SISCER 2023: Missing Data, NHANES Data

Katie Wilson, Anna Plantinga

Contents

This file provides an analysis in `R` of a subset of the NHANES 2009-2012 survey data. We'll focus on modeling relationships between different physical measurements while accounting for missingness in some of the variables.

Data Overview

NHANES (American National Health and Nutrition Examination surveys) uses complex survey sampling designs in selecting individuals to be part of the study. We will work with a resampled version of NHANES that, for our purposes, we will treat as a random sample from the US population. Specifically, we will use data obtained from the `R` package `NHANES`– more information can be found here `?data(NHANES)`.

Our subset includes female participants ages 20-44 years who were not pregnant at time of the health examination. Ultimately, we have 1,503 observations.

Variables

- `ID`: participant ID
- `SurveyYr`: survey
- `TotChol`: total cholesterol in mmol/L. This will be our outcome of interest
- `Age`: age in years
- `Pulse`: 60 second pulse rate
- `BPSysAve`: systolic blood pressure in mm Hg
- `Diabetes`: whether participant was told by a doctor or health professional that they have diabetes

```r
nha <- read.csv("Datasets/nhanes_subset.csv")
head(nha)
```

```
##      ID SurveyYr TotChol Age Pulse BPSysAve
## 1 51710  2009_10    5.95  26    94      106
## 2 51724  2009_10    4.32  37    82      102
## 3 51731  2009_10    3.36  28    60      103
## 4 51741  2009_10    4.24  21    86      120
## 5 51741  2009_10    4.24  21    86      120
## 6 51760  2009_10    4.09  27    90      110
```

R Packages

```
library(tidyverse)  # for data manipulation, etc. (e.g., the "pipe" %>%)
library(GGally)  # for scatterplot matrix
library(gridExtra)  # for combining graphs into a grid "by hand"
library(mice)  # multiple imputation; visualization with md.pattern()
library(VIM)    # visualization with aggr()
library(lavaan)  # maximum likelihood
library(lmtest)   # for inference with robust SE
library(sandwich)   # for calculation of robust SE
library(drgee)   # for doubly robust estimation
```

Exploratory Analyses

First, we'll obtain some univariate descriptives:

```
summary(nha)
```

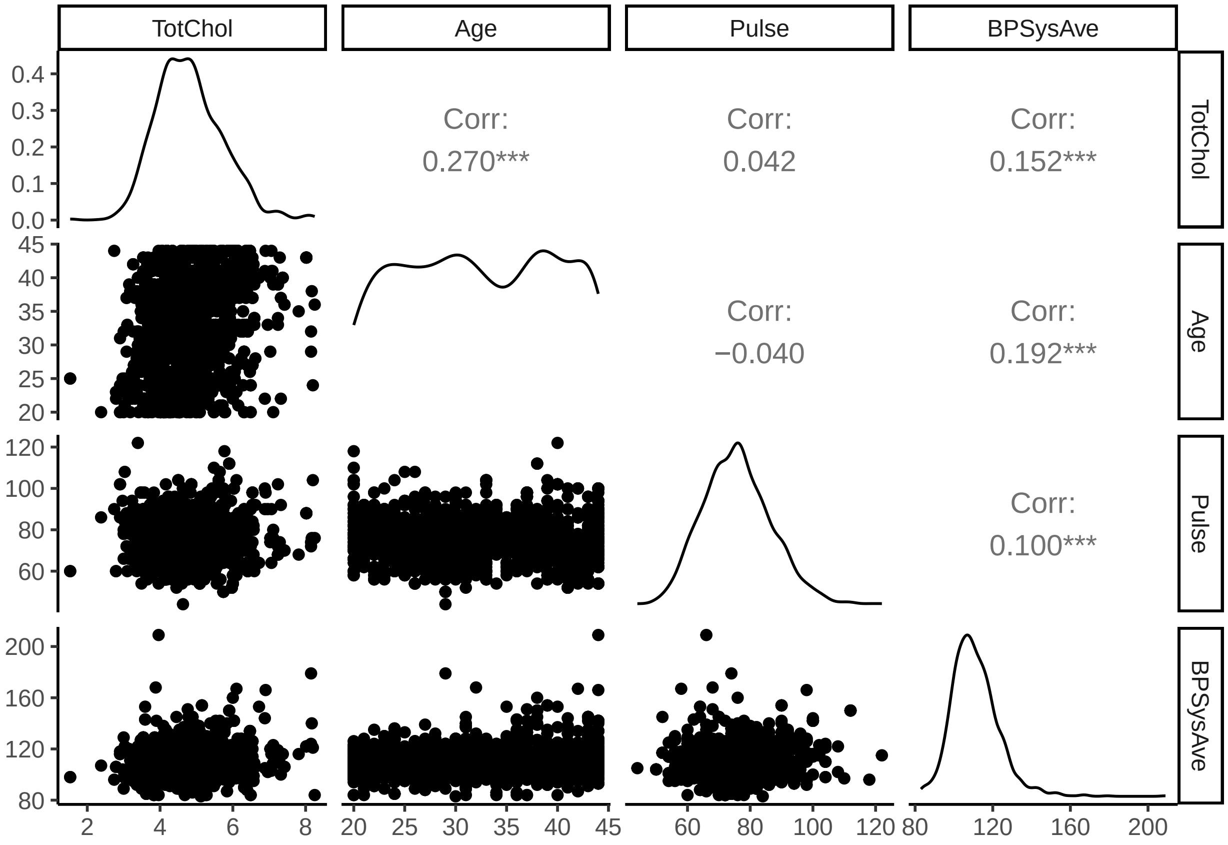
```
##        ID          SurveyYr            TotChol           Age       
##  Min.   :51710   Length:1503        Min.   :1.530   Min.   :20.00  
##  1st Qu.:56444   Class :character   1st Qu.:4.160   1st Qu.:26.00  
##  Median :61635   Mode  :character   Median :4.730   Median :32.00  
##  Mean   :61668                      Mean   :4.804   Mean   :32.38  
##  3rd Qu.:66614                      3rd Qu.:5.380   3rd Qu.:39.00  
##  Max.   :71873                      Max.   :8.250   Max.   :44.00  
##                                                                    
##      Pulse          BPSysAve    
##  Min.   : 44.0   Min.   : 83.0  
##  1st Qu.: 68.0   1st Qu.:103.0  
##  Median : 76.0   Median :109.5  
##  Mean   : 75.8   Mean   :110.9  
##  3rd Qu.: 82.0   3rd Qu.:118.0  
##  Max.   :122.0   Max.   :209.0  
##  NA's   :73      NA's   :75
```

```
table(nha$SurveyYr)
```

```
## 
## 2009_10 2011_12 
##     789     714
```

