

An Example of mi Usage

Ben Goodrich and Jonathan Kropko, for this version, based on earlier versions written by Yu-Sung Su, I

06/16/2014

There are several steps in an analysis of missing data. Initially, users must get their data into R. There are several ways to do so, including the `read.table`, `read.csv`, `read.fwf` functions plus several functions in the **foreign** package. All of these functions will generate a `data.frame`, which is a bit like a spreadsheet of data. <http://cran.r-project.org/doc/manuals/R-data.html> for more information.

```
options(width = 65)
suppressMessages(library(mi))
data(nlsyV, package = "mi")
```

From there, the first step is to convert the `data.frame` to a `missing_data.frame`, which is an enhanced version of a `data.frame` that includes metadata about the variables that is essential in a missing data context.

```
mdf <- missing_data.frame(nlsyV)
```

```
## NOTE: In the following pairs of variables, the missingness pattern of the first is a subset of the second.
## Please verify whether they are in fact logically distinct variables.
##      [,1]      [,2]
## [1,] "b.marr" "income"
```

The `missing_data.frame` constructor function creates a `missing_data.frame` called `mdf`, which in turn contains seven `missing_variables`, one for each column of the `nlsyV` dataset.

The most important aspect of a `missing_variable` is its class, such as `continuous`, `binary`, and `count` among many others (see the table in the Slots section of the help page for `missing_variable-class`). The `missing_data.frame` constructor function will try to guess the appropriate class for each `missing_variable`, but rarely will it correspond perfectly to the user's intent. Thus, it is very important to call the `show` method on a `missing_data.frame` to see the initial guesses

```
show(mdf) # momrace is guessed to be ordered
```

```
## Object of class missing_data.frame with 400 observations on 7 variables
##
## There are 20 missing data patterns
##
## Append '@patterns' to this missing_data.frame to access the corresponding pattern for every observation
##
##           type missing method  model
## ppvtr.36   continuous      75   ppd linear
## first      binary         0   <NA> <NA>
## b.marr     binary        12   ppd logit
## income    continuous     82   ppd linear
## momage     continuous      0   <NA> <NA>
## momed     ordered-categorical 40   ppd ologit
## momrace   ordered-categorical 117  ppd ologit
##
```

```
##          family    link transformation
## ppvtr.36 gaussian identity    standardize
## first      <NA>    <NA>          <NA>
## b.marr     binomial  logit        <NA>
## income     gaussian identity    standardize
## momage     <NA>    <NA>          standardize
## momed     multinomial  logit        <NA>
## momrace   multinomial  logit        <NA>
```

and to modify them, if necessary, using the `change` function, which can be used to change many things about `amissing_variable`, so see its help page for more details. In the example below, we change the class of the `momrace` (race of the mother) variable from the initial guess of `ordered-categorical` to a more appropriate `unordered-categorical` and change the `income` `nonnegative-continuous`.

```
mdf <- change(mdf, y = c("income", "momrace"), what = "type",
              to = c("non", "un"))
```

```
## NOTE: In the following pairs of variables, the missingness pattern of the first is a subset of the second.
## Please verify whether they are in fact logically distinct variables.
```

```
##      [,1]      [,2]
## [1,] "b.marr" "income"
```

```
show(mdf)
```

```
## Object of class missing_data.frame with 400 observations on 7 variables
```

```
##
```

```
## There are 20 missing data patterns
```

```
##
```

```
## Append '@patterns' to this missing_data.frame to access the corresponding pattern for every observation.
```

```
##
```

```
##          type missing method  model
## ppvtr.36      continuous    75   ppd linear
## first         binary         0   <NA> <NA>
## b.marr         binary        12   ppd logit
## income        nonnegative-continuous  82   ppd linear
## income:is_zero binary        82   ppd logit
## momage         continuous     0   <NA> <NA>
## momed          ordered-categorical   40   ppd ologit
## momrace        unordered-categorical 117   ppd mlogit
```

```
##
```

```
##          family    link transformation
## ppvtr.36 gaussian identity    standardize
## first      <NA>    <NA>          <NA>
## b.marr     binomial  logit        <NA>
## income     gaussian identity    logshift
## income:is_zero binomial  logit        <NA>
## momage     <NA>    <NA>          standardize
## momed     multinomial  logit        <NA>
## momrace   multinomial  logit        <NA>
```

Once all of the `missing_variables` are set appropriately, it is useful to get a sense of the raw data, which can be accomplished by looking at the `summary`, `image`, and / or `hist` of a `missing_data.frame`

