Missing Data

Joseph Nathan Cohen

11/10/2019

Contents

Introduction **1**

Missing Data Affects Results **2**

Dealing with Missing Data **2**
Listwise Deletion . . . 2
Mean Imputation . . . 3
Regression Imputation . . . 3
Multiple Imputation with Randomness . . . 6

Practical Walk Through of the Method **7**
Prepare Data . . . 7
Review Summary Statistics . . . 7
Transform Variables? . . . 10
Create Imputed Sets . . . 10
Diagnose Imputations . . . 13
Post-Imputation Transformations . . . 15

Descriptive Statistics **16**
Discrete Univariate . . . 16
Continous Univariate . . . 16
Cross-Tabulations . . . 17
Summary Statistics Table . . . 17
Correlations . . . 18

Regression Analysis **18**
Linear model . . . 18
Logit . . . 19

Introduction

As you know, data sets often include observations with missing data. In general, analysts ignore the impact of missing data. It is a mistake to ignore missing data. This program can influence your findings considerably, and affect the consequences of any decision that is premised on your findings.

Our discussion today proceeds in four parts. I start with a discussion of how missing data affects results. Then, we talk about different strategies for dealing with missing data. I will then describe the process we will learn today: multiple imputation with randomness. The lesson will conclude with a walk through of this method.

Missing Data Affects Results

In your methods class, you should have learned about *response bias*, a form of sample bias that results from particular kinds of people being unwilling to answer particular survey questions (or answer survey questions at all). For example, imagine wealthier people were far more likely to refuse to answer questions about personal finances. Then our estimates would be based on data that underrepresented people with more money. We would assume that the general population more closely resembles those with less wealth than it actually does.

Insofar as wealthier people's refusals to answer *any* questions are concerned, the impact of response bias might be mitigated by weighting (see last week's lessons). For example, if wealthier people tend to be older, married, white, and college-educated, then our estimates could weight observations along these variables and develop estimates that could better reflect people with these characteristics. But what about when people's non-responses are selected – when they answer some questions but not others? Reweighting observations is will not address this problem.

In situations where people answer questions selectively, the effect on our descriptive statistics is more straightforward. For example, if wealthier people were more likely to skip questions about money, then we might be more likely to under-estimate what the average person earns or owns. If you people with money aren't part of your data, then a lack of money looks much more prevalent.

The problem is even more pernicious with regression results. Using conventional analysis methods, *a missing observation on any single variable removes* ***all*** *of a subject's data from our estimates.* Any missing data point causes the regression analysis to throw away all other data associated with an observation, even if their data is complete across all other variables.

These problems can be even worse in non-survey analysis. For example, I found missing data to be a big headache in my research to model country-level political or economic outcomes. Poorer countries are generally missing data. Data becomes more spotty is one reaches further into the past. Many data sets only cover particular countries. In general, it is worth learning about the pitfalls and redresses to missing data problems.

Dealing with Missing Data

There are many strategies for dealing with missing data.

Listwise Deletion

This is the default method for dealing with missing data, and is likely the method employed in any analysis that does not articulate a deliberate strategy for dealing with missing data. This method simply ignores missing data in descriptive statistics, and drops observations with any missing data from our estimates.

Some special considerations when contemplating regression modeling with missing data:

If missing data are *missing completely at random* (MCAR), then listwise deletion is not expected to render biased results due to missing data. To be MCAR, every observation on every variable has an equal chance of being missing. Missingness is not more prevalent among certain types of respondents or particular variable scores. An example of this kind of missingness is that resulting from people accidentially skipping a question in a survey due to a momentary lapse in attention or inadvertent error in filling out answers. This is a problem can happen to anyone at any time. It's not an error that you get from particular types of people or when people have particular answers to your question. Under MCAR, listwise deletion is fine.

Also, listwise deletion is also an effective strategy when our predictors, but not our outcome, is *missing at random* (MAR).[1] MAR is:

> a situation in which an observation's likelihood of missingness can be predicted by other predictors in the model, but not the values of the outcome variable. Allison notes, "In general, data are *not* missing at random if those individuals with missing data on a particular variable tend to

[1] Allison, Paul (2002) *Missing Data* Sage.

> have lower (or higher) values on that variable than those with data present, controlling for other observed variables." It is impossible to test whether MAR ... is satisfied [because] we do not know the values of the missing data, we can not compare the values of those with and without missing data to see if they differ systematically on that variable

If missingness is MCAR or MAR, then the missing data mechanism is said to be "ignorable". In other words, we can ignore missing data concerns and just drop observations when we are generating regression models. If missing data is *non-ignorable*, then we need some stratgy to mitigate the biasing effects of missingness.

Mean Imputation

This method is one in which we fill-in missing values with the sample means. Imagine we had a variable measuring people's ratings for the movie *Star Wars*. If the sample mean score of *Star Wars* ratings with 3.5 out of 5, then we would plug 3.5 into every missing data cell.

Through this method, our regression will no longer drop observations that were previously missing data. The method is illustrated in Figure 1 (below) from Iris Eekhout.[2]

This strategy will likely distort correlations, will reduce the metric's sample variability, and bias estimates.

Regression Imputation

Another method is to use impute missing values by using regressions to predict their values. Imagine the relationship:

$$y = \alpha + \beta_1 \times x_1 + \beta_2 \times x_2 + \epsilon$$

Where observations are missing values on x_2. We could predict x_2 by:

$$x_2 = \omega + \gamma_1 \times y + \gamma_2 \times x_2 + \zeta$$

Where ω is an intercept, γ terms are coefficients, and ζ is an error term.

So we use regression to predict the missing values, then plug them into the data in lieu of missing values, and reanalyze.

The method can be illustrated as by Eekhout in Figure 2 below.

The primary issue with this method is that it tends to reinforce observed relationships by imputing values that strongly conform to observed relationships. In effect, they bake regression results into their imputations, and these imputations reinforce the model from which they were constructed. Such a strategy has the potential to artificially exaggregate coefficients' significance estimates.