We notice that 73 (4.86%) are missing a `Pulse` measurement and 75 (4.99%) are missing a `BPSysAve` measurement. We'll look into this a bit more later. First, we'll look at some bivariate plots of the different variables. The `ggpairs()` function that we'll use appears to do pairwise deletion. For example, the scatterplot of total cholesterol vs pulse will use all observations except the 73 missing the pulse measurement.

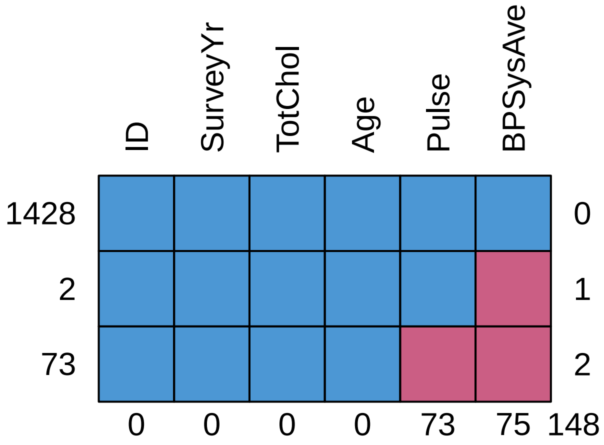
```
ggpairs(nha %>% select(-ID,-SurveyYr)) +
  theme_classic()
```



Missing Data Patterns

Now, we'll start exploring the missingness more. The `md.pattern()` function in the `mice` package offers a quick way to visualize the missing data patterns. We see that most individuals (1,428) have no missingness (top row). 2 individuals are missing only `BPSysAve` and 73 individuals are missing `Pulse` and `BPSysAve`. This is a **monotone** missing data pattern.

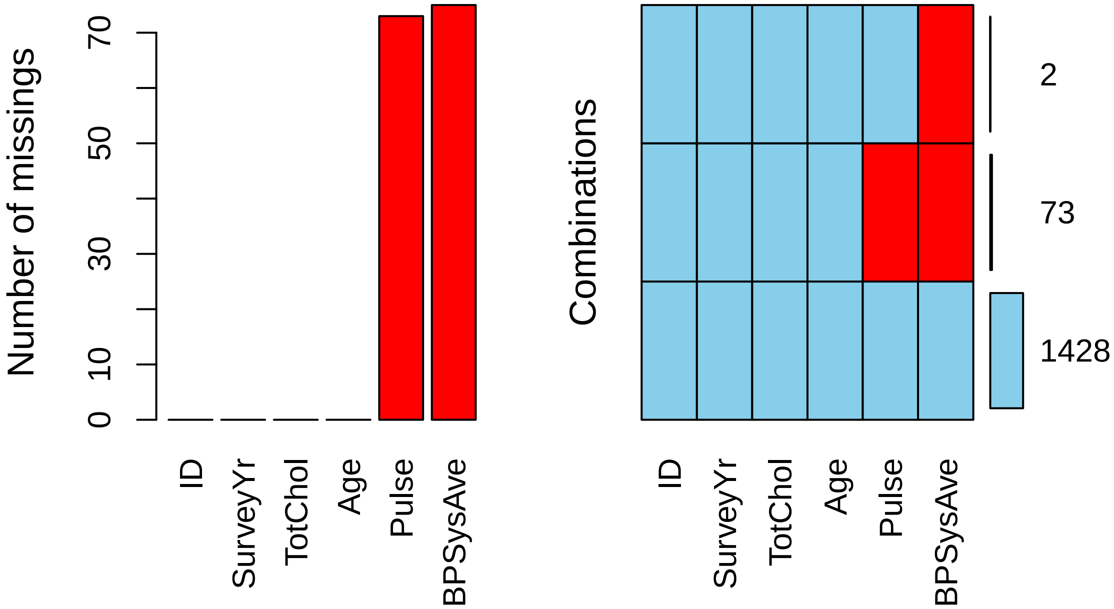
```
md.pattern(nha, plot = TRUE, rotate.names = TRUE)
```



```
##      ID SurveyYr TotChol Age Pulse BPSysAve
## 1428  1        1       1   1     1        1   0
## 2     1        1       1   1     1        0   1
## 73    1        1       1   1     0        0   2
##       0        0       0   0    73       75 148
```