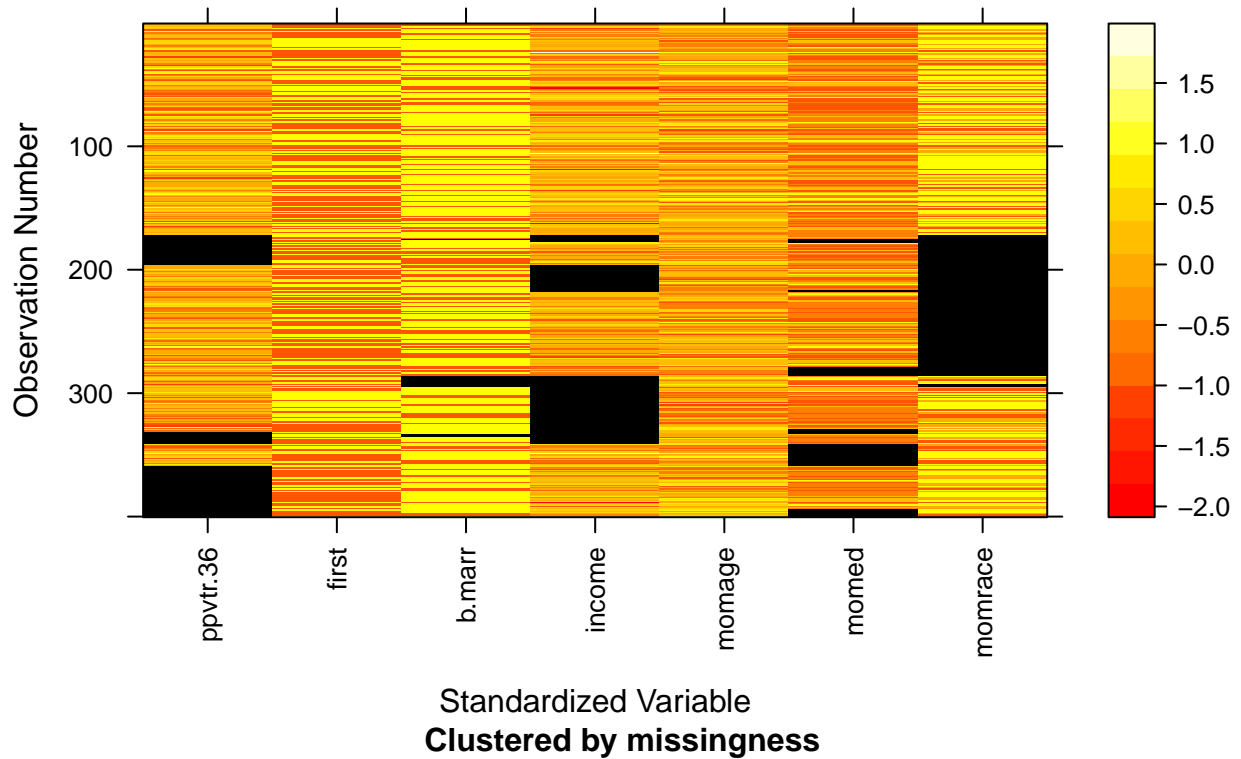
```
summary(mdf)
```

```
##      ppvtr.36      first      b.marr
## Min.   : 41.00   Min.   :0.000   Min.   :0.0000
```

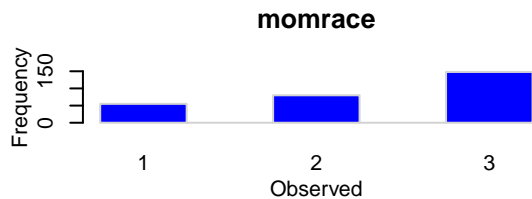
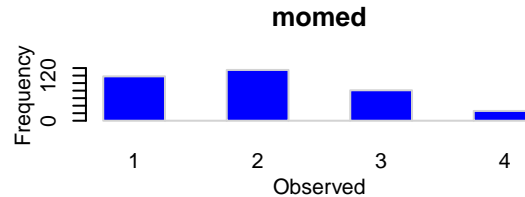
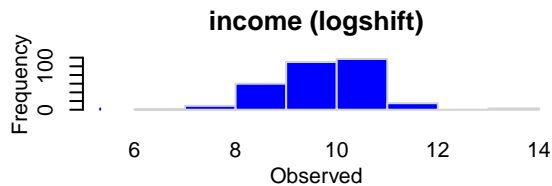
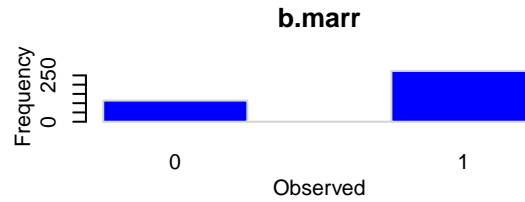
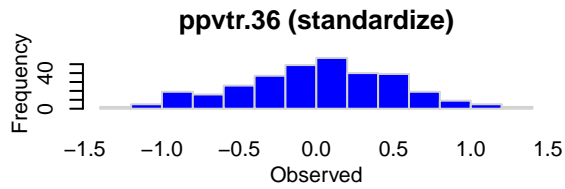
```
## 1st Qu.: 74.00 1st Qu.:0.000 1st Qu.:0.0000
## Median : 87.00 Median :0.000 Median :1.0000
## Mean : 85.94 Mean :0.435 Mean :0.7062
## 3rd Qu.: 99.00 3rd Qu.:1.000 3rd Qu.:1.0000
## Max. :132.00 Max. :1.000 Max. :1.0000
## NA's :75 NA's :12
## income momage momed momrace
## Min. : 0 Min. :16.00 Min. :1.000 1 : 55
## 1st Qu.: 8590 1st Qu.:22.00 1st Qu.:1.000 2 : 80
## Median : 17906 Median :24.00 Median :2.000 3 :148
## Mean : 32041 Mean :23.75 Mean :2.042 NA's:117
## 3rd Qu.: 31228 3rd Qu.:26.00 3rd Qu.:3.000
## Max. :1057448 Max. :32.00 Max. :4.000
## NA's :82 NA's :40
```

```
image(mdf)
```

Dark represents missing data



```
hist(mdf)
```



Next we use the `mi` function to do the actual imputation, which has several extra arguments that, for example, govern how many independent chains to utilize, how many iterations to conduct, and the maximum amount of time the user is willing to wait for all the iterations of all the chains to finish. The imputation step can be quite time consuming, particularly if there are many `missing_variables` and if many of them are categorical. One important way in which the computation time can be reduced is by imputing in parallel, which is highly recommended and is implemented in the `mi` function by default on non-Windows machines. If users encounter problems running `mi` with parallel processing, the problems are likely due to the machine exceeding available RAM. Sequential processing can be used instead for `mi` by using the `parallel=FALSE` option.

```
rm(nlsyV) # good to remove large unnecessary objects to save RAM
options(mc.cores = 2)
imputations <- mi(mdf, n.iter = 30, n.chains = 4, max.minutes = 20)
show(imputations)
```

Object of class `mi` with 4 chains, each with 30 iterations.