[2]Eekhout, Iris (n.d.) "Single Imputation Methods" Web Page https://www.iriseekhout.com/missing-data/missing-data-methods/imputation-methods/

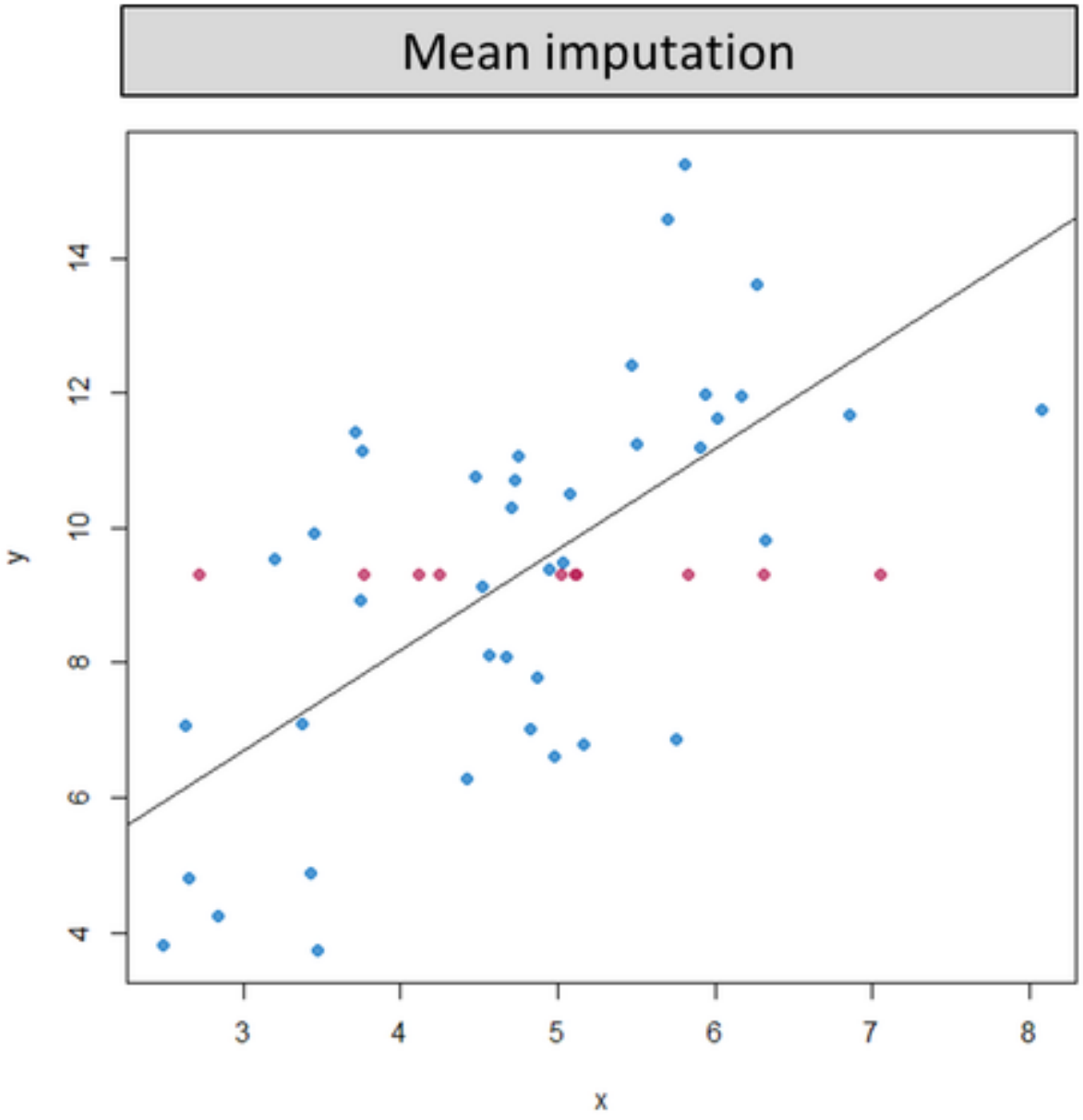


Figure 1: Mean Imputation Illustrated

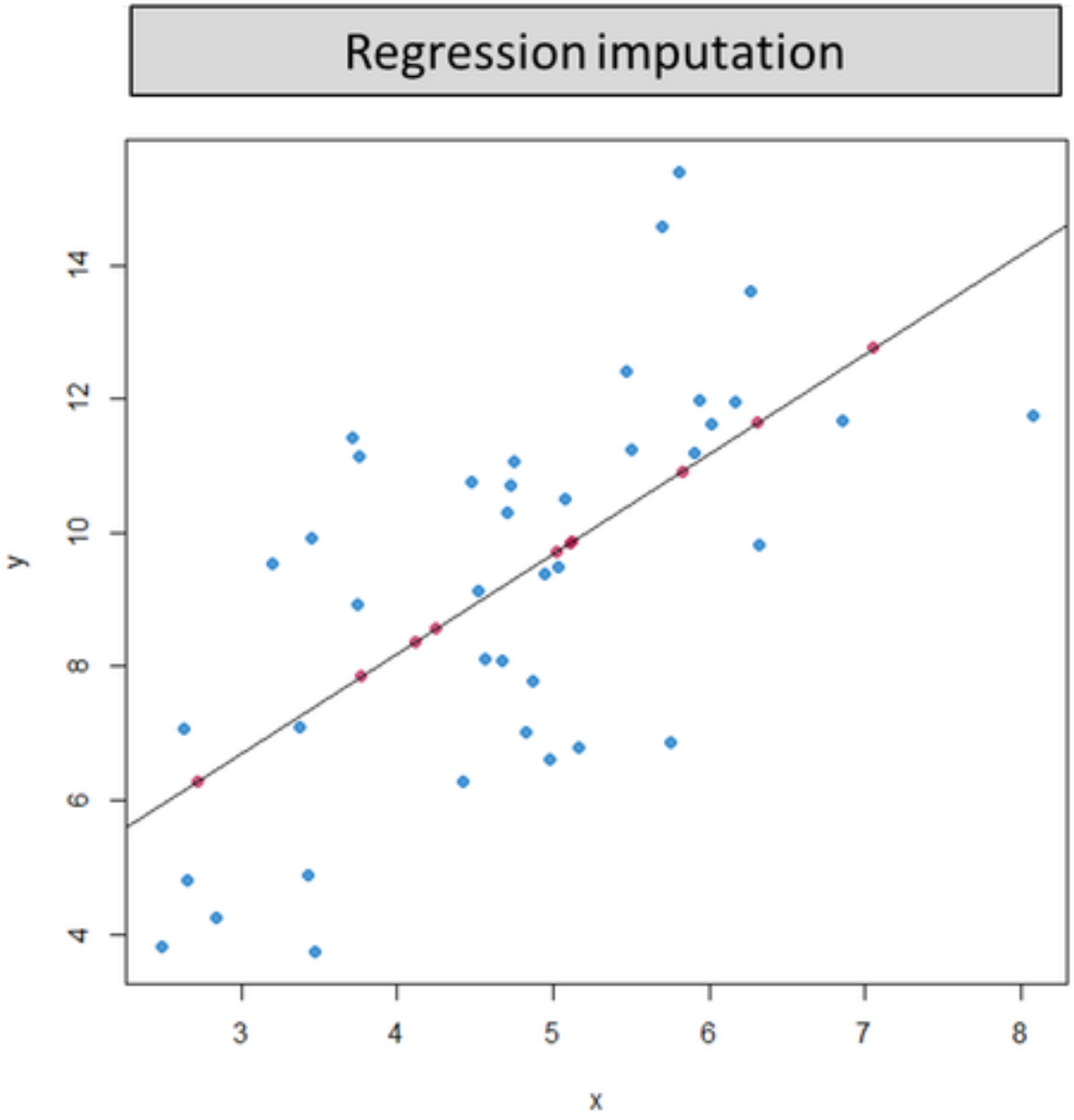


Figure 2: Regression Imputation Illustrated

Multiple Imputation with Randomness

Multiple imputation with randomness can be understood as roughly similar to regression imputation, but one that injects randomness into estimates in order to counteract regression imputation's penchant to overestimate the strength of observed relationships. We do this by generating multiple imputated data sets with randomness injected into missing value estimates, and then we see which coefficients tend to be strong predictors across all our data sets.

This is how the process works:

First, we begin by generating between 5 and 20 imputed data sets. For each data set, the algorithm will use regression imputation to generate missing value estimates. However, each estimate will have some random modification to each imputatied value, thereby injecting variability into these estimates. Once it creates these imputations, the algorithm reestimates missing values, reinjects another round of random variation into the estimates, and so on. The process will continue until the model is said to have "converged", which means that it keeps on getting the same imputation estimates over and over gain. Repeat the process for as many imputed sets you want to generate. Studies suggest that the benefit of generating more sets falls quickly by the fifth set.

Then, we take our separate imputed sets and run the same regression on them. Then, we combine the results of the same regression that has been applied to all the imputed sets we generated. The coefficient of our imputed data analysis is just the mean value of the coefficients yielded each of our imputed sets:

$$\bar{r} = \frac{\sum_{i=0}^{M} \beta_i}{M}$$

$$SE = \sqrt{\frac{1}{M}\sum_{k} s_k^2 + (1 + \frac{1}{M}) \times (\frac{1}{M-1}) \times \sum (r_k - \bar{r})^2}$$

Where

- M is the number of imputation models
- r_k is the coefficient esimate of model k
- $\bar{r}$ is the coefficient estimate of all imputation models
- s is the standard error estimate of
- SE is our overall standard error estimates

