MLE - Lab 10

Andy Ballard

March 22, 2017

First, let's set up our workspace

Today

- Summary tables in LaTeX
- Missing data and imputation (Amelia and MICE)

Descriptive Summary Statistics

Let's learn how to show some descriptive summary statistics in a table in LaTeX. Fortunately, there is a package our there that allows one to easily display descriptive summary statistics for variables in LaTeX.

Let us use some data to illustrate.

```
LDC <- read.dta(paste0(labPath, "LDC_IO_replication.dta"))
```

We'll use the package `reporttools` for this.

```
# Pick out variables we want
varsLDC <- LDC[, c("newtar", "fdignp", "polityiv_update2", "gdp_pc_95d")]
# Make a caption
capLDC <- "Descriptive Statistics: LDC dataset"
# Make the table
tableContinuous(vars = varsLDC, cap = capLDC, lab = "tab: cont1", longtable = F,
    prec = 2)
```

The output code (which is hidden here to save space) can easily be used in LaTeX.

Imputation of Missing Data

Let's look at some data and run a model.

```
####################################################
# Example model
apsr <- read.dta(paste0(labPath,"apsrfinaldata.dta"))

data <- apsr[,c('gini_net_std', 'polity2', 'ELF_ethnic')]
n <- nrow(data)
set.seed(6886)

dv <- "pg"
ivs <- c("betweenstd", "lngdpstd")

# Run regression with LM
form <- formula(pg ~ betweenstd + lngdpstd)
summary(lm1 <- lm(form, data=apsr))
```

```
##
## Call:
## lm(formula = form, data = apsr)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.44477 -0.26331 -0.00354  0.15106  1.30639
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.047e-10  6.976e-02   0.000  1.00000
## betweenstd  -2.283e-01  7.521e-02  -3.036  0.00406 **
## lngdpstd     7.997e-01  7.521e-02  10.633  1.3e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4732 on 43 degrees of freedom
## Multiple R-squared:  0.7928, Adjusted R-squared:  0.7831
## F-statistic: 82.25 on 2 and 43 DF,  p-value: 2.013e-15
##################################################
```

Cool, we have a model, but as you can see we didn't have any missing data. I guess we're done, y'all can go home.

Gotcha.

Let's introduce some random missingness to our dataset

```
set.seed(6886)
missMatrix <- cbind(rbinom(n, 1, .9),rbinom(n, 1, .85),rbinom(n, 1, .83)) #3 new variables with differe
missMatrix[missMatrix==0] <- NA #make 0s NAs
apsrMiss <- apsr[,c(dv, ivs)] * missMatrix #convert those proportions of missingness to the IVs and DV

# Check out new dimensions
dim(na.omit(apsrMiss))
```

```
## [1] 31  3
```

Simply throwing away missing data is increasingly being recognized in our field as bad practice, which is why we've been learning about what to do with missing data this week. Data that is missing systematically and not arbitrarily can challenge the validity of our regression results. The accuracy of our model profits from the completeness of data.

Imputation allows us to make statistical inferences about missing data values. Our imputed values are based on the data that we have for other observations and for other variables of the unit that has missing values. The most sophisticated imputation techniques compute multiple alternative datasets based on the existing dataset and then make an inference based on those datasets to estimate the final imputed value for the missing values of any given variable (therefore "multiple imputation").

Lets first rerun our regression without the missing values, and compare it to the original model.

```
lm1ListDel <- lm(form, data=apsrMiss) #This is one reason that creating model formula objects is useful
round(summary(lm1ListDel)$'coefficients',3)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.003      0.077  -0.042    0.967
## betweenstd    -0.139      0.084  -1.657    0.109
## lngdpstd       0.840      0.078  10.731    0.000
```

```
# Full data results
round(summary(lm1)$'coefficients',3)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.000      0.070   0.000    1.000
## betweenstd    -0.228      0.075  -3.036    0.004
## lngdpstd       0.800      0.075  10.633    0.000
```