Each chain is the evolution of an object of `missing_data.frame` class with 400 observations on 7 variables.

The next step is very important and essentially verifies whether enough iterations were conducted. We want the mean of each completed variable to be roughly the same for each of the 4 chains.

```
round(mipply(imputations, mean, to.matrix = TRUE), 3)
```

```
##          chain:1 chain:2 chain:3 chain:4
## ppvtr.36      0.003  0.004  0.010  0.007
## first         1.435  1.435  1.435  1.435
## b.marr        1.685  1.688  1.685  1.685
## income        9.547  9.564  9.512  9.515
## momage         0.000  0.000  0.000  0.000
## momed         2.053  2.053  2.067  2.053
## momrace       2.292  2.277  2.260  2.283
## missing_ppvtr.36 0.188  0.188  0.188  0.188
## missing_b.marr  0.030  0.030  0.030  0.030
```

```
## missing_income      0.205  0.205  0.205  0.205
## missing_momed       0.100  0.100  0.100  0.100
## missing_momrace     0.292  0.292  0.292  0.292
```

```
Rhats(imputations)
```

```
## mean_ppvtr.36      mean_b.marr      mean_income      mean_momed
##      0.9989349      1.0184656      1.0959858      1.0009582
## mean_momrace       sd_ppvtr.36      sd_b.marr         sd_income
##      1.1251871      1.0055942      1.0167196      1.1231192
##      sd_momed       sd_momrace
##      0.9939488      1.2023858
```

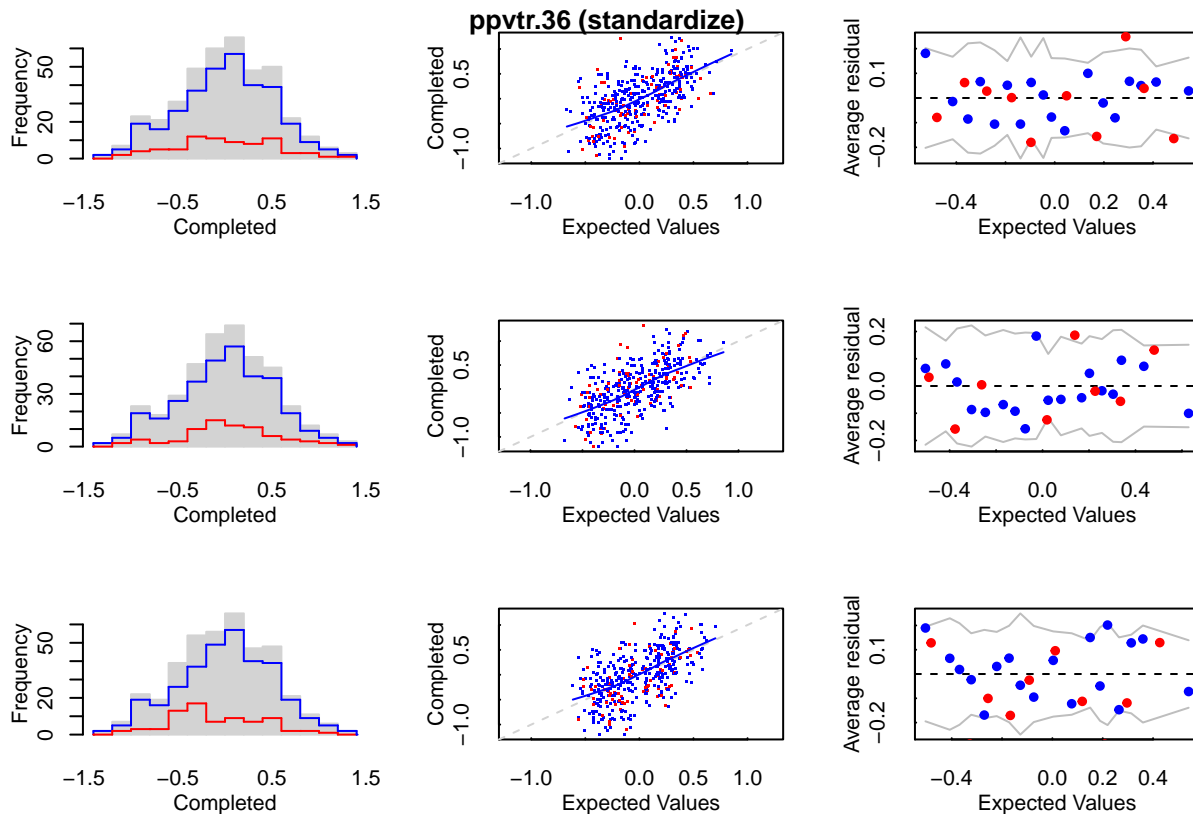
If so — and when it does in the example depends on the pseudo-random number seed — we can proceed to diagnosing other problems. For the sake of example, we continue our 4 chains for another 5 iterations by calling