There is also the `aggr()` function in the `VIM` package:

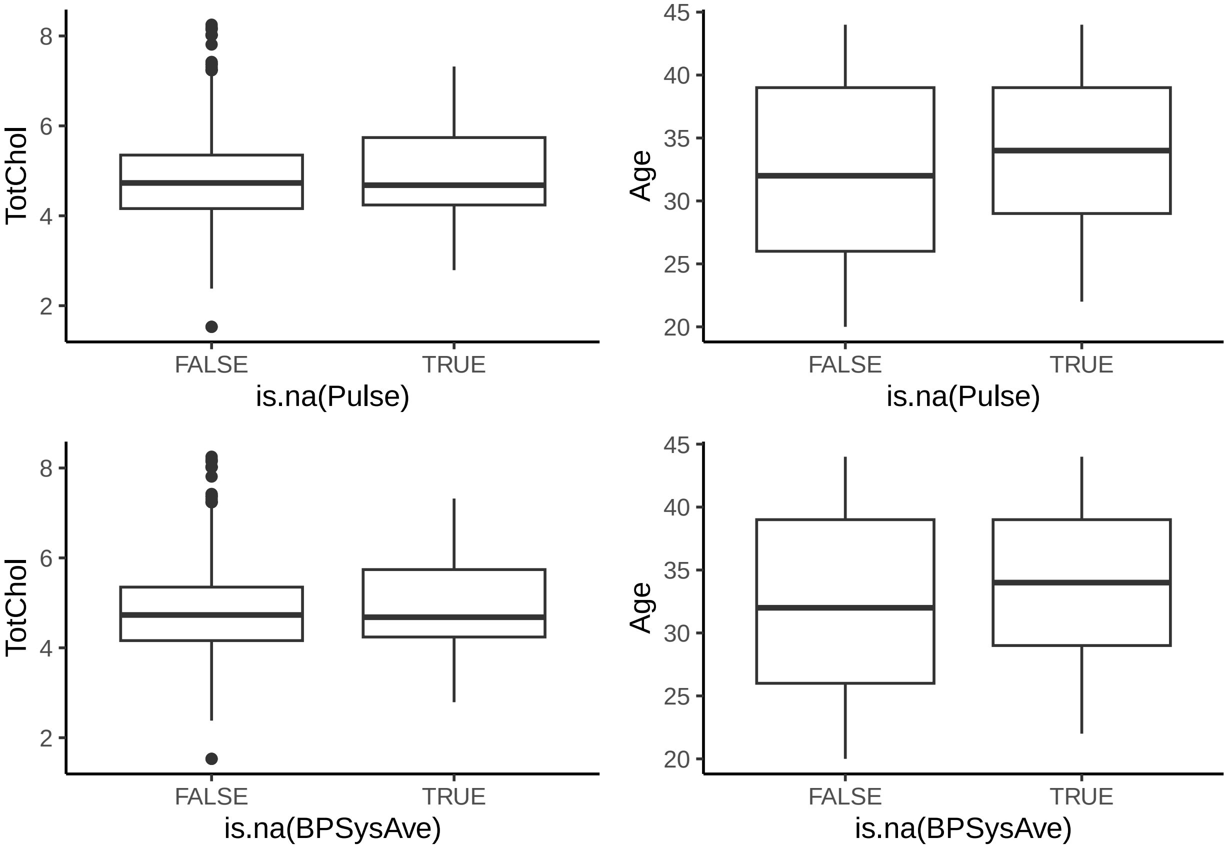
```
a <- aggr(nha, plot = F)
plot(a, numbers = T, prop = F)
```



Now, we'll look into whether missing `Pulse` or `BPSysAve` may be associated with other variables. For this, we'll create some boxplots or tables, stratifying by whether `Pulse` or `BPSysAve` is missing.

Continuous variables:

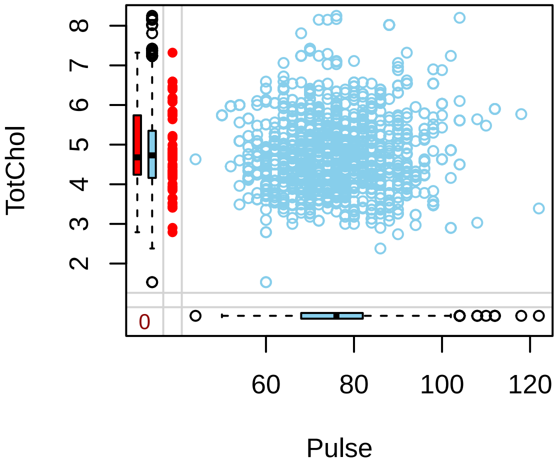
```
p1 <- ggplot(nha, aes(x = is.na(Pulse), y = TotChol)) +
  geom_boxplot() +
  theme_classic()
p2 <- ggplot(nha, aes(x = is.na(Pulse), y = Age)) +
  geom_boxplot() +
  theme_classic()
p3 <- ggplot(nha, aes(x = is.na(BPSysAve), y = TotChol)) +
  geom_boxplot() +
  theme_classic()
p4 <- ggplot(nha, aes(x = is.na(BPSysAve), y = Age)) +
  geom_boxplot() +
  theme_classic()
grid.arrange(p1, p2, p3, p4, nrow = 2)
```