Potentially substantively different! Now let's look at imputing the missing data, using Amelia, which is the most commonly used package for missing data in politcal science. Gary King and his minions put it together a while back.

```
apsrAmelia = amelia(x=apsrMiss, m=1)
```

```
## -- Imputation 1 --
##
##   1  2  3  4  5  6
```

```
# lets just use the second imputed dataset for now
## and reestimate our model
lm1Amelia <- lm(form, data=apsrAmelia$imp$imp1)
round(summary(lm1Amelia)$'coefficients',3)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.019      0.057  -0.336    0.739
## betweenstd    -0.190      0.062  -3.067    0.004
## lngdpstd       0.829      0.065  12.806    0.000
```

Another tool for imputing missing data is `sbgcop`, developed by Peter Hoff (of Duke stats!). `sbgcop` is a bayesian model and thus we sample our way to posterior values. the nsamp parameter controls how many samples we draw, here I just set it to 5000, but you might need to set it higher, so that you can be sure the results have converged.

```
apsrSbgcop <- sbgcop.mcmc(Y = apsrMiss, nsamp = 5000, seed = 6886, verb = FALSE)
```

`apsrSbgcop` is a list comprised of a number of objects, the most relevant for us is Y.impute

In this case it has dimensions: 46 x 3 x 1000 the first two dims correspond to the size of our original dataset and the last dimension, 1000, represents the iterations from our sampler.

```
names(apsrSbgcop)
```

```
## [1] "C.psamp"  "Y.pmean"  "Y.impute" "LPC"
```

```
dim(apsrSbgcop$Y.impute)
```

```
## [1]   46    3 1000
```

We need to account for the fact that it took some time for our sampler to converge, so we need to burn some of our initial draws. below I burn (throw out) the first 50% of the draws we pulled from our bayesian model.

```
toKeep <- (dim(apsrSbgcop$Y.impute)[3]/2 + 1):dim(apsrSbgcop$Y.impute)[3]
apsrSbgcopPost <- apsrSbgcop$Y.impute[,,toKeep] #why two commas?
```

Next we average across these results so that we can have just one consolidated imputed dataset for our analysis.

```
apsrSbgcopPostAvg <- apply(apsrSbgcopPost, c(1,2), mean)

# Now lets add colnames back in and turn this back into a dataframe
colnames(apsrSbgcopPostAvg) <- names(apsrMiss)
```

```
apsrSbgcopPostAvg <- data.frame(apsrSbgcopPostAvg)

# Last lets rerun our model using the imputed data from sbgcop
lm1Sbgcop <- lm(form, data=apsrSbgcopPostAvg)
round(summary(lm1Sbgcop)$'coefficients',3)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.013      0.055  -0.240    0.811
## betweenstd    -0.156      0.067  -2.339    0.024
## lngdpstd       0.831      0.062  13.294    0.000
```

```
# Compare three sets of results:

# True model, i.e., no randomly excluded data
round(summary(lm1)$'coefficients',3)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.000      0.070   0.000    1.000
## betweenstd    -0.228      0.075  -3.036    0.004
## lngdpstd       0.800      0.075  10.633    0.000
```

```
# Listwise deletion
round(summary(lm1ListDel)$'coefficients',3)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.003      0.077  -0.042    0.967
## betweenstd    -0.139      0.084  -1.657    0.109
## lngdpstd       0.840      0.078  10.731    0.000
```

```
# Amelia
round(summary(lm1Amelia)$'coefficients',3)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.019      0.057  -0.336    0.739
## betweenstd    -0.190      0.062  -3.067    0.004
## lngdpstd       0.829      0.065  12.806    0.000
```

```
# Sbgcop
round(summary(lm1Sbgcop)$'coefficients',3)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.013      0.055  -0.240    0.811
## betweenstd    -0.156      0.067  -2.339    0.024
## lngdpstd       0.831      0.062  13.294    0.000
```

```
######################################################
```

Were the imputed models closer to the real model? You bet!