To calculate significance estimates, divide the overall coefficient estimate by the overall standard error estimate. The result is a t-test score. Look up critical values on a t-test critical values table (e.g., https://www.itl.nist.gov/div898/handbook/eda/section3/eda3672.htm). With a large sample, a t value above 1.96 is good for $p<0.05$ significance. Above 2.58 for $p<0.01$. Above 3.29 for $p<0.001$.

Simulation studies suggest that this method compensates for the aggressive significance estimates of regression imputation, and still allows you the benefit of retaining data that otherwise would have been discarded due to missing data.

To learn more about this method, I recommend a small, concise introduction given by Paul Allison (2002) *Missing Data* Sage Publications.

Practical Walk Through of the Method

Prepare Data

Your goal is to have a clean data set for imputation. This data set should include any variables that you plan to model, plus any additional data that you expect to be correlated with missing values.

Today, we will use data on people's personal finances, demographics, and political views from the *International Social Survey Program*. This is a multinational survey that is often used for studies that probe the relationship between these money, politics, and demography across countries. For today's discussion, we will use data from the Australian, French, British, and American surveys from this program.

This week, I will hide the code used to clean these data in the Markdown file. From the original file, I created four clean data objects: ISSP-IT.RDS (for Italy), ISSP-FR.RDS (France), ISSP-KR.RDS (Korea), and ISSP-US.RDS (USA).

Each of these four sets have the same variables, which are described in the accompanying text file called "ISSP Codebook.txt".

Let's try looking at the United States survey data.

```
#Load USA data set
dat <- readRDS("ISSP.US.RDS")

#Variables  in the set
names(dat)
```

```
##  [1] "respid"          "ahead.famwealth" "ahead.fameduc"
##  [4] "ahead.perseduc"  "ahead.ambition"  "ahead.hardwork"
##  [7] "ahead.knowppl"   "ahead.polcon"    "ahead.bribes"
## [10] "ahead.race"      "ahead.religion"  "ahead.sex"
## [13] "sex"             "age"             "marital"
## [16] "degree"          "wrkst"           "party_lr"
## [19] "weight"          "hinc"
```

Your set should only include variables that you intend to use in your analysis, or those that you ahve included in order to help better impute missing variables. As you add more extraneous variables, the imputation process becomes more computationally demanding and more likely to fail.