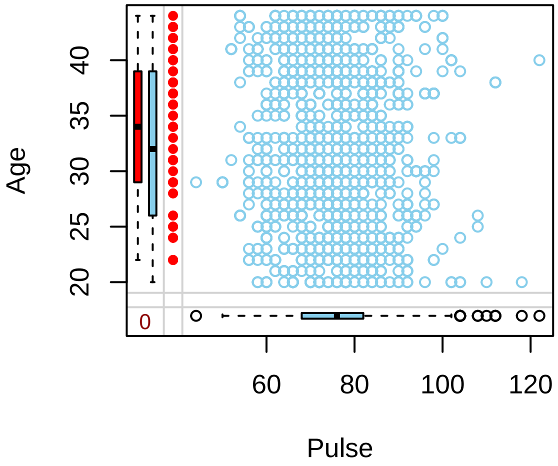


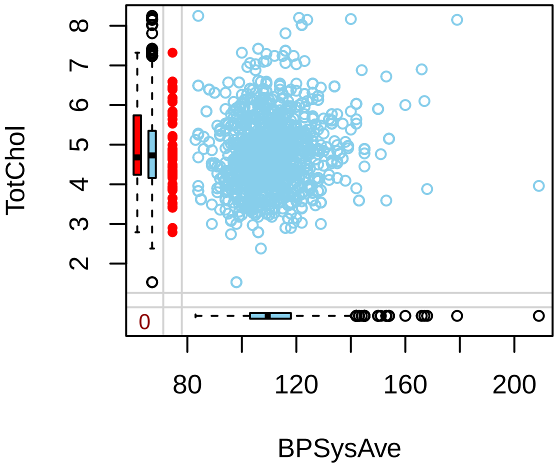
Note that there were 2 additional people missing only systolic blood pressure (and everyone missing pulse was missing systolic blood pressure). If there was more missingness here, we could also look at, among those with a pulse measurement, whether missingness of systolic blood pressure is associated with pulse.

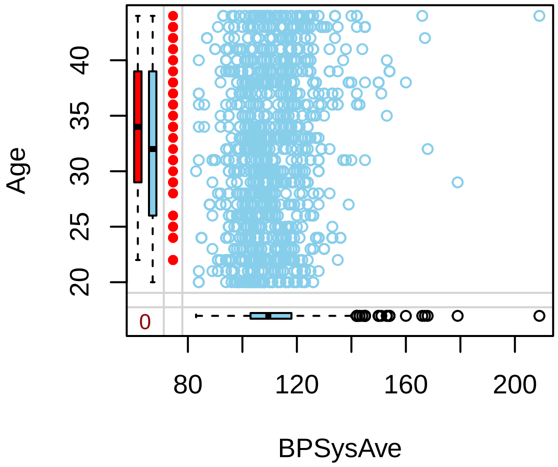
This can also be accomplished using the `marginplot()` function:

```r
par(mfrow = c(2,2))
marginplot(nha %>% select(Pulse, TotChol))
marginplot(nha %>% select(Pulse, Age))
marginplot(nha %>% select(BPSysAve, TotChol))
marginplot(nha %>% select(BPSysAve, Age))
```









From these figures, it looks like those that are missing pulse/systolic blood pressure may tend to be a bit older (comparing the medians). Based on these particular summaries, MCAR could be a reasonable assumption; however, it is important to keep in mind that there is no way from the data we have to rule out missingness depending on the missing value itself.

Ultimately, given that there is only a small amount of missingness (and it doesn't appear very correlated with our other variables – including the outcome we're modeling, total cholesterol), we would not expect the various approaches to give very different results.

Naive Approaches

We'll start with demonstrating how to use `R` to run the naive approaches. First, we'll re-scale our variables so that age is measured in 10 yr increments, pulse in beats per second, and systolic blood pressure in 10 mmHg increments.

```
nha$Age10 <- nha$Age/10
nha$PulseSec <- nha$Pulse/60
nha$BPSysAve10 <- nha$BPSysAve/10
```

Listwise Deletion

By default, `lm()` will use listwise deletion and restrict to only those with complete data:

```
listwise_mod <- lm(TotChol ~ Age10 + PulseSec + BPSysAve10, data = nha)
summary(listwise_mod)
```

```
##
## Call:
## lm(formula = TotChol ~ Age10 + PulseSec + BPSysAve10, data = nha)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8992 -0.5863 -0.0927  0.5696  3.5398
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.72763    0.26472  10.304  < 2e-16 ***
## Age10        0.30073    0.03199   9.402  < 2e-16 ***
## PulseSec     0.20972    0.12905   1.625 0.104373
## BPSysAve10   0.07551    0.01951   3.871 0.000113 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8762 on 1424 degrees of freedom
##   (75 observations deleted due to missingness)
## Multiple R-squared:  0.08087,    Adjusted R-squared:  0.07893
## F-statistic: 41.76 on 3 and 1424 DF,  p-value: < 2.2e-16
```