Imputation of real world missing data with Amelia

We'll look at the LDC data set, which is panel data for 138 countries, which has a ton of different variables in it.

```
LDC <- read.dta(paste0(labPath, "LDC_IO_replication.dta"))
```

```
# summary(LDC)
```

As you can see, we have quite a few NAs in this dataset. Let's impute some of the missing values. With a large dataset, we could write an imputation command that would take a looooooong time. One time I had one running over the weeknd and it still didn't finish. That was when I was young and naive (first year) and didn't know a damn thing about computing power so I thought running it on a regular university computer was a good idea. Side note, I wouldn't use INEGI data to measure violent deaths in Mexico, the data are incredibly corrupt. More applicable side note: it doesn't make sense to impute your missing data unless you have values for the vast majority of your data. Let's say half your data contains missing values. You're actually just kind of stuck and shouldn't impute. You can run models and maybe find things, but it might be best to try and get better data. There really isn't a threshhold for this type of thing, so it's up to you to decide.

Anyway, we will demonstrate the imputation of missing data with only a small subset of three countries with a small number of variables (due to time constraints).

```
LDCs <- subset(LDC, ctylabel == "SouthAfrica" | ctylabel == "Turkey" | ctylabel ==
    "Indonesia")

# Let's reduce our dataset to some variables that we might be most
# interested in

keep <- c("ctylabel", "date", "newtar", "fdignp", "gdp_pc_95d", "polityiv_update2",
    "usheg", "lnpop")
LDCs <- LDCs[, keep]
```

What are these variables?

- `ctylabel`: is an identifier for countries
- `date`: year, 1970-2002
- `newtar`: average tariff rates, 1980-1999
- `fdignp`: Net change in foreign direct investment in the reporting country, expressed as % GNP
- `gdp_pc_95d`: GDP per capita in 1995 dollars
- `polityiv_update2`: Policy scores (higher=more democratic)
- `usheg`: US exports and imports as a percentage of world exports and imports
- `lnpop`: Log of the country's population

Two things that you should be aware of when you use Amelia to impute data:

1. The imputation of missing values in large datasets can take a lot of time.
2. Amelia has problems with variables that are perfectly correlated to other variables.

The content below is based on the Amelia guide. A link is provided at the end.

Now let us do the imputation. We will create only 5 imputed datasets (m=5).

```
a.out <- amelia(LDCs, m = 5, ts = "date", cs = "ctylabel")
```

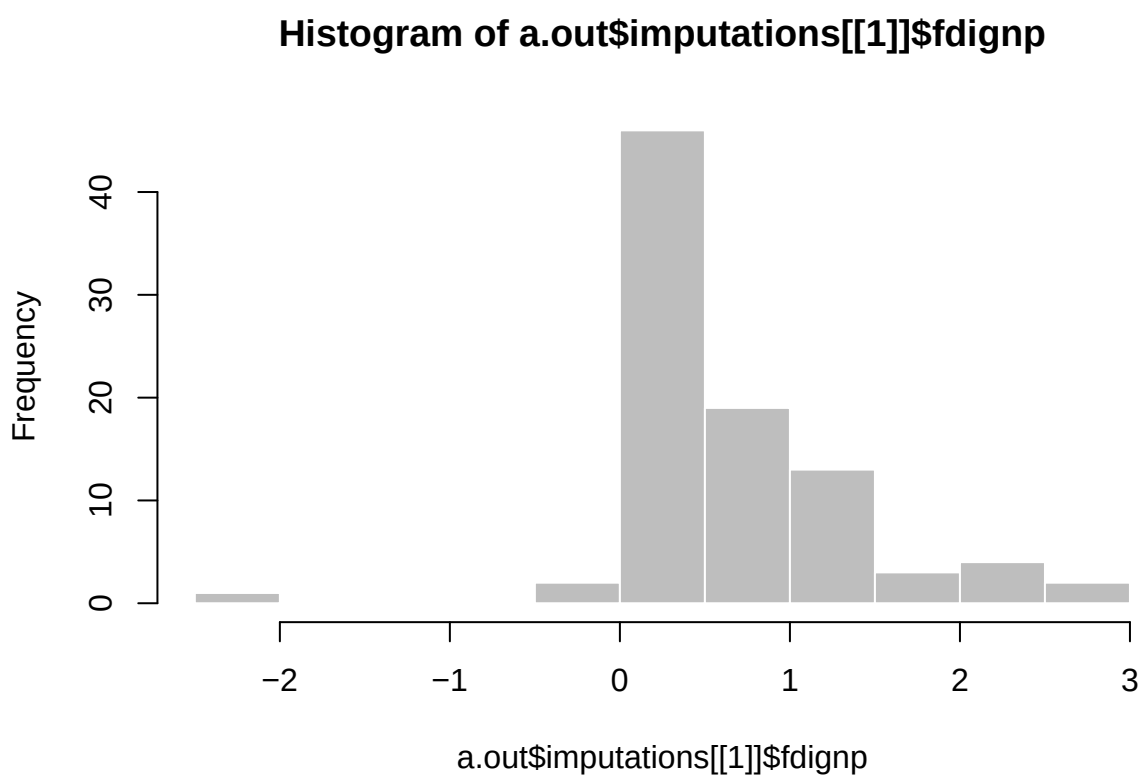
```
## -- Imputation 1 --
##
##   1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
##  41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
##
##
## -- Imputation 2 --
##
##   1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 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
##
## -- 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
##
## -- 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
##
## -- 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
```