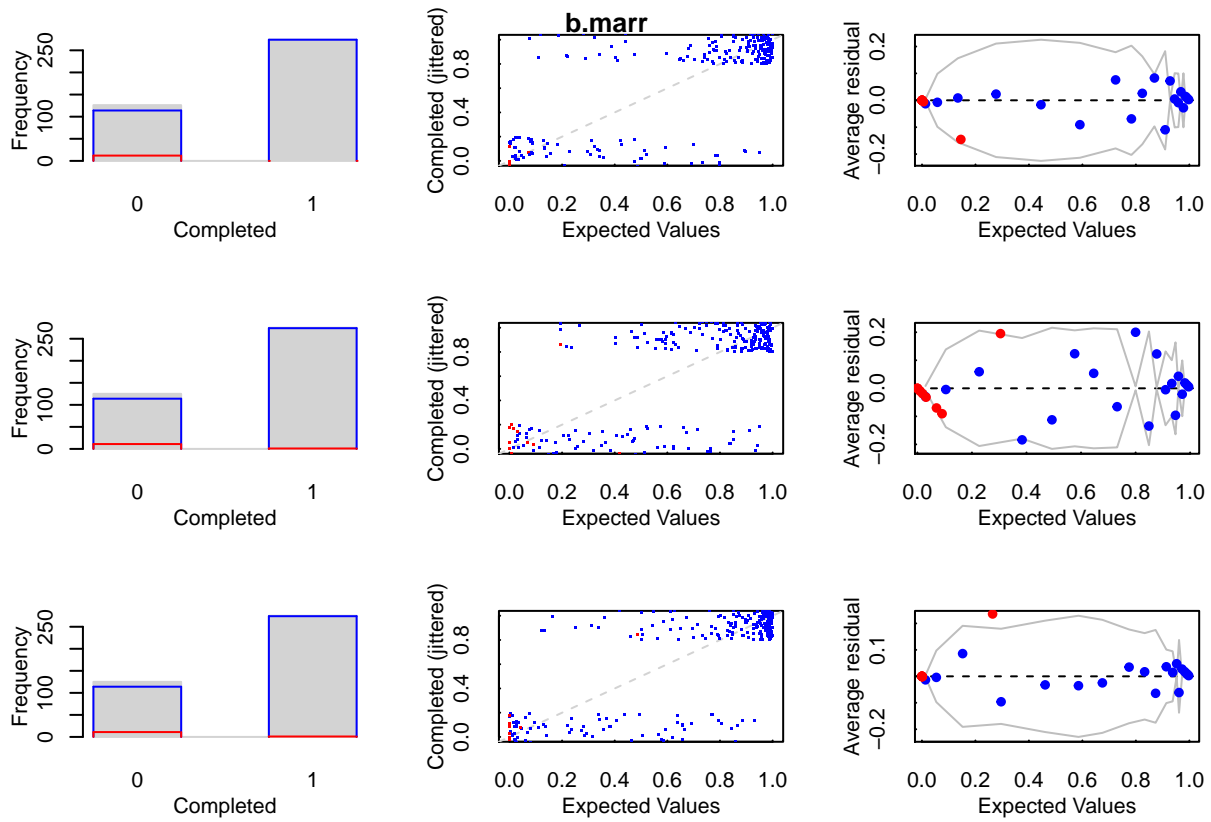
```
imputations <- mi(imputations, n.iter = 5)
```

to illustrate that this process can be continued until convergence is reached.