In this example, we will keep all of our variables.

Review Summary Statistics

Once the data is cleaned and ready, review your summary statistics. You are looking for:

- Completely missing variables (which should be dropped from your imputation)
- Severely missing variable (which may pose problems)
- Which variables are nominal (i.e., unordered discrete), ordinal, or
- Which variables are identifiers that should not be included in the calculations

Our variables' summary statistics, after cleaning:

```
summary(dat)
```

```
##      respid               ahead.famwealth            ahead.fameduc
##  Min.   :8001   Essential         : 79    Essential         :112
##  1st Qu.:8396   Very Important    :387    Very Important    :667
##  Median :8791   Fairly Important  :492    Fairly Important  :563
##  Mean   :8791   Not Very Important:388    Not Very Important:170
##  3rd Qu.:9186   Not Important     :183    Not Important     : 49
##  Max.   :9581   NA's              : 52    NA's              : 20
##
##             ahead.perseduc            ahead.ambition
##  Essential         :477    Essential         :627
##  Very Important    :903    Very Important    :798
##  Fairly Important  :168    Fairly Important  :113
##  Not Very Important: 14    Not Very Important: 19
##  Not Important     :  6    Not Important     :  3
##  NA's              : 13    NA's              : 21
##
##             ahead.hardwork            ahead.knowppl
##  Essential         :686    Essential         :156
##  Very Important    :818    Very Important    :572
##  Fairly Important  : 61    Fairly Important  :647
##  Not Very Important:  7    Not Very Important:170
##  Not Important     :  3    Not Important     : 23
##  NA's              :  6    NA's              : 13
##
##              ahead.polcon             ahead.bribes
##  Essential         : 61   Essential         :   8
##  Very Important    :246   Very Important    :  33
##  Fairly Important  :458   Fairly Important  :  79
##  Not Very Important:565   Not Very Important: 313
##  Not Important     :187   Not Important     :1006
##  NA's              : 64   NA's              : 142
##
##               ahead.race             ahead.religion
##  Essential         : 21   Essential         : 49
##  Very Important    :149   Very Important    :139
##  Fairly Important  :291   Fairly Important  :165
##  Not Very Important:492   Not Very Important:497
##  Not Important     :569   Not Important     :687
##  NA's              : 59   NA's              : 44
##
##               ahead.sex       sex            age
##  Essential         : 22   Male  :   0   Min.   : 6.00
##  Very Important    :142   Female:   0   1st Qu.:22.00
##  Fairly Important  :253   NA's  :1581   Median :35.00
##  Not Very Important:548                 Mean   :36.13
##  Not Important     :545                 3rd Qu.:48.00
##  NA's              : 71                 Max.   :85.00
##                                         NA's   :11
##                  marital                        degree
##  Married/Cohabiting  :775   No Formal            :  0
##  Widowed             :125   Lowest Formal        : 20
```

```
##  Divorced            :237   Above Lowest Formal   : 48
##  Separated           : 45   Higher Secondary      :135
##  Single Never Married:397   Above Higher Secondary:508
##  NA's                :  2   University            :417
##                             NA's                  :453
##                      wrkst          party_lr       weight
##  Employed, Part-Time   :740   Far Left :  0   Min.   :0.3219
##  Homemaker             :278   Left     :  0   1st Qu.:0.5290
##  Disabled              :187   Center   :559   Median :0.8688
##  Employed, Less than PT:180   Right    :614   Mean   :1.0000
##  Studying              : 85   Far Right:368   3rd Qu.:1.1121
##  (Other)               : 50   NA's     : 40   Max.   :6.7555
##  NA's                  : 61
##       hinc
##  10     :  50
##  9      :  47
##  8      :  26
##  1      :  19
##  7      :  17
##  (Other):  50
##  NA's   :1372
```

Note High Missing Variables

If we examine the summary statistics, we see that the **sex** variable is completely missing, and that there is severe missingness on the variables **degree** and **hinc**. This suggests that we have to drop **sex** from our analysis, and that we may not be able to use **degree** and **hinc** if the operation has trouble making these imputations.

```
#Remove sex from data set:
dat <- dat[-13]
names(dat)
```

```
##  [1] "respid"          "ahead.famwealth" "ahead.fameduc"
##  [4] "ahead.perseduc"  "ahead.ambition"  "ahead.hardwork"
##  [7] "ahead.knowppl"   "ahead.polcon"    "ahead.bribes"
## [10] "ahead.race"      "ahead.religion"  "ahead.sex"
## [13] "age"             "marital"         "degree"
## [16] "wrkst"           "party_lr"        "weight"
## [19] "hinc"
```

Identify Variable Types

Specify the variables types by storing their variable names as objects:

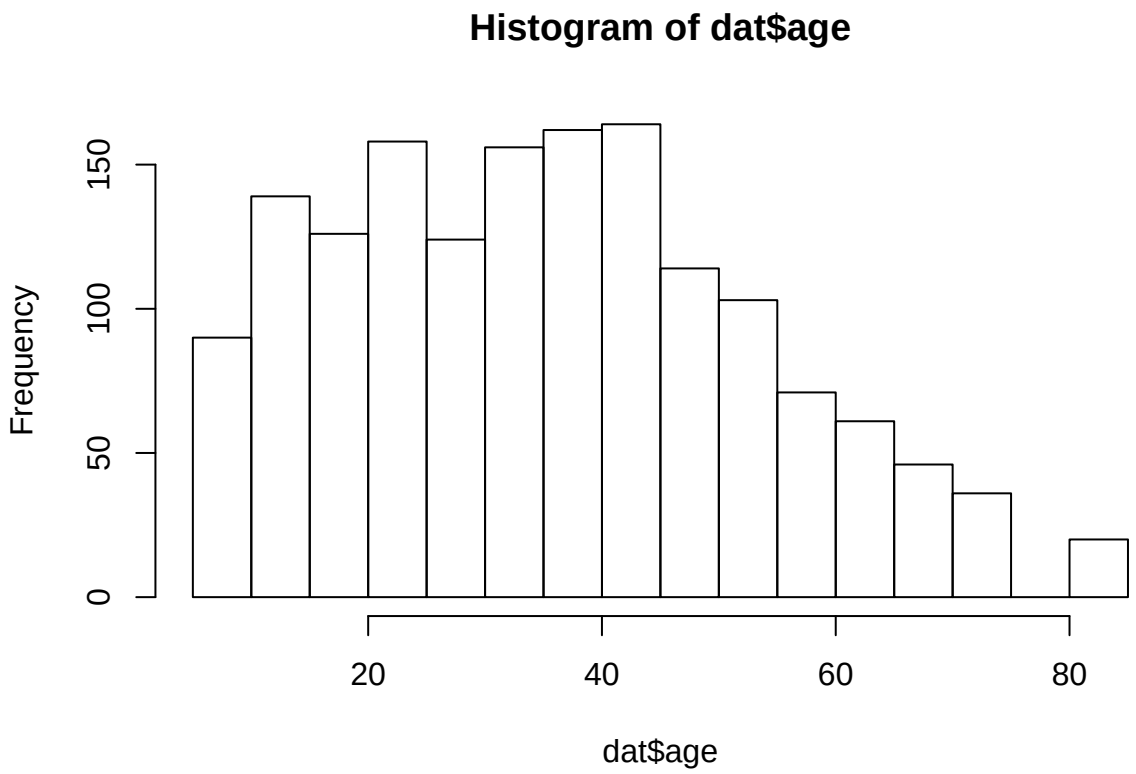
```
id.vars <- c("respid", "weight")
nom.vars <- names(dat)[c(14, 16)]
ord.vars <- names(dat)[c(2:12, 15,17, 19)]
```

Transform Variables?

You can aid the imputation process by transforming any variables with strong skews or outliers. So once your data set is loanded and clean, look to see if you have to correct for any crazy skews or outliers.

This particular set is mainly comprised of discrete variables. We have one continuous variable: **age**. However, it seems to have a reasonably centered distribution without crazy outliers, so there might not be so much to gain by transforming the variable.

```
hist(dat$age)
```



However, if there was a skew, you might transform the variable by logging it, top- or bottom-coding it, or cutting it, or using some other strategy to help render a more centered and tight distribution.

Create Imputed Sets

We are going to use a package called ***Amelia***, created by Gary King and colleagues at Harvard. For more on Amelia, visit: https://gking.harvard.edu/amelia

There are a lot of options to contemplate when imputing data, and you have to set these options thoughtfully. Refer to the Amelia R documentation for more details: https://cran.r-project.org/web/packages/Amelia/Amelia.pdf

In this case, we will generate imputations using the following command. The output should give you a running update of the above-described chain of estimation and re-estimation described above. These iterations will stop when the model has "converged" (it keeps on getting the same estimates, despite the continuous infusion of randomness into estimates). Under good circumstances, convergence comes quickly. Be careful if it takes several hundred iterations or does not converge at all.:

```r
library(Amelia)
set.seed(2019)   # Set a random seed to ensure replicability
dat.imp <- amelia(dat,     #Data set to be used
                  m = 5,   #Number of imputed sets
                  idvars = id.vars, #List of identifier variables (see above)
                  ords = ord.vars,  #Ordinal variables
                  noms = nom.vars,  #Nominal variables
                  empri = 0,   #Ridge prior (see below)
                  emburn = c(50,1000) #Minimum & Max number of iterations
                  )
```

```
## -- Imputation 1 --
##
##   1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
##  41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
##  61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
##  81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
##  101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
##  121 122 123 124 125 126 127 128 129 130 131 132 133
##
## -- Imputation 2 --
##
##   1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
##  41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
##  61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
##  81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
##  101 102 103 104
##
## -- Imputation 3 --
##
##   1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
##  41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
##  61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
##  81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
##  101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
##  121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139
##
## -- Imputation 4 --
##
##   1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
##  41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
##  61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
##  81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
##  101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
##  121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
##  141 142 143 144 145 146 147 148 149 150 151 152
##
## -- Imputation 5 --
##
##   1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
##  21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
##  41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
##  61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
##  81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
##  101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
##  121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
##  141
```

The *Ridge Prior* shrinks the covariance in data, and can be used when your model fails (see below). Should be kept as small as possible. King *et al.* recommend less that 1% to 0.5% of number of rows in data.

The "emburn=" specifies a minimum number of iterations ("burn-in") and a maxmimum. The maximum is to prevent Amelia from waiting too long for convergence to happen. (If it's not converging, there's probably a problem). The burn-in is used to make sure that the imputation model makes some minimum effort. I usually don't use a minimum, but will here as an exposition of the option.

The operation will create an object with the results of the imputation:

```
names(dat.imp)
```

```
##  [1] "imputations" "m"           "missMatrix"  "overvalues"  "theta"
##  [6] "mu"          "covMatrices" "code"        "message"     "iterHist"
## [11] "arguments"   "orig.vars"
```

The sub-object "imputations* contains each of the data sets created by this operation:

```
names(dat.imp$imputations)
```

```
## [1] "imp1" "imp2" "imp3" "imp4" "imp5"
```

```
#Take a peek at the imputed set:
summary(dat.imp$imputations$imp1[1:5])
```

```
##      respid               ahead.famwealth            ahead.fameduc
##  Min.   :8001   Essential         : 86    Essential         :117
##  1st Qu.:8396   Very Important    :400    Very Important    :673
##  Median :8791   Fairly Important  :502    Fairly Important  :570
##  Mean   :8791   Not Very Important:397    Not Very Important:171
##  3rd Qu.:9186   Not Important     :196    Not Important     : 50
##  Max.   :9581
##             ahead.perseduc            ahead.ambition
##  Essential         :485    Essential         :637
##  Very Important    :905    Very Important    :805
##  Fairly Important  :169    Fairly Important  :115
##  Not Very Important: 15    Not Very Important: 21
##  Not Important     :  7    Not Important     :  3
##
```