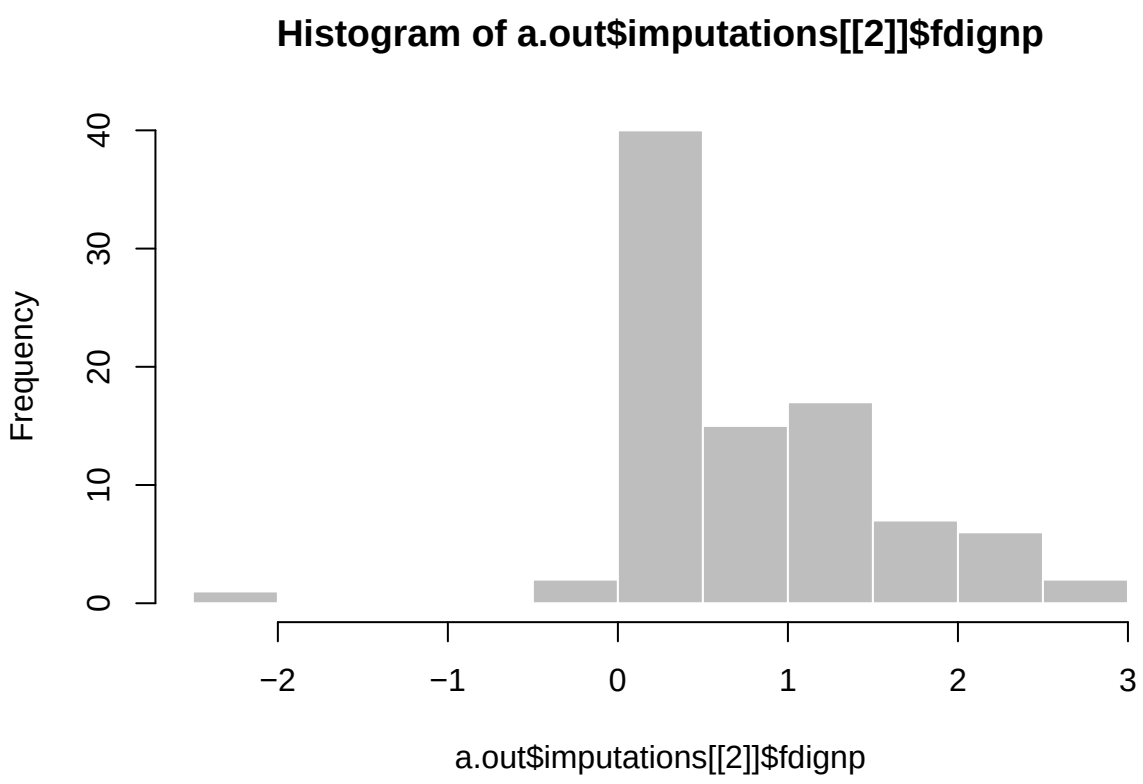
```r
# It is very important to include ts as the time variable and cs as the unit
# variable. Otherwise Amelia would treat all observations as independent
```