Maximum Likelihood

To implement full information maximum likelihood, we'll use the `lavaan` package. This implementation assumes a multivariate normal distribution for all the variables:

```
ml_mod <- sem('TotChol ~ Age10 + PulseSec + BPSysAve10', data = nha,
              missing = 'fiml', fixed.x = F)
summary(ml_mod)
```

```
## lavaan 0.6.15 ended normally after 34 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                        14
##
##   Number of observations                          1503
##   Number of missing patterns                         3
##
## Model Test User Model:
##
##   Test statistic                                 0.000
##   Degrees of freedom                                 0
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Observed
##   Observed information based on                Hessian
##
## Regressions:
##                     Estimate  Std.Err  z-value  P(>|z|)
##   TotChol ~
##     Age10              0.314    0.031    9.997    0.000
##     PulseSec           0.210    0.129    1.625    0.104
##     BPSysAve10         0.076    0.020    3.879    0.000
##
## Covariances:
##                     Estimate  Std.Err  z-value  P(>|z|)
##   Age10 ~~
##     PulseSec          -0.005    0.004   -1.476    0.140
##     BPSysAve10         0.173    0.024    7.196    0.000
##   PulseSec ~~
##     BPSysAve10         0.022    0.006    3.758    0.000
##
## Intercepts:
##                     Estimate  Std.Err  z-value  P(>|z|)
##    .TotChol            2.680    0.264   10.138    0.000
##     Age10              3.238    0.019  170.487    0.000
##     PulseSec           1.263    0.005  264.281    0.000
##     BPSysAve10        11.095    0.032  344.420    0.000
##
## Variances:
##                     Estimate  Std.Err  z-value  P(>|z|)
##    .TotChol            0.769    0.028   27.393    0.000
##     Age10              0.542    0.020   27.414    0.000
##     PulseSec           0.033    0.001   26.741    0.000
##     BPSysAve10         1.485    0.056   26.720    0.000
```

Multiple Imputation

Now, we'll implement multiple imputation using the `mice` package. We'll first create $m = 10$ imputed datasets. By default, `mice()` will use predictive mean matching (which is what we'll use here), but this could get swapped out for something like `method="norm"`. You could also have different methods for different variables. Specifying only one method will use that for all variables with missingness. Here, we have monotone missingness and so we only need one iteration `maxit = 1`. If we did not, and wanted to implement chained equations, we would change `maxit` and then want to assess convergence (https://stefvanbuuren.name/fimd/sec-algoptions.html)

```
step1 <- mice(nha %>% select(TotChol, Age10, PulseSec, BPSysAve10),
              maxit = 1, m = 10, visitSequence = "monotone",
              method = "pmm", seed = 6, print = F)
```

The `predictorMatrix` tells `mice()` what variables to include in the imputation model. We used the default that `mice()` creates, which is shown here:

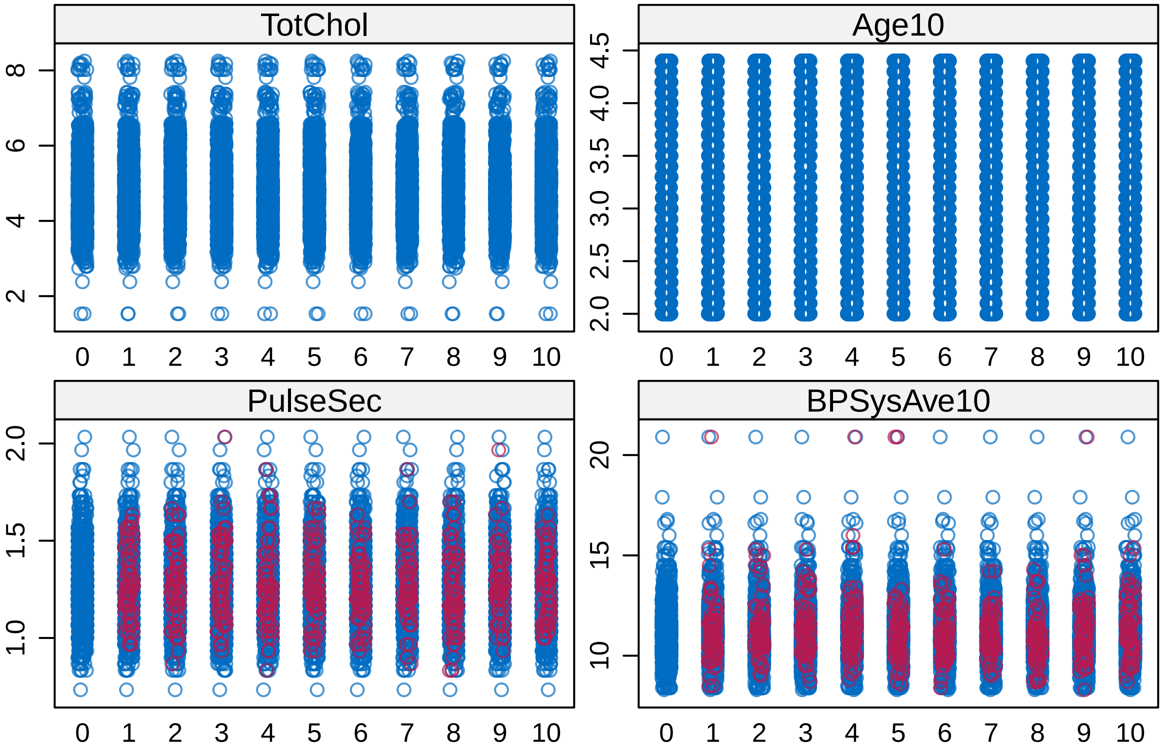
```
step1$predictorMatrix
```

```
##            TotChol Age10 PulseSec BPSysAve10
## TotChol          0     0        0          0
## Age10            1     0        0          0
## PulseSec         1     1        0          0
## BPSysAve10       1     1        1          0
```