Next, the `plot` of an object produced by `mi` displays, for all `missing_variables` (or some subset thereof), a histogram of the observed, imputed, and completed data, a comparison of the completed data to the fitted values implied by the model for the completed data, and a plot of the associated binned residuals. There will be one set of plots on a page for the first three chains, so that the user can get some sense of the sampling variability of the imputations. The `hist` function yields the same histograms as `plot`, but groups the histograms for all variables (within a chain) on the same plot. The `imagefunction` gives a sense of the missingness patterns in the data.

```
plot(imputations)
```





```

## Warning: algorithm did not converge

## Warning in .local(y, data, ...): bayesglm() did not converge for
## variable income:is_zero on iteration 36

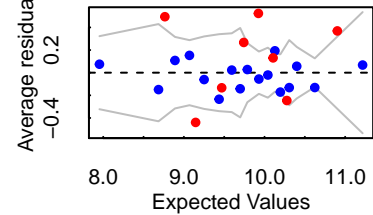
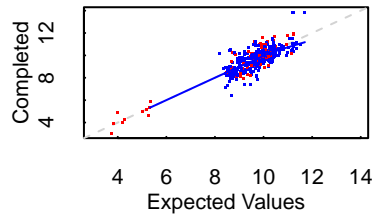
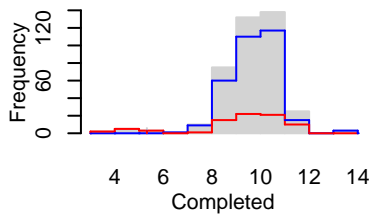