Let us have a look at the imputed values of the first three datasets that were generated.

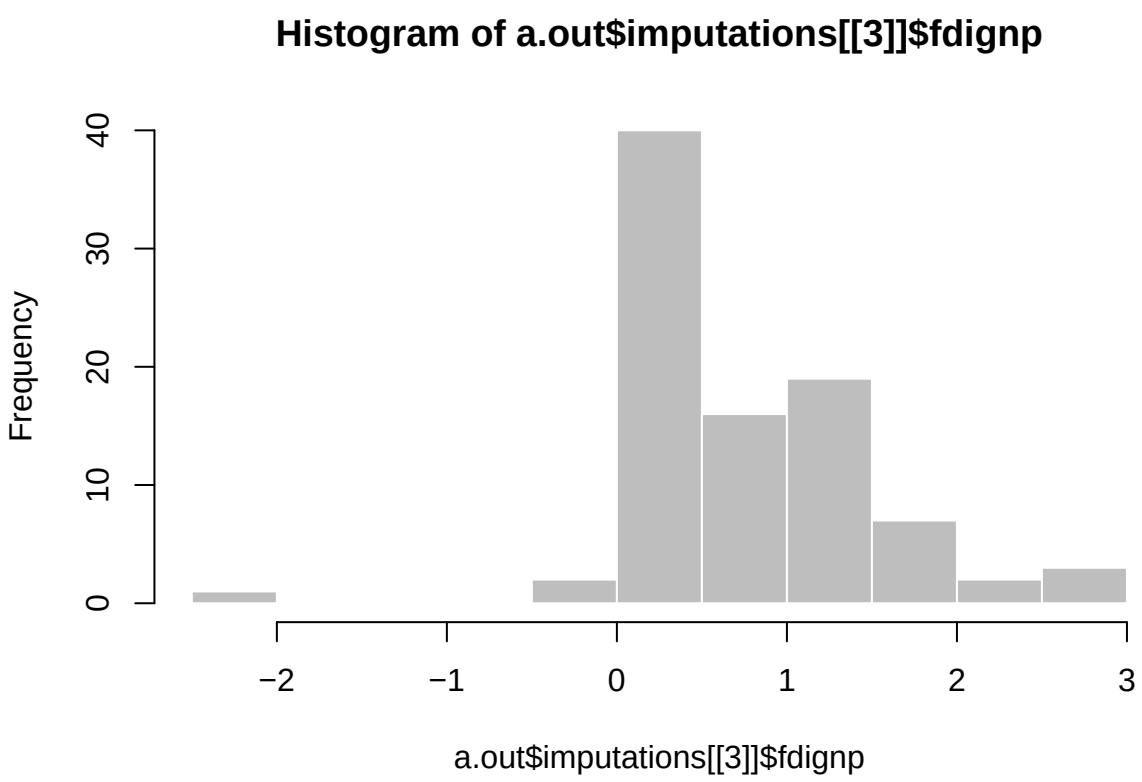
```r
hist(a.out$imputations[[1]]$fdignp, col = "grey", border = "white")
```




```
hist(a.out$imputations[[2]]$fdignp, col = "grey", border = "white")
```




```
hist(a.out$imputations[[3]]$fdignp, col = "grey", border = "white")
```