This is most easily interpreted by looking at each row one by one. Each row corresponds to a variable to impute and the `1`s indicate which variables are going to be used in the imputation model. We can ignore `TotChol` and `Age10` since there is no missingness and see that `Pulse10` is including `TotChol` and `Age10` in the imputation model. We could adjust this predictor matrix if we wanted. It can be handy to run `ini <- mice(data, maxit = 0)` so that it'll set up some defaults (`ini$meth`, `ini$predictorMatrix`), adjust them, and then use those adjusted versions as input into the desired run of `mice()`.

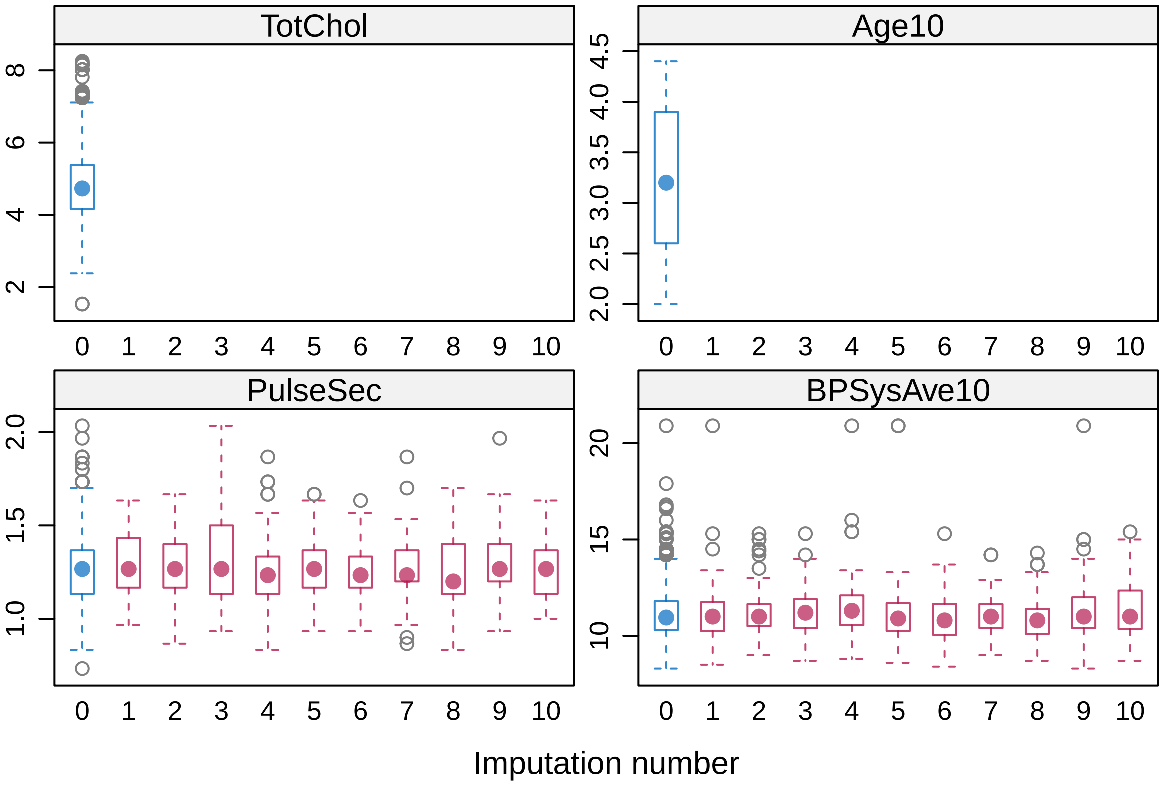
Here's some code to visualize outputs:

```
stripplot(step1)
```

