.000
##  Max.   :70.000
##  NA's   :4000
```

We have 17631 individuals in our dataset, and some variables have a large proportion of missing values (e.g., the reading and behavioural scores and the dependent variable, `noqual2`, in our logistic regression model). Let us now further inspect the missing data patterns.

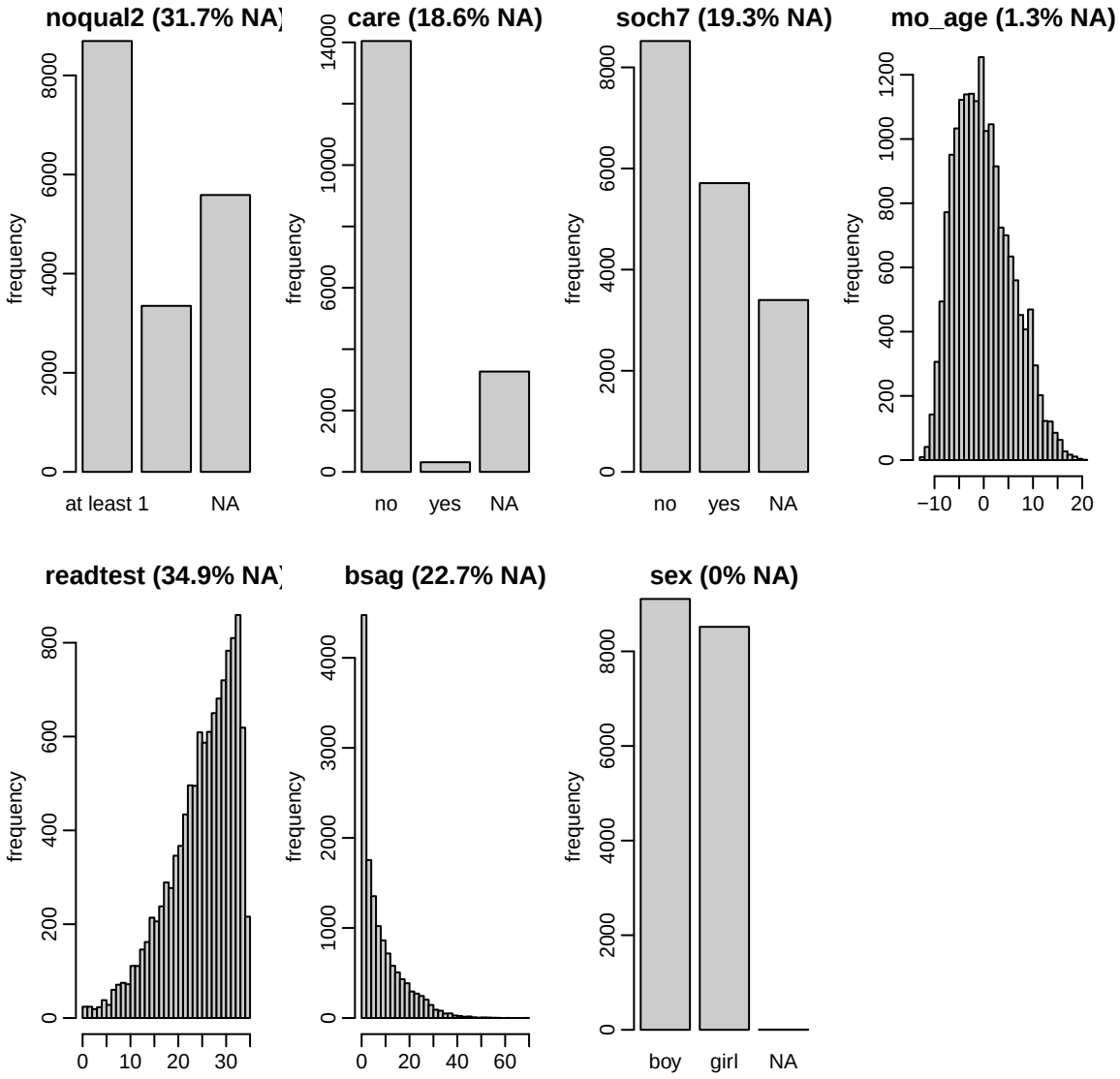
```
require(JointAI)
mdp <- md_pattern(ncds, pattern = TRUE, color = c('#34111b', '#e30f41'))
mdp$plot
```



From the above plot we can see, for instance, that there are 7384 individuals with no missing values in any of the variables and 2086 who have missing values in the reading test but do not have missing values in any of the other variables.

Let us now visualise how the distribution of the observed values in the different variables look like.

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



We can see that the distributions of the continuous variables are all quite skewed, and so predictive mean matching is definitely the best option here. We can also see that the plot for sex indicates that we have 0% of missing values, but this is due to rounding the percentage of missing values (which is, $5/17134 = 0.0002918174$).

We will now proceed to the imputation step. We will start with the dry/setup run of `mice()`.