We can now run an analysis with our imputed dataset. The Amelia package is integrated with the `Zelig` package (both are by Gary King), so we will use this package to estimate a new regression with the imputed data.

```
z.out.imp <- zelig(polityiv_update2 ~ gdp_pc_95d + fdignp, data = a.out$imputations,
    model = "ls")
```

```
## How to cite this model in Zelig:
##   R Core Team. 2007.
##   ls: Least Squares 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/
```

```
# Kind of annoying that it asks you to cite itself when you use the
# function, if you ask me.
summary(z.out.imp)
```

```
## Model: Combined Imputations
##
##              Estimate Std.Error z value Pr(>|z|)
## (Intercept) -4.631664  0.979844   -4.73  2.3e-06
## gdp_pc_95d   0.003617  0.000565    6.40  1.6e-10
## fdignp      -3.128931  0.879032   -3.56  0.00037
##
## For results from individual imputed datasets, use summary(x, subset = i:j)
## Next step: Use 'setx' method
```

One of the cool things about `Zelig` is that we didn't have to specify which of the imputed datasets we're

using. The last time I checked, the default was to average over them. If we were to do this with `lm` it yells at us

```
m.out.imp <- lm(polityiv_update2 ~ gdp_pc_95d + fdignp, data = a.out$imputations)
```

Instead we have to specify which dataset we want to use, or average over it ourselves like the first example.

```
summary(m.out.imp <- lm(polityiv_update2 ~ gdp_pc_95d + fdignp, data = a.out$imputations$imp1))
```

```
##
## Call:
## lm(formula = polityiv_update2 ~ gdp_pc_95d + fdignp, data = a.out$imputations$imp1)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.836 -3.409 -2.087  3.779 12.509
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.6881742  0.9717906  -4.824 5.94e-06 ***
## gdp_pc_95d   0.0033244  0.0003245  10.244  < 2e-16 ***
## fdignp      -2.3404968  0.7018352  -3.335  0.00126 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.557 on 87 degrees of freedom
## Multiple R-squared:  0.5562, Adjusted R-squared:  0.546
## F-statistic: 54.51 on 2 and 87 DF,  p-value: 4.512e-16
```

`Amelia` has a graphical interface that *might* make it easier for you to use it. When I tried to use the graphical interface, however, my R session crashed multiple times and I had to restart it completely. So be cautious and save your work before you use the following command. `Zelig` has had some issues with its dependencies recently. It's nice when it works, but it's pretty finicky. I get the feeling that the authors don't update it much anymore.

```
# AmeliaView()
```

For more information on how to use the package (for your own research) see the very helpful introduction:

https://cran.r-project.org/web/packages/Amelia/vignettes/amelia.pdf

Imputation with MICE

MICE (Multivariate Imputation via Chained Equations) is one of the commonly used package by R users. Creating multiple imputations as compared to a single imputation (such as mean) takes care of uncertainty in missing values.

MICE assumes that the missing data are Missing at Random (MAR), which means that the probability that a value is missing depends only on observed value and can be predicted using them. It imputes data on a variable by variable basis by specifying an imputation model per variable.

For example: Suppose we have $X_1, X_2, ..., X_k$ variables. If X_1 has missing values, then it will be regressed on other variables X_2 to X_k. The missing values in X_1 will be then replaced by predictive values obtained. Similarly, if X_2 has missing values, then X_1, X_3 to X_k variables will be used in prediction model as independent variables. Then, missing values will be replaced with predicted values.

We'll use a native `R` dataset, `iris` (yes, the flowers) to impute data using MICE.

```
data <- iris
head(data)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
identical(dim(iris), dim(na.omit(iris)))
```

```
## [1] TRUE
```

Since there are no missing values, we'll have to create some. This time we'll use the `prodNA` function from the `missForest` package, and create 10% missingness at random.

```
iris.mis <- prodNA(iris, noNA = 0.1)

summary(iris.mis) #NAs!
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.500   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.250   Median :1.350
##  Mean   :5.843   Mean   :3.079   Mean   :3.729   Mean   :1.221
##  3rd Qu.:6.400   3rd Qu.:3.325   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##  NA's   :13      NA's   :14      NA's   :18      NA's   :14
##        Species
##  setosa    :46
##  versicolor:45
##  virginica :43
##  NA's      :16
##
##
##
```

I'm going to remove the categorical variable so we don't have to do multiple methods of imputation. We could encode the factor levels as numbers and follow the same procedure, if we wanted, and put the labels back in later. That would be fine, but we aren't going to right now.