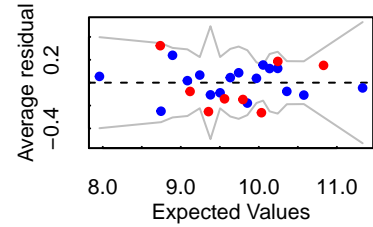
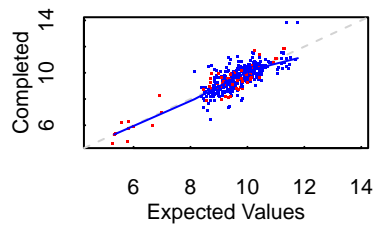
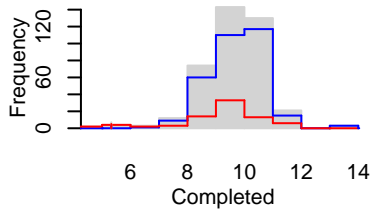
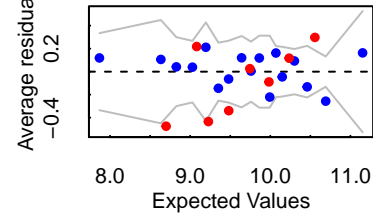
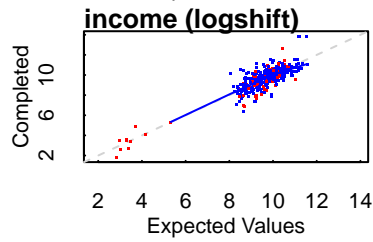
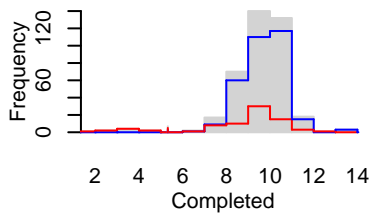
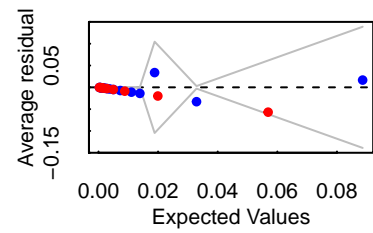
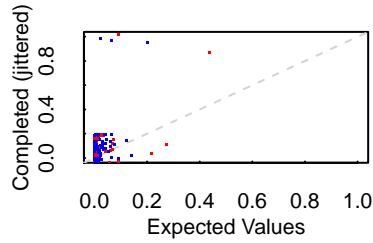
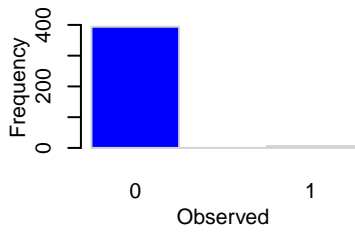
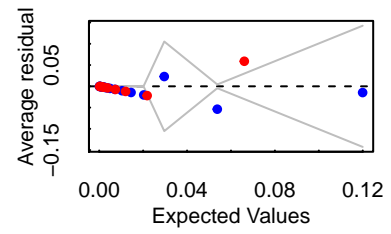
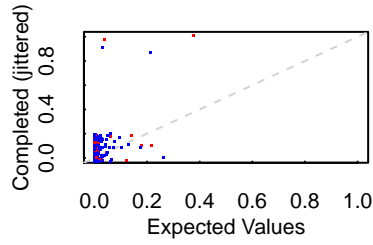
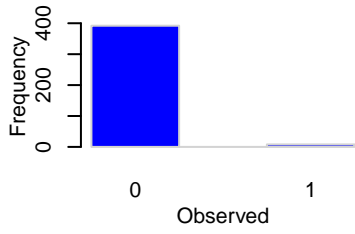
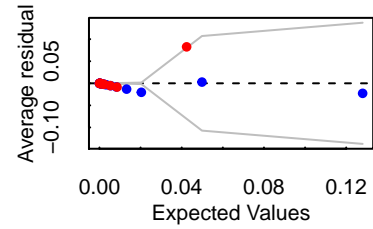
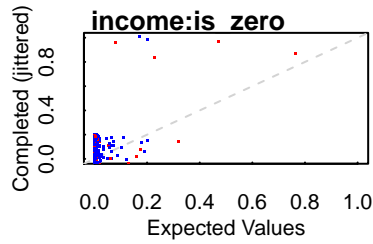
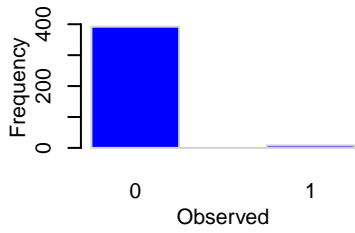
## Warning: algorithm did not converge

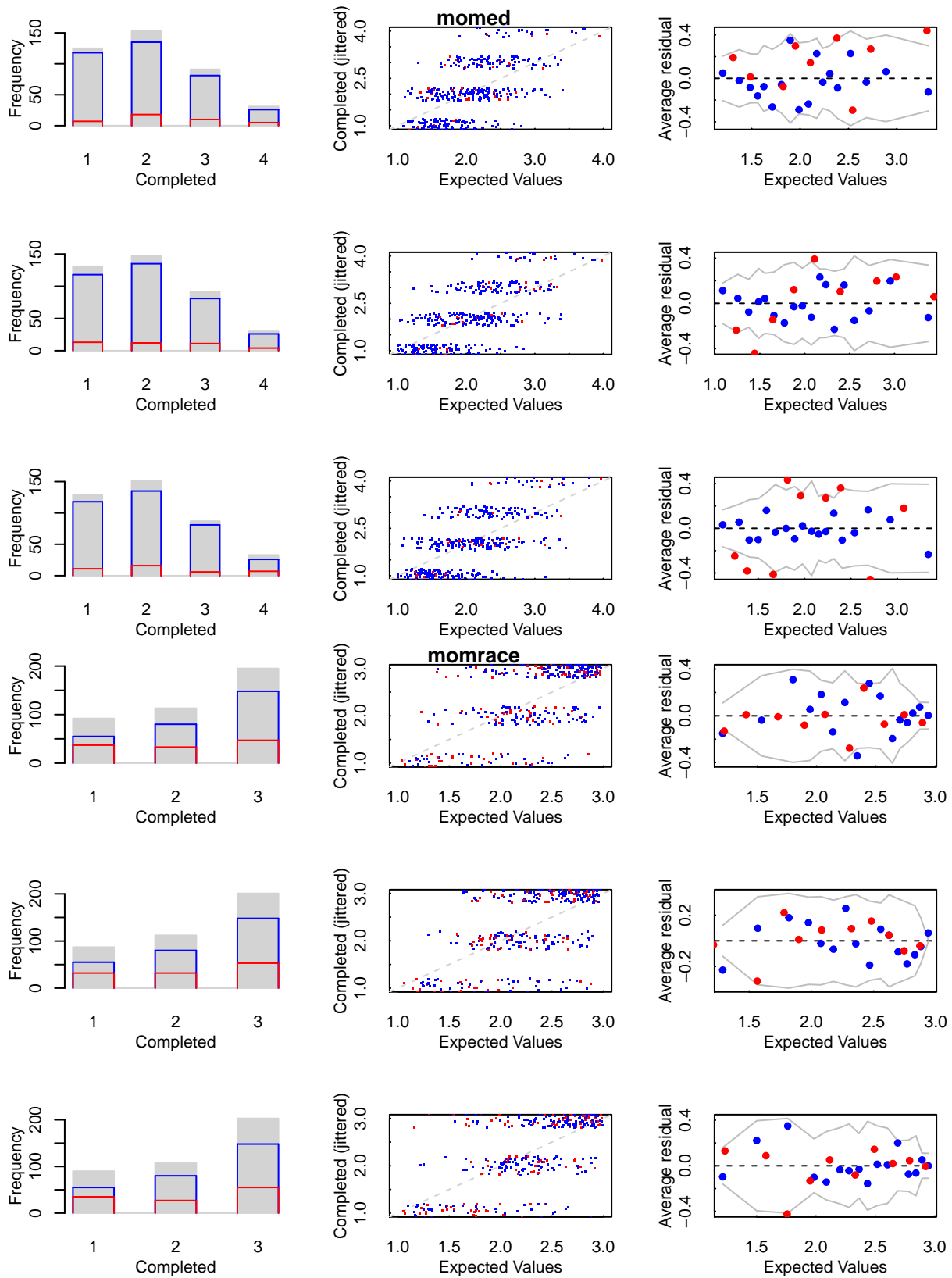
## Warning in .local(y, data, ...): bayesglm() did not converge for
## variable income:is_zero on iteration 36

## Warning: algorithm did not converge

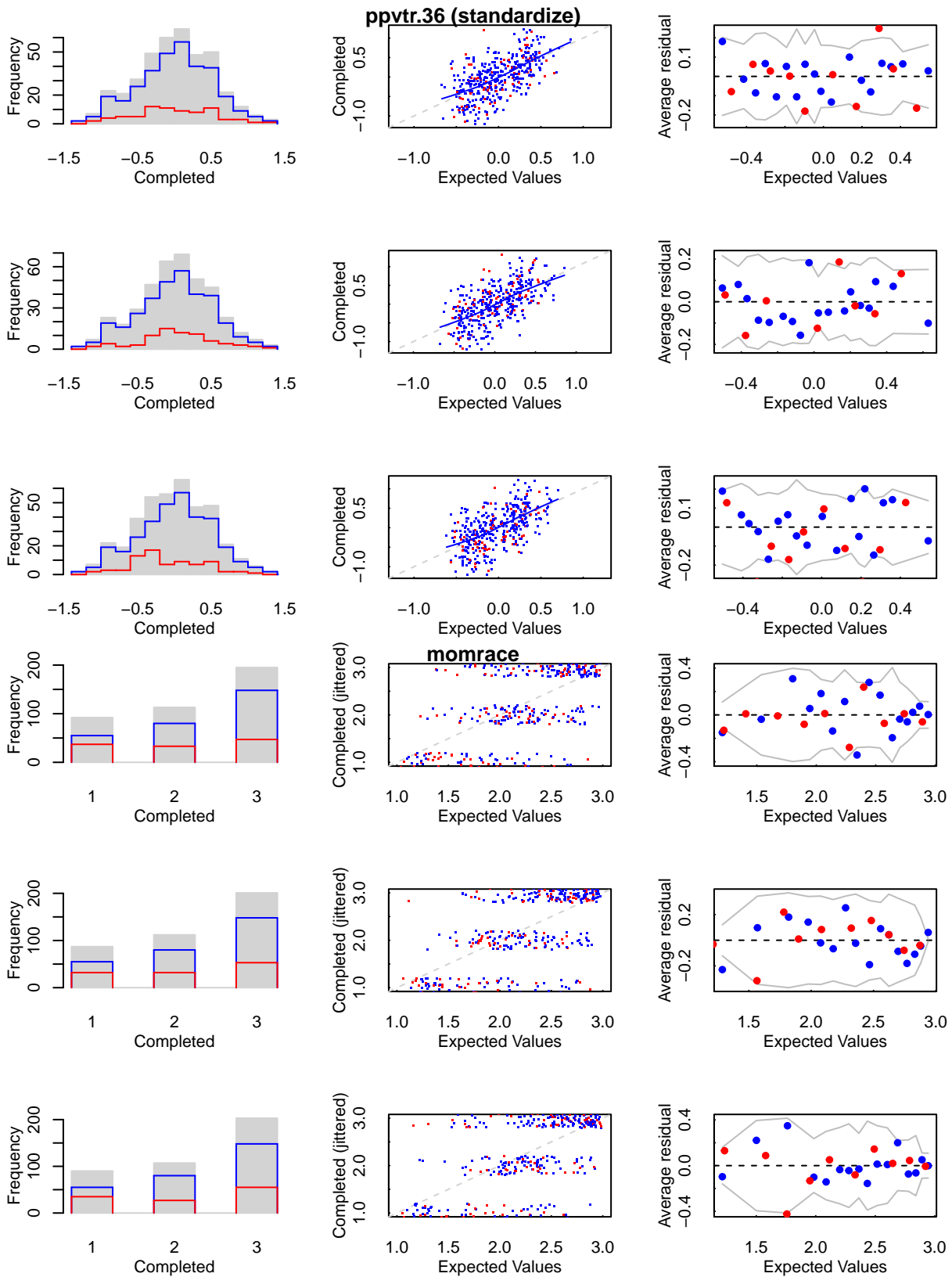
## Warning in .local(y, data, ...): bayesglm() did not converge for
## variable income:is_zero on iteration 36