```
imp0 <- mice(data = ncds, maxit = 0)
imp0$method
```

```
##  noqual2     care    soch7   mo_age readtest     bsag      sex
## "logreg" "logreg" "logreg"    "pmm"    "pmm"    "pmm" "logreg"
```

```
imp0$predictorMatrix
```

```
##          noqual2 care soch7 mo_age readtest bsag sex
## noqual2        0    1     1      1        1    1   1
## care           1    0     1      1        1    1   1
## soch7          1    1     0      1        1    1   1
## mo_age         1    1     1      0        1    1   1
## readtest       1    1     1      1        0    1   1
## bsag           1    1     1      1        1    0   1
```

```
## sex             1     1      1       1         1     1    0
```

We do not have any 'derived' variables here (in the sense that one variable in our dataset can be written as a function of others in our dataset) and further, from the information available from the study, I do not see any reason to change the `predictorMatrix`. The variables that are not in our substantive model, act here as auxiliary variables. Some of them do have a heavy percentage of missing values but nonetheless I will include them. Remember that including auxiliary variables typically also improves the plausibility of the missing at random assumption. For the final imputation I will use `maxit=20` and `M=20`.

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

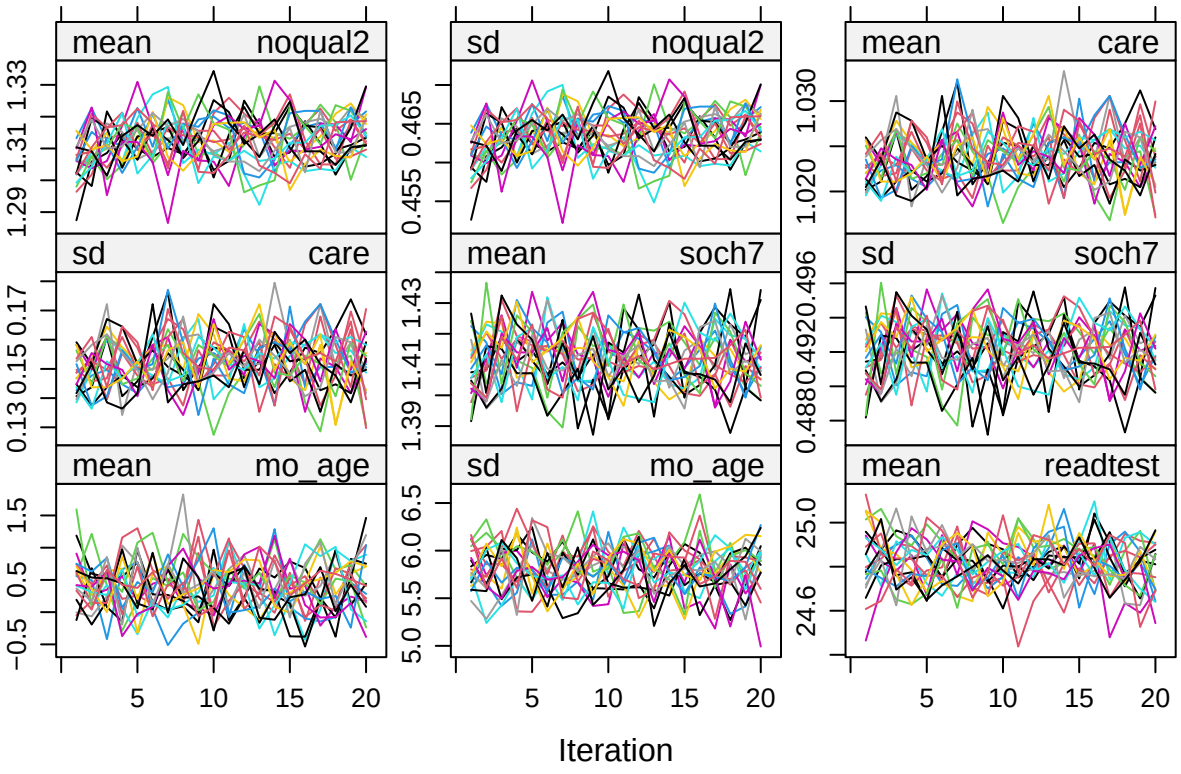
Let us check if `mice()` found any problem during the imputation.

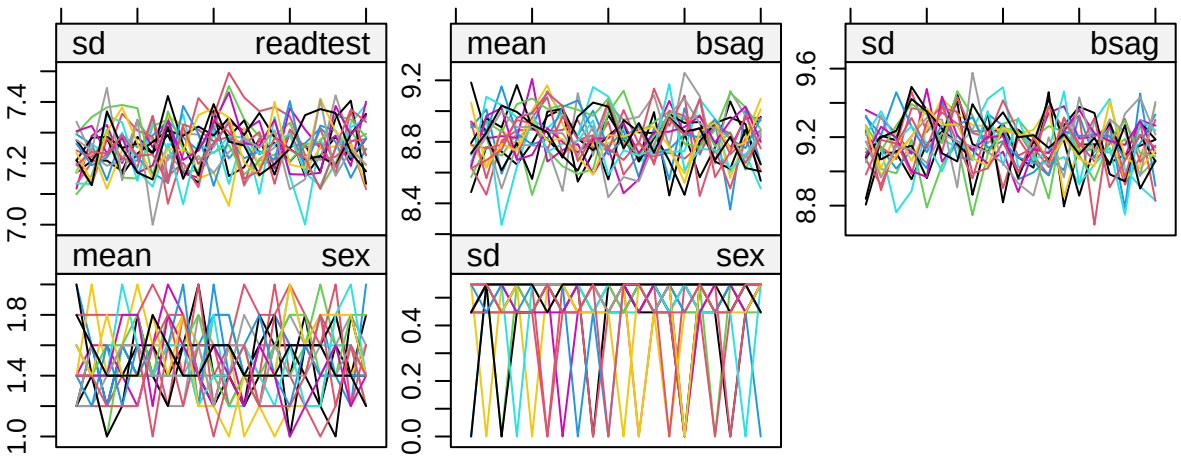
```
imp$loggedEvents
```