We'll also save a dataframe with the values missing to compare models later.

```
iris.del <- iris.mis

iris.mis <- subset(iris.mis, select = -c(Species))
summary(iris.mis)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.500   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.250   Median :1.350
##  Mean   :5.843   Mean   :3.079   Mean   :3.729   Mean   :1.221
##  3rd Qu.:6.400   3rd Qu.:3.325   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##  NA's   :13      NA's   :14      NA's   :18      NA's   :14
```

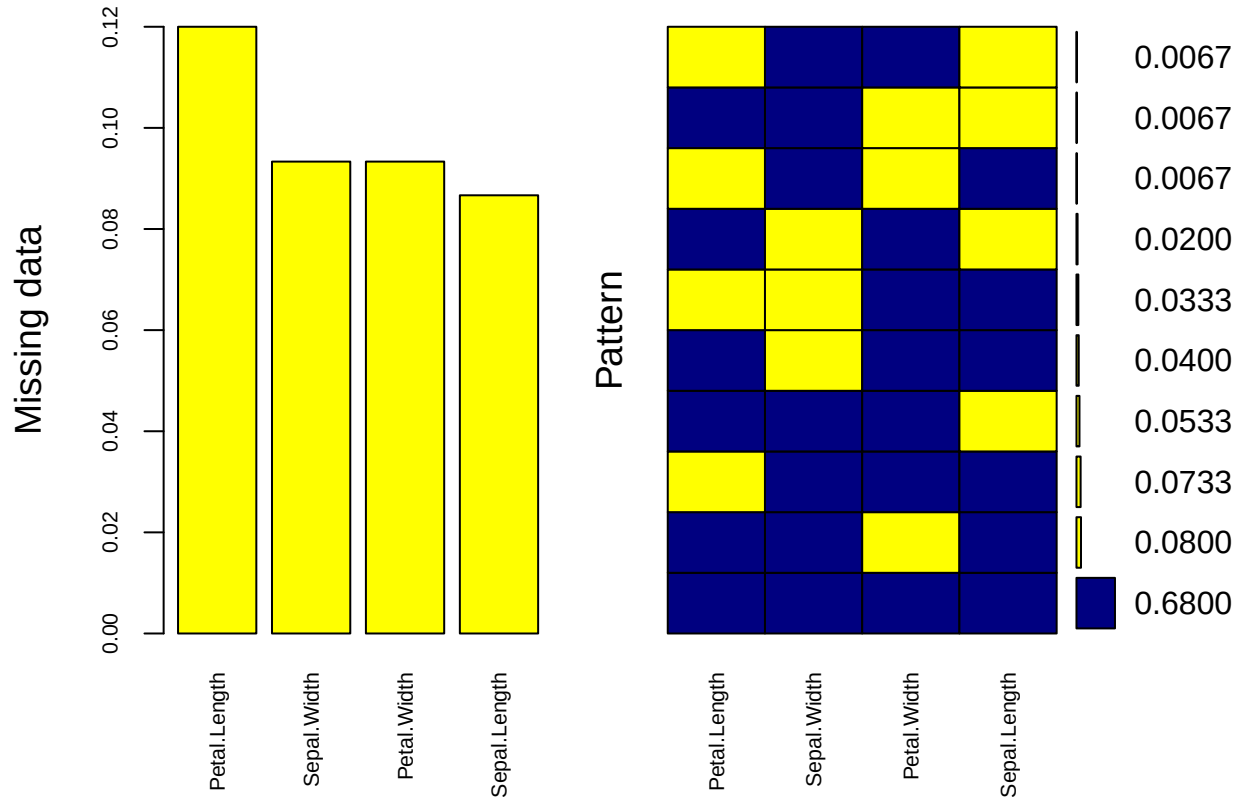
We can look at a table of the missing data with the `md.pattern` function in the `mice` package. What is this table telling us?

```
md.pattern(iris.mis)
```

```
##     Sepal.Length Sepal.Width Petal.Width Petal.Length
## 102            1           1           1            1  0
##   8            0           1           1            1  1
##   6            1           0           1            1  1
##  11            1           1           1            0  1
##  12            1           1           0            1  1
##   3            0           0           1            1  2
##   1            0           1           1            0  2
##   5            1           0           1            0  2
##   1            0           1           0            1  2
##   1            1           1           0            0  2
##               13          14          14           18 59
```