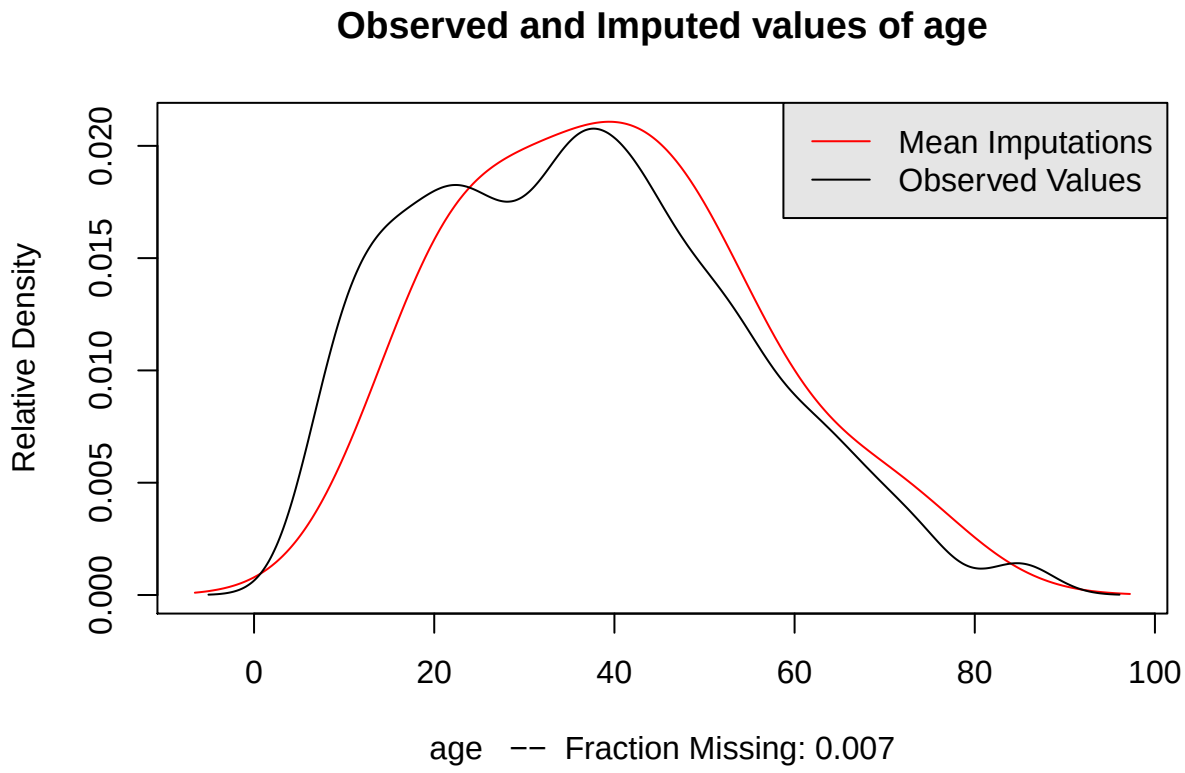
Diagnose Imputations

The next step is a long, drawn-out process that cannot be skipped: diagnosing your imputations.

Comparing Observed & Imputed Variables' Densities

This diagnostic compares the distribution of the reported versus imputed data. We use the **compare.density()** command. This works for continuous variables.

```
compare.density(dat.imp, var = "age")
```

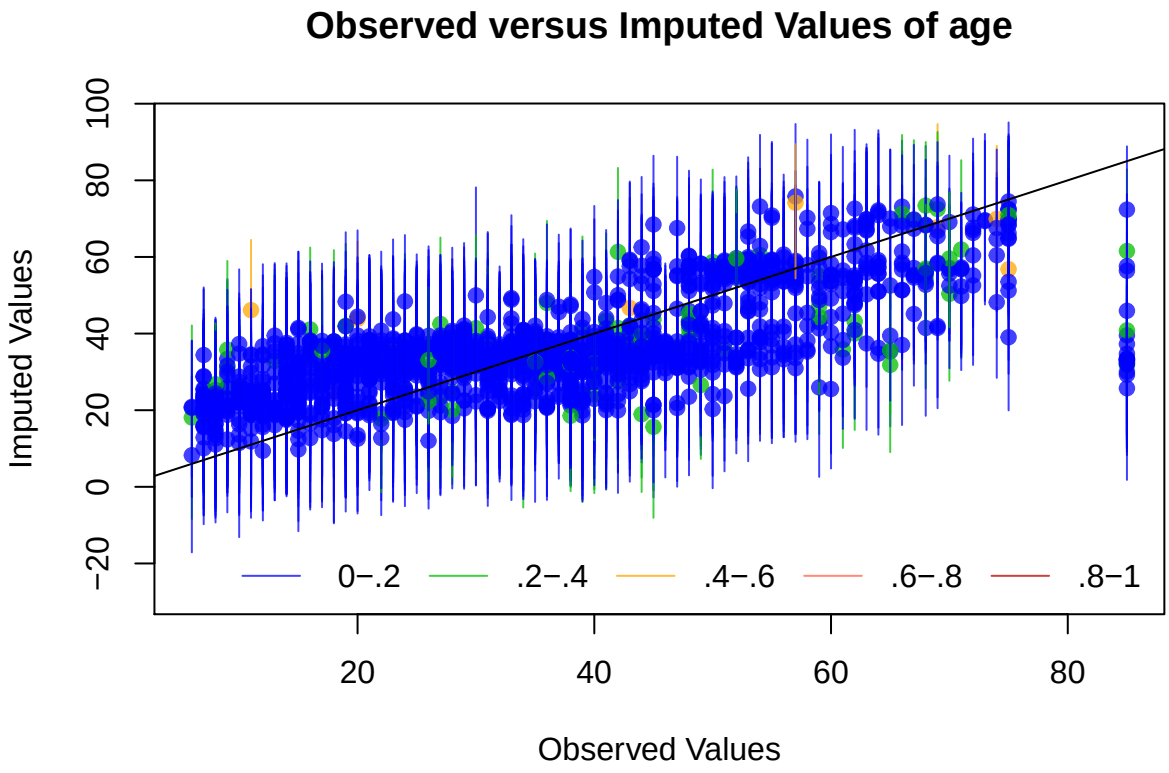


This diagnostic suggests that the model is imputing values that are similar to the distribution of observed values. This is a positive sign.