```
## NULL
```

Let us now look at the chains of the imputed values to check whether there are convergence problems.

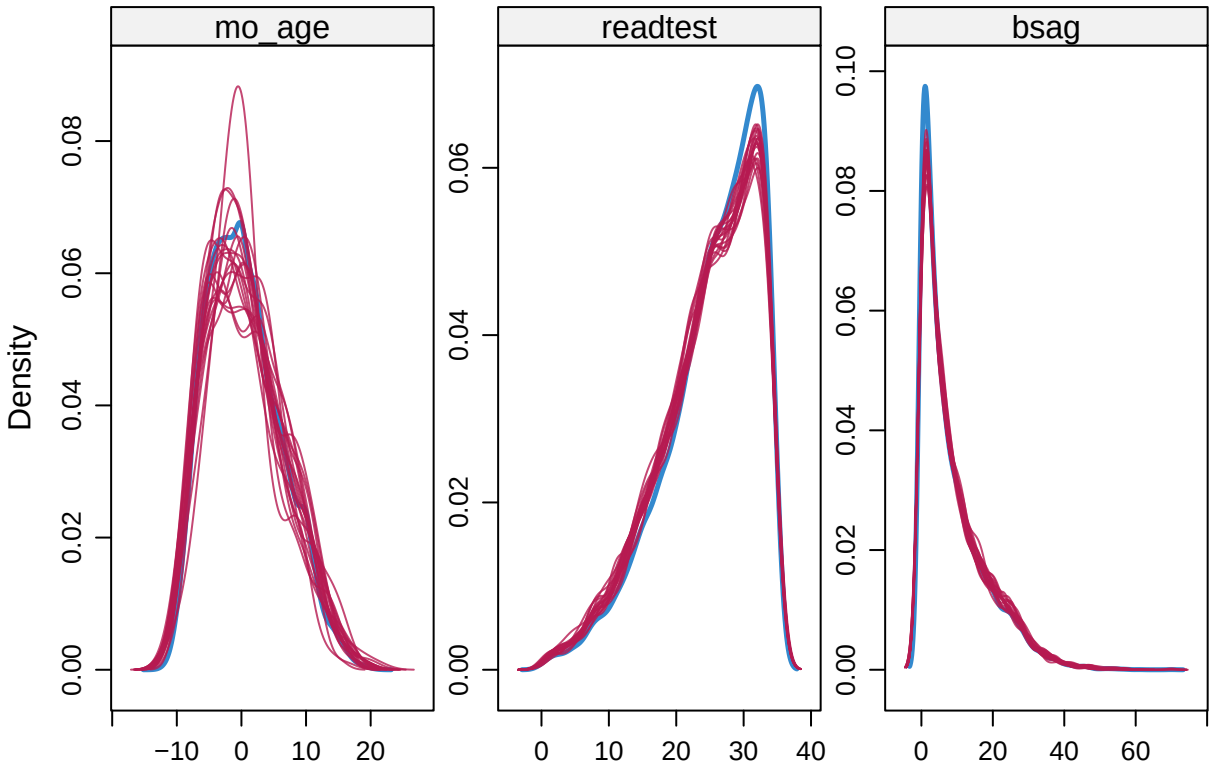
```
plot(imp, layout = c(3, 3))
```





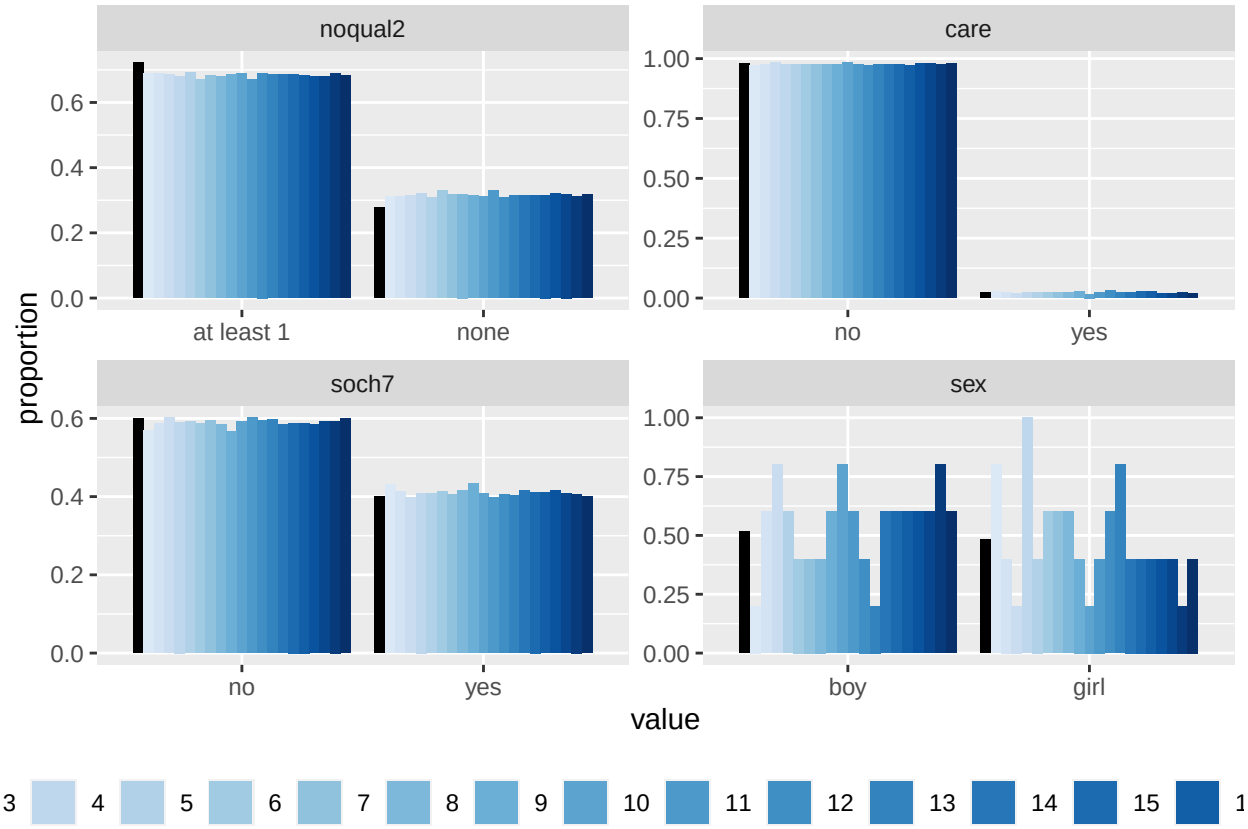
All seems good in what regards convergence of the chains of the different variables. In practice, more iterations should be done.. We can now inspect if the distribution of the imputed values agrees with the distribution of the observed ones.