```

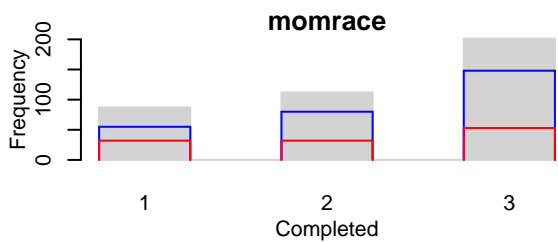
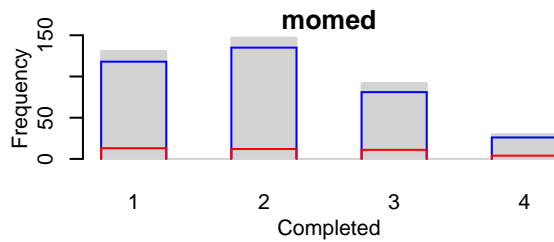
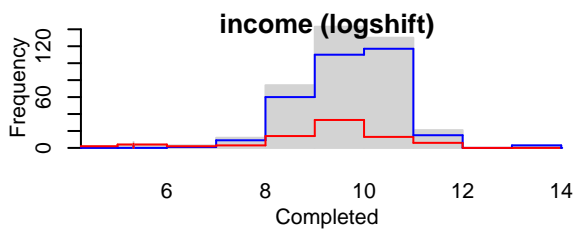
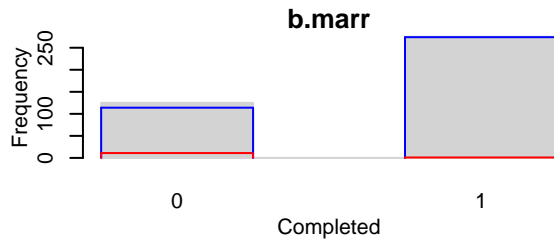
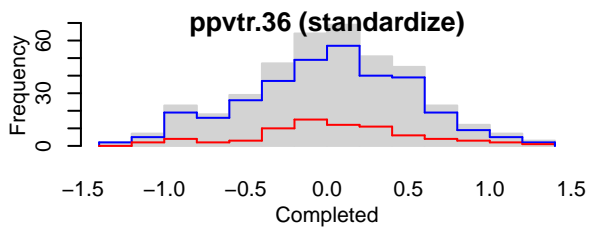
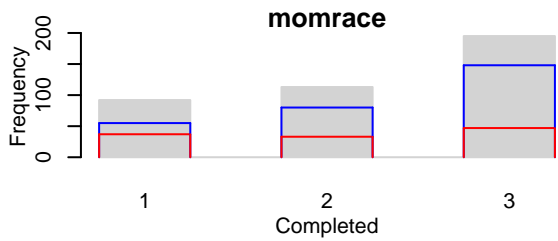
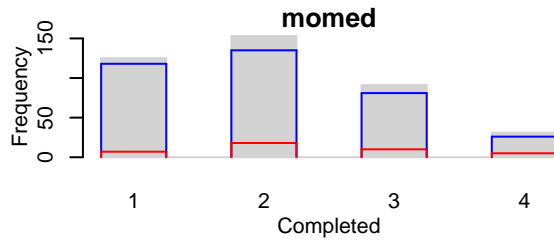
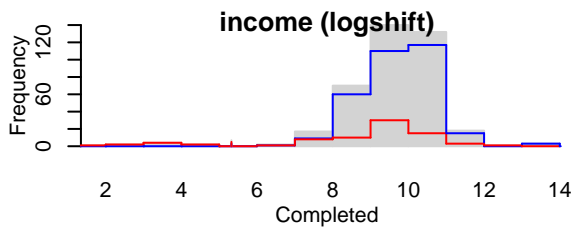
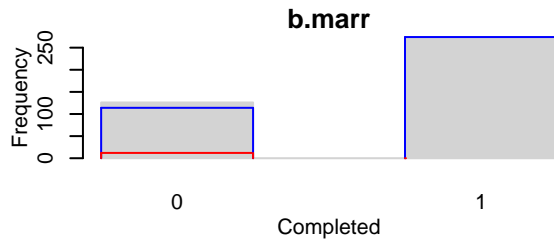
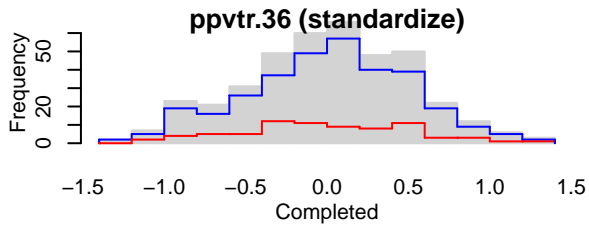


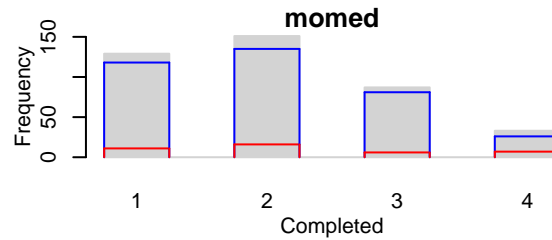
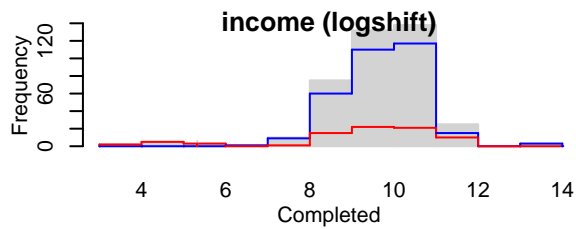
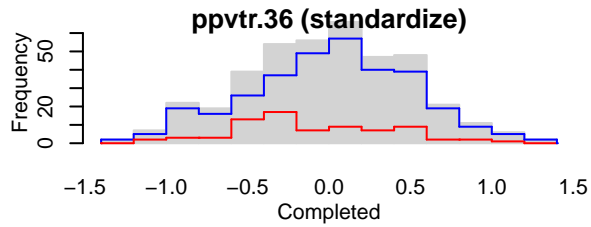


```
plot(imputations, y = c("ppvtr.36", "momrace"))
```

```
hist(imputations)
```