Overimputation

Overimputation is a diagnostic in which we remove observed variables from our imputation, and see if the model does a good job of predicting their values. Use use the **overimpute()** command. For continuous variables, the result is a scatterplot of observed and imputed values, with lines representing 90% confidence interval. The color of the dots represent missingness at these values.

```
overimpute(dat.imp, var = "age")
```

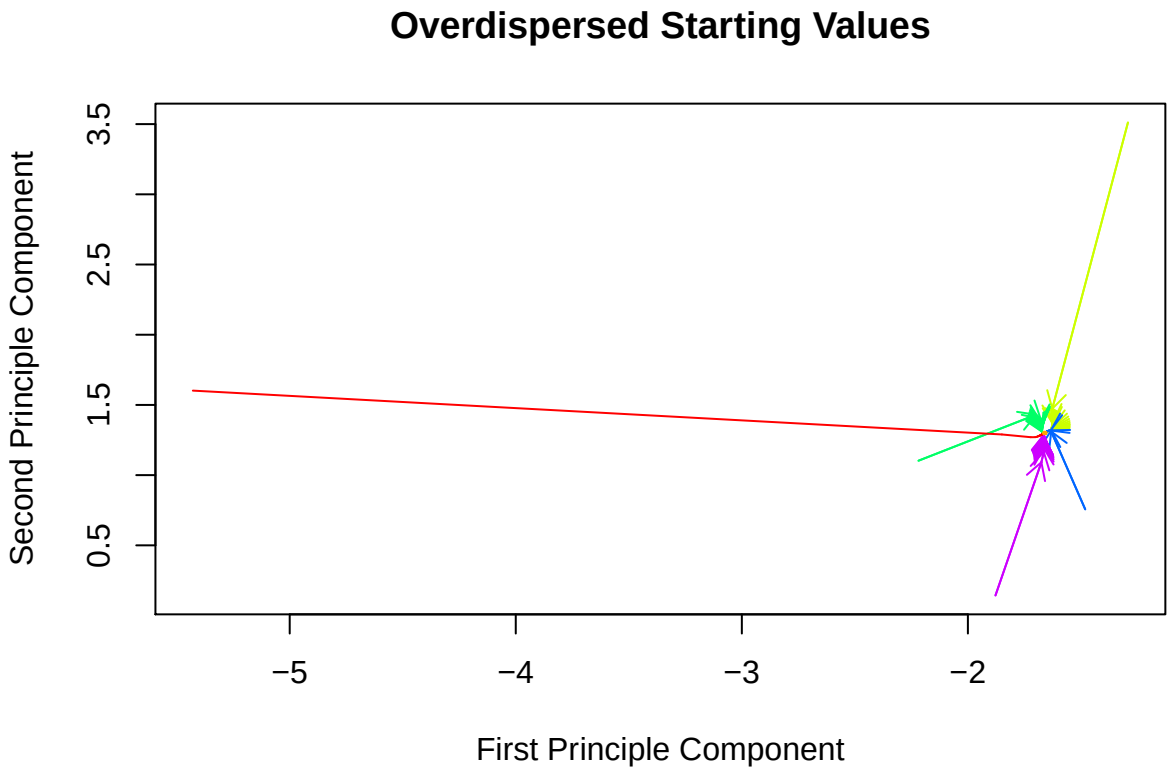


Here, there is some positive relationship, although it is not excellent. Given the low level of missingness and the generally positive correlations that do seem to keep within the confidence interval of true values, I would accept.

Convergence from Overdispersed Starting Values

This diagnostic asks if the imputation model would converge to similar estimates if we began this chain of estimation and reestimation from radically different initial values. The "m=" option is the number of imputed data sets you are diagnosing. You want the paths to end up in the same destination.

```
disperse(dat.imp, dims=2, m=5)
```



Looks good here.

Post-Imputation Transformations

After performing the imputation, you may wish to transform variables. For this, use **transform()**. It works like update() when we studied survye analysis. For example, imainge that I wanted to create a binary variable that was equal to one if a R thinks family wealth is either "essential" or "very important" in getting ahead:

```
dat.imp <- transform(dat.imp,
                     import.famwealth = ifelse(ahead.famwealth %in% c("Essential", "Very Important"), 1
```

Descriptive Statistics

One strategy is to consolidate all of the imputed data sets into one huge set, and then run your descriptives and visualizations from that combined set. Here, I'm using the **rbind()** command, which stacks data frames vertically.

Note that this data is survey data with weights.

```
all.imp.dat <- rbind(dat.imp$imputations$imp1, dat.imp$imputations$imp2,
                     dat.imp$imputations$imp3, dat.imp$imputations$imp4,
                     dat.imp$imputations$imp5)
```

Discrete Univariate

We are going to use **wpct()** to get weighted percents without going through the whole process of using survey designs. Here, we aren't working with complex sampling schemes encoded in the data, so we can get away with it.

```
library(weights)
wpct(all.imp.dat$ahead.race, weight=all.imp.dat$weight, na.rm=T)
```

```
##          Essential     Very Important   Fairly Important
##         0.01348415         0.09312223         0.18018576
## Not Very Important      Not Important
##         0.31914570         0.39406216
```

Continous Univariate

```
mean(all.imp.dat$age, weight = imp.dat$weight, na.rm= T)
```

```
## [1] 36.14825
```

```
median(all.imp.dat$age, weight = imp.dat$weight, na.rm= T)
```

```
## [1] 35
```

```
quantile(all.imp.dat$age, prob = 0.25,  weight = imp.dat$weight, na.rm= T)
```

```
## 25%
##  22
```