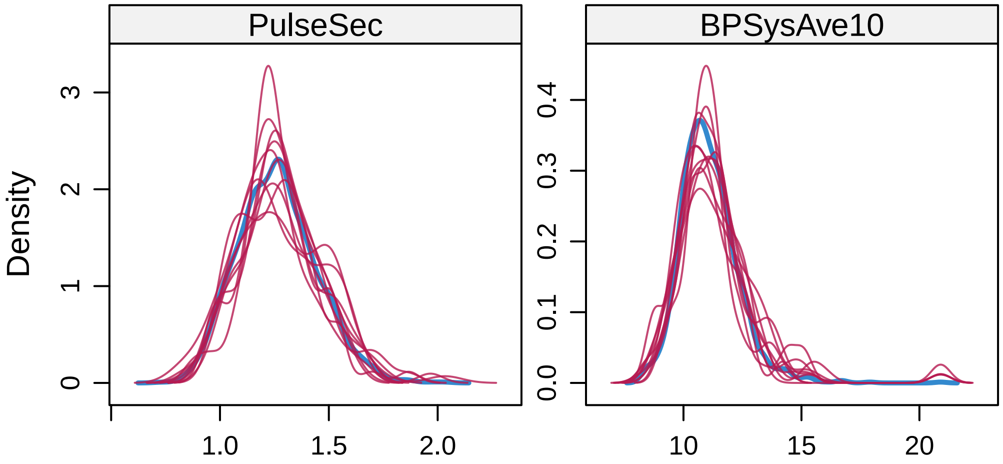


The x-axis corresponds to an imputed dataset. 0 is the original dataset (with missing values). The red circles indicate the imputed values. This is a useful plot when the number of observations is fairly small.

```
bwplot(step1)
```



```r
densityplot(step1)
```



These density plots are useful for when there are more observations. We can generally see that the distribution of the imputed values (there are 10 red lines where each displays the distribution of the imputed values from the $m = 10$ imputed datasets) is fairly similar to the observed values (blue line). Keep in mind with predictive mean matching, it is not possible to impute a value that we did not observe in our dataset.

Moving on to the next stage, for each of our $m = 10$ datasets, we'll fit our model:

```r
step2 <- with(step1, lm(TotChol ~ Age10 + PulseSec + BPSysAve10))
```

and then pool:

```r
mi_mod <- pool(step2)
summary(mi_mod)
```

```
##          term   estimate  std.error statistic        df      p.value
## 1 (Intercept) 2.72841506 0.27056066 10.084301  566.1499 4.180071e-22
## 2       Age10 0.31470154 0.03150073  9.990295 1483.3079 8.661320e-23
## 3    PulseSec 0.19691174 0.13155178  1.496838  707.2267 1.348815e-01
## 4  BPSysAve10 0.07280496 0.01960694  3.713223  722.4427 2.203976e-04
```

Inverse Probability Weighting

Next, we'll implement inverse probability weighting. We begin by creating a missingness model using complete variables only (total cholesterol and age). Because the missingness is monotonic and only two participants have data on pulse but not blood pressure, this does not reduce the information available for the missingness model substantially. We can do this by creating an indicator that a case is a complete case and then fitting a logistic regression model.

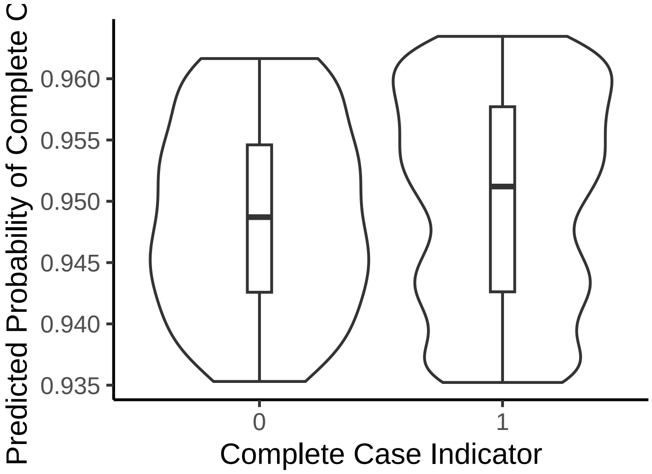
```
nha$C <- as.numeric(!is.na(nha$BPSysAve10))
miss_mod <- glm(C ~ Age10 + TotChol, data = nha, family = binomial(link="logit"))
```

We will need the predicted probability that each observed case is a complete case. To obtain predicted probabilities from the logistic regression model, we can use the `predict()` function with `type="response"`. Comparing the predicted probabilities between cases that did and did not end up actually being complete, we find that the overall range of predicted probabilities is very small (0.935 to 0.965 or so) and, although the median probability is a little higher among complete cases than among cases with missing data, there is a huge amount of overlap between the two distributions (as shown by the violin plots).

```
nha$pred_probs <- predict(miss_mod, type="response")
summary(nha$pred_probs)
```

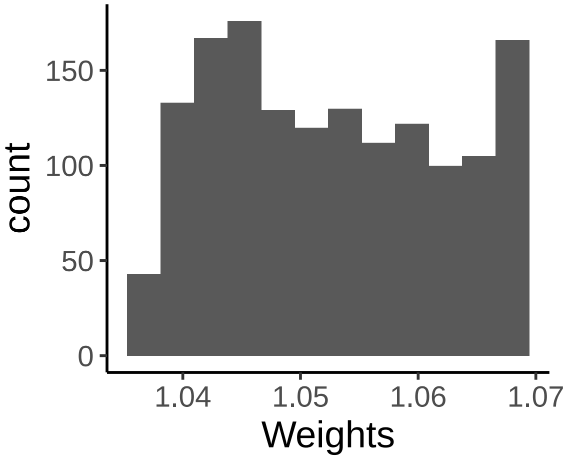
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.9352  0.9426  0.9512  0.9501  0.9577  0.9635
```