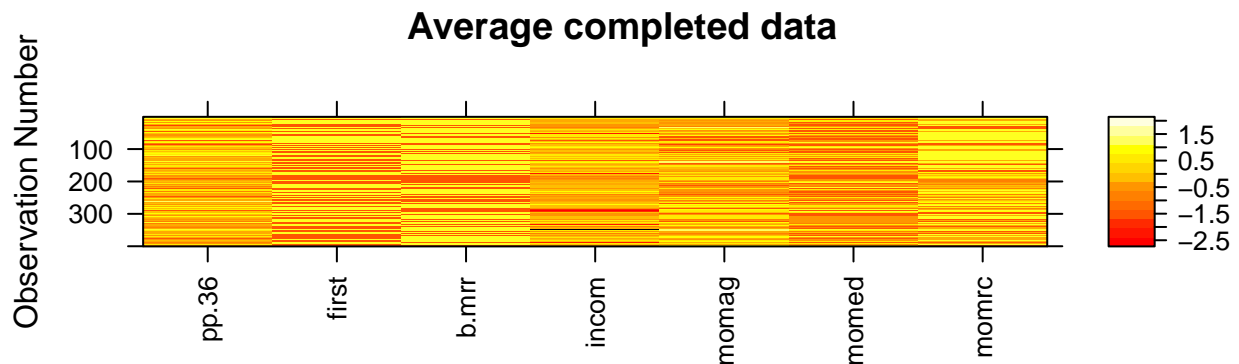
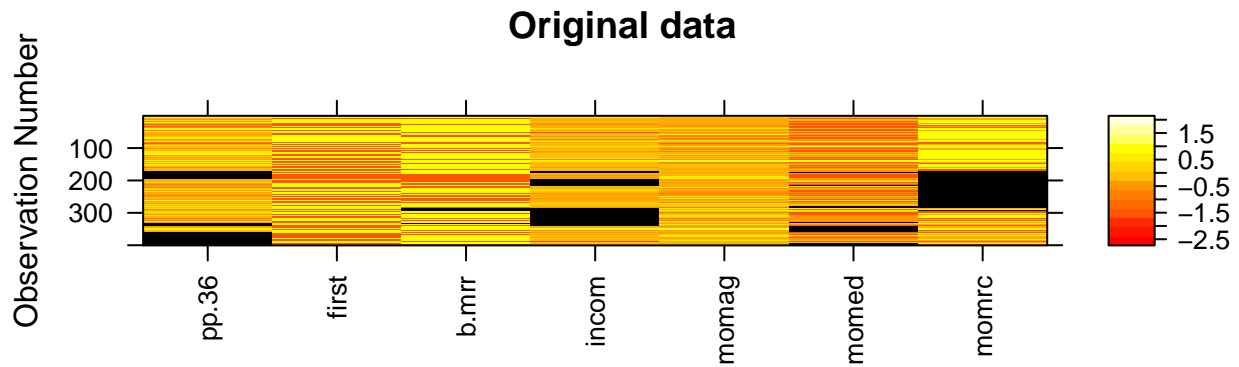
```
image(imputations)
summary(imputations)
```

```
## $ppvtr.36
## $ppvtr.36$is_missing
## missing
## FALSE TRUE
## 325 75
##
## $ppvtr.36$imputed
##   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.36038 -0.37383 -0.05496 -0.02295 0.35011 1.32727
##
## $ppvtr.36$observed
##   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.20189 -0.31924 0.02848 0.00000 0.34944 1.23210
##
##
## $first
## $first$is_missing
## [1] "all values observed"
##
## $first$observed
##
## 1 2
## 226 174
##
##
```

```

## $b.marr
## $b.marr$crosstab
##
##      observed imputed
##  0         456      45
##  1         1096      3
##
##
## $income
## $income$is_missing
## missing
## FALSE TRUE
##   318   82
##
## $income$imputed
##   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  1.827  8.439   9.451   8.895 10.122 12.665
##
## $income$observed
##   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  5.323  9.079   9.804   9.699 10.358 13.872
##
##
## $momage
## $momage$is_missing
## [1] "all values observed"
##
## $momage$observed
##   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.21377 -0.27468  0.03835  0.00000  0.35137  1.29046
##
##
## $momed
## $momed$crosstab
##
##      observed imputed
##  1         472      43
##  2         540      60
##  3         324      35
##  4         104      22
##
##
##
## $momrace
## $momrace$crosstab
##
##      observed imputed
##  1         220      141
##  2         320      122
##  3         592      205

```



Finally, we pool over $m = 5$ imputed datasets – pulled from across the 4 chains – in order to estimate a descriptive linear regression of test scores (*ppvtr.36*) at 36 months on a variety of demographic variables pertaining to the mother of the child.

```
analysis <- pool(ppvtr.36 ~ first + b.marr + income + momage + momed + momrace,
                data = imputations, m = 5)
display(analysis)
```

```
## bayesglm(formula = ppvtr.36 ~ first + b.marr + income + momage +
##          momed + momrace, data = imputations, m = 5)
##           coef.est coef.se
## (Intercept) 80.79      7.57
## first1      4.26      1.72
## b.marr1     5.47      2.47
## income      0.00      0.00
## momage     -0.12      0.31
## momed.L    10.72      3.11
## momed.Q     1.08      2.35
## momed.C     0.85      1.76
## momrace2   -6.16      3.69
## momrace3   11.75      2.92
## n = 390, k = 10
## residual deviance = 93528.0, null deviance = 142095.8 (difference = 48567.8)
## overdispersion parameter = 239.8
## residual sd is sqrt(overdispersion) = 15.49
```

The rest is optional and only necessary if you want to perform some operation that is not supported by the **mi** package, perhaps outside of R. Here we create a list of **data.frames**, which can be saved to the hard disk and / or exported in a variety of formats with the **foreign** package. Imputed data can be exported to Stata by using the **mi2stata** function instead of **complete**.

```
dfs <- complete(imputations, m = 2)
```