```
densityplot(imp)
```



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

```
propplot(imp)
```



Everything looks reasonable, although there are some discrepancies between the observed and imputed values for sex, but because this variable has only 5 missing values, it is not too problematic. Having confirmed that our imputation step was successful, we can proceed to the analysis of the imputed data and fit our substantive model of interest.

```
fits <- with(imp, glm(noqual2 ~ care + soch7 + mo_age, family = "binomial"))
```

At this stage, we should also check that our model is fitting well the (completed) data. Diagnostic checks for logistic regression are a bit tricky, and so, only for the sake of the example, I will be skipping them here but, in practice, we should always look at a few of the completed/imputed datasets and check our assumptions.

And finally we can pool the results.

```
pool_ests <- pool(fits)
pool_ests
```

```
## Class: mipo    m = 20
##          term  m     estimate         ubar            b            t dfcom
## 1 (Intercept) 20 -1.341544853 5.831065e-04 3.971890e-04 1.000155e-03 17627
## 2     careyes 20  1.166407884 1.136148e-02 7.886158e-03 1.964194e-02 17627
## 3    soch7yes 20  0.935953710 1.163858e-03 7.470561e-04 1.948267e-03 17627
```

```
## 4      mo_age 20 -0.008338709 8.870124e-06 2.901752e-06 1.191696e-05 17627
##          df       riv    lambda       fmi
## 1 108.1238 0.7152184 0.4169839 0.4274770
## 2 105.7992 0.7288194 0.4215706 0.4322035
## 3 115.9197 0.6739732 0.4026189 0.4126657
## 4 284.3600 0.3434945 0.2556724 0.2608529
```

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

```
##          term     estimate   std.error  statistic       df      p.value
## 1 (Intercept) -1.341544853 0.031625227 -42.420086 108.1238 0.000000e+00
## 2     careyes  1.166407884 0.140149721   8.322584 105.7992 3.266276e-13
## 3    soch7yes  0.935953710 0.044139173  21.204605 115.9197 0.000000e+00
## 4      mo_age -0.008338709 0.003452096  -2.415550 284.3600 1.634203e-02
##          2.5 %       97.5 %
## 1 -1.40423073 -1.278858979
## 2  0.88854134  1.444274425
## 3  0.84852988  1.023377542
## 4 -0.01513361 -0.001543806
```

```
df <- data.frame("Estimate" = summary[,2],
                 "lq" = summary[,7],
                 "uq" = summary[,8]
                 )

rownames(df) <- c("$\\beta_0$", "$\\beta_1$","$\\beta_2$", "$\\beta_3$")
colnames(df) <- c("Estimate", "2.5% quantile", "97.5% quantile")
knitr::kable(df, escape = FALSE, digits = 3,
             caption = "Regression coefficient estimates and corresponding 95% CI")
```

Table 1: Regression coefficient estimates and corresponding 95% CI

	Estimate	2.5% quantile	97.5% quantile
β_0	-1.342	-1.404	-1.279
β_1	1.166	0.889	1.444
β_2	0.936	0.849	1.023
β_3	-0.008	-0.015	-0.002

The (pooled) regression coefficient estimates and corresponding 95% CIs are given in Table 1. Note that we should try different seeds and if results/conclusions do change by a large extent we should increase our value of M. Given the amount of missing data we have in this problem, I definitely think we should use $M = 50$ or $M = 100$. Because we have a quite large sample size, running the procedure with such large values of M takes a considerable amount of time. The perils of working with real data!!! Because here the goal is only to practice multiple imputation and we are not drawing scientific conclusions, it is fine, but we should keep in mind that in practice a larger value of M was needed.