```
ggplot(data=nha) +
  geom_violin(aes(x=factor(C), y=pred_probs)) +
  geom_boxplot(aes(x=factor(C), y=pred_probs), width=0.1) +
  xlab("Complete Case Indicator") +
  ylab("Predicted Probability of Complete Case") +
  theme_classic()
```



We then take the inverse of these probabilities to obtain the weights and fit a weighted regression model. The coefficient estimates are appropriately weighted, but robust or bootstrap standard errors should be used instead of the standard errors reported from `lm()`. The robust (sandwich) standard errors will typically be valid, but inefficient; bootstrap standard errors are likely to perform better overall. We also examine the weights: very large or very small weights may indicate a misspecified missingness model, and it may be useful to truncate them – for example, at the 1st and 99th percentile, or at the 5th and 95th percentile, or only on the high end at 10 or 20 (all common rules of thumb).

```
# Obtain weights and check distribution
nha$weights <- 1/nha$pred_probs
ggplot(nha) +
  geom_histogram(aes(x=weights), bins=12) +
  xlab("Weights") +
  theme_classic() +
  theme(text=element_text(size=14))
```



```
# Fit model
ipw_mod <- lm(TotChol ~ Age10 + PulseSec + BPSysAve10,
              weights = weights, data = nha)
model.res <- coef(summary(ipw_mod))

# Obtain robust standard errors (lmtest package with SE functions from sandwich)
robust.res <- coeftest(ipw_mod, vcov = vcovHC(ipw_mod, type = "HC3"))

# Obtain bootstrap standard errors
boot_ipw_ests <- matrix(nrow=1000, ncol=4)
for (i in 1:1000) {
  set.seed(i)
  nha.star <- sample_n(nha, size=nrow(nha), replace = T)
  missmod.star <- glm(C ~ Age10 + TotChol, data = nha.star,
                      family = binomial(link="logit"))
  obsprobs.star <- predict(missmod.star, type="response")
  ipwmod.star <- lm(TotChol ~ Age10 + PulseSec + BPSysAve10,
              weights = 1/obsprobs.star, data = nha.star)
  boot_ipw_ests[i,] <- coef(ipwmod.star)
}
boot.se <- apply(boot_ipw_ests, 2, sd)

# Summarize all results
data.frame(estimates = model.res[, "Estimate"],
           model.se = model.res[, "Std. Error"],
           robust.se = robust.res[, "Std. Error"],
           boot.se = boot.se)
```

```
##              estimates   model.se  robust.se    boot.se
## (Intercept) 2.72947196 0.26463302 0.30417135 0.30373921
## Age10       0.30096310 0.03199094 0.03212458 0.03226483
## PulseSec    0.20960535 0.12908648 0.14158028 0.13649857
## BPSysAve10  0.07529751 0.01949033 0.02399534 0.02315915
```

Doubly Robust Estimation

An R package called `drgee` can perform doubly robust estimation. There are more options for this package than we will use in this simple example, so I highly recommend reading through the corresponding article (https://www.degruyter.com/document/doi/10.1515/em-2014-0021/html?lang=en). There are a few other R packages that perform doubly robust estimation in the causal inference framework (e.g., `drtmle` performs doubly-robust estimation of the average treatment effect using targeted maximum likelihood estimation (TMLE)).

For our use of `drgee`, we'll need to specify a model for the outcome (`oformula`), a model for the exposure (`eformula`), and link functions for both outcome and exposure. Suppose we're primarily interested in the association between blood pressure and cholesterol, and we're treating age and pulse as nuisance covariates:

```
library(drgee)
dr_mod <- drgee(oformula = TotChol ~ Age10 + PulseSec,
             eformula = BPSysAve10 ~ Age10 + PulseSec,
             olink = "identity", elink = "identity",
             estimation.method = "dr",
             data = nha)
summary(dr_mod)
```

```
##
## Call:  drgee(oformula = TotChol ~ Age10 + PulseSec, eformula = BPSysAve10 ~
##     Age10 + PulseSec, olink = "identity", elink = "identity",
##     data = nha, estimation.method = "dr")
##
## Outcome:  TotChol
##
## Exposure:  BPSysAve10
##
## Covariates:  Age10,PulseSec
##
## Main model:  TotChol ~ BPSysAve10
##
## Outcome nuisance model:  TotChol ~ Age10 + PulseSec
##
## Outcome link function:  identity
##
## Exposure nuisance model:  BPSysAve10 ~ Age10 + PulseSec
##
## Exposure link function:  identity
##
##            Estimate Std. Error z value Pr(>|z|)
## BPSysAve10  0.07551    0.02361   3.198  0.00138 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Note: The estimated parameters quantify the conditional
## exposure-outcome association, given the covariates
## included in the nuisance models)
##
##  1428  complete observations used
```

Summary of Results

Finally, let's compare the parameter estimates and standard errors for the association between blood pressure and cholesterol.

Method	Estimate	SE
Listwise deletion	0.0755	0.0195
Maximum likelihood	0.0758	0.0196
Multiple imputation	0.0728	0.0196
IPW with model-based SE	0.0753	0.0195
IPW with robust SE	0.0753	0.0240
IPW with bootstrap SE	0.0753	0.0232
Doubly robust	0.0755	0.0236