Cross-Tabulations

Something new: Check out **crosstab()** from the ***descr*** package:

```
#Using descr library
library(descr)
crosstab(all.imp.dat$ahead.famwealth, all.imp.dat$degree, weight = all.imp.dat$weight,
         prop.c = T, prop.r = T, plot = F)
```

```
##    Cell Contents
## |-------------------------|
## |                   Count |
## |             Row Percent |
## |          Column Percent |
## |-------------------------|
##
## ============================================================================
##             all.imp.dat$degree
## all.m.$.    Lwst Frm   Abv Lw F   Hghr Scn   Abv Hg S   Univrsty   Total
## ----------------------------------------------------------------------------
## Essentil         19         39         45        236         98     437
##                 4.3%       8.9%      10.3%      54.0%      22.4%    5.5%
##                14.1%       9.8%       4.9%       7.7%       2.9%
## ----------------------------------------------------------------------------
## Vry Impr         43        125        264        722        756    1910
##                 2.3%       6.5%      13.8%      37.8%      39.6%   24.2%
##                31.9%      31.2%      28.6%      23.7%      22.2%
## ----------------------------------------------------------------------------
## Frly Imp         37        119        285        890       1123    2454
##                 1.5%       4.8%      11.6%      36.3%      45.8%   31.0%
##                27.4%      29.8%      30.8%      29.2%      33.0%
## ----------------------------------------------------------------------------
## Nt Vry I         27         85        196        819        999    2126
##                 1.3%       4.0%       9.2%      38.5%      47.0%   26.9%
##                20.0%      21.2%      21.2%      26.9%      29.4%
## ----------------------------------------------------------------------------
## Nt Imprt          9         32        134        379        423     977
##                 0.9%       3.3%      13.7%      38.8%      43.3%   12.4%
##                 6.7%       8.0%      14.5%      12.4%      12.4%
## ----------------------------------------------------------------------------
## Total           135        400        924       3046       3399    7904
##                 1.7%       5.1%      11.7%      38.5%      43.0%
## ============================================================================
```

Summary Statistics Table

```
aggregate(age ~ ahead.polcon, data = all.imp.dat, weight = weight, na.rm=T, mean)
```

```
##         ahead.polcon      age
## 1          Essential 32.33017
## 2     Very Important 39.27843
## 3   Fairly Important 36.85417
## 4 Not Very Important 33.84557
## 5      Not Important 38.36777
```

Correlations

For correlations (note that I am treating the income variable as continuous in this example):

```
#Yes, I'm not using it properly.
wtd.cor(all.imp.dat$age, as.numeric(as.character(all.imp.dat$hinc)))
```

```
##   correlation    std.err  t.value      p.value
## Y  0.06675768 0.01122365 5.947947 2.830833e-09
```

Regression Analysis

For regression analysis, we are going to use the package ***Zelig***, which runs regressions across multiple sets and combines them as described above. Zelig can be used to run many types of models, see: http://docs.zeligproject.org/articles/

Note: With these commands, we use the original imputation object.

Linear model

"Age" is the only truly continuous variable in this set:

```
library(Zelig)
model.1 <- zelig(age ~ marital + degree, model = "normal",
                 weight = weight, data=dat.imp)
```

```
## How to cite this model in Zelig:
##   R Core Team. 2008.
##   normal: Normal Regression for Continuous Dependent Variables
##   in Christine Choirat, Christopher Gandrud, James Honaker, Kosuke Imai, Gary King, and Olivia Lau,
##   "Zelig: Everyone's Statistical Software," http://zeligproject.org/
```

```
summary(model.1)
```

```
## Model: Combined Imputations
##
##                               Estimate Std.Error z value Pr(>|z|)
## (Intercept)                     44.814     3.092   14.49   <2e-16
## maritalWidowed                  18.480     1.851    9.99   <2e-16
## maritalDivorced                  1.211     1.185    1.02   0.3066
## maritalSeparated                -7.739     2.497   -3.10   0.0019
## maritalSingle Never Married    -18.612     0.885  -21.03   <2e-16
## degreeAbove Lowest Formal       -3.735     3.813   -0.98   0.3273
## degreeHigher Secondary          -5.716     3.624   -1.58   0.1148
## degreeAbove Higher Secondary    -5.012     3.081   -1.63   0.1037
## degreeUniversity                -8.684     3.192   -2.72   0.0065
##
## For results from individual imputed datasets, use summary(x, subset = i:j)
## Next step: Use 'setx' method
```

Logit

If we wanted to model the binary outcome "important.famwealth" (see above):

```
model.2 <-  zelig(import.famwealth ~ age + marital + degree, model = "logit",
                  weight = weight, data=dat.imp)
```

```
## How to cite this model in Zelig:
##   R Core Team. 2007.
##   logit: Logistic Regression for Dichotomous Dependent Variables
##   in Christine Choirat, Christopher Gandrud, James Honaker, Kosuke Imai, Gary King, and Olivia Lau,
##   "Zelig: Everyone's Statistical Software," http://zeligproject.org/
```

```
summary(model.2)
```

```
## Model: Combined Imputations
##
##                               Estimate Std.Error z value Pr(>|z|)
## (Intercept)                  -0.67428   0.51324   -1.31  0.18892
## age                           0.00842   0.00405    2.08  0.03731
## maritalWidowed                0.00829   0.28311    0.03  0.97663
## maritalDivorced               0.51402   0.17755    2.90  0.00379
## maritalSeparated              1.09832   0.34892    3.15  0.00165
## maritalSingle Never Married   0.55377   0.15854    3.49  0.00048
## degreeAbove Lowest Formal    -0.24886   0.62884   -0.40  0.69229
## degreeHigher Secondary       -0.53642   0.59162   -0.91  0.36457
## degreeAbove Higher Secondary -0.64550   0.49555   -1.30  0.19272
## degreeUniversity             -0.93411   0.51588   -1.81  0.07019
##
## For results from individual imputed datasets, use summary(x, subset = i:j)
## Next step: Use 'setx' method
```