Let's create a visual representation of the missing data, since the table is sort of tough to read.

```
mice_plot <- aggr(iris.mis, col=c('navyblue','yellow'),
                  numbers=TRUE, sortVars=TRUE,
                  labels=names(iris.mis), cex.axis=.7,
                  gap=3, ylab=c("Missing data","Pattern"))
```




```
##
##  Variables sorted by number of missings:
##       Variable      Count
```

```
##  Petal.Length 0.12000000
##   Sepal.Width 0.09333333
##   Petal.Width 0.09333333
##  Sepal.Length 0.08666667
```

Now we can impute the missing values. There are four methods in `MICE` for use with different types of data:

1. PMM (Predictive Mean Matching) - For numeric variables
2. logreg (Logistic Regression) - For Binary Variables(with 2 levels)
3. polyreg (Bayesian polytomous regression) - For Factor Variables (>= 2 levels)
4. Proportional odds model (ordered, >= 2 levels)

Since we have numeric variables, we'll use PMM.

```
imputed_Data <- mice(iris.mis, m = 5, maxit = 50, method = "pmm", seed = 500,
    print = FALSE)
```

```
summary(imputed_Data)
```

```
## Multiply imputed data set
## Call:
## mice(data = iris.mis, m = 5, method = "pmm", maxit = 50, printFlag = FALSE,
##     seed = 500)
## Number of multiple imputations:  5
## Missing cells per column:
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##           13           14           18           14
## Imputation methods:
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##        "pmm"        "pmm"        "pmm"        "pmm"
## VisitSequence:
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##            1            2            3            4
## PredictorMatrix:
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length            0           1            1           1
## Sepal.Width             1           0            1           1
## Petal.Length            1           1            0           1
## Petal.Width             1           1            1           0
## Random generator seed value:  500
```

Here, `m` refers to the number of imputed datasets (5), `maxit` refers to the number of iterations (this took us a while with just 50, often you'll want to use thousands, or tens of thousands). Now let's look at the imputed values.

```
imputed_Data$imp$Sepal.Width
```

```
##       1   2   3   4   5
## 5   3.3 3.5 3.1 3.4 3.4
## 9   2.8 3.0 2.9 3.1 3.0
## 28  3.1 3.7 3.6 3.0 3.4
## 37  3.8 4.2 4.2 3.4 3.4
## 50  3.8 3.8 2.9 3.6 3.4
## 59  3.0 3.4 3.1 3.4 3.3
## 73  2.8 3.0 2.8 3.0 3.0
## 81  2.7 3.2 2.5 3.2 2.7
## 88  3.1 3.1 3.1 3.8 2.8
## 99  3.8 3.6 3.8 3.0 3.2
```

```
## 103 2.7 2.4 2.6 3.1 3.3
## 107 2.9 2.7 2.7 2.6 2.7
## 120 2.8 3.4 3.4 3.0 3.4
## 129 3.2 3.1 3.0 2.8 2.9
```

Now we can run some models.

```
#Imputation with MICE
fit.imp <- with(data = imputed_Data, exp = lm(Sepal.Width ~ Sepal.Length + Petal.Width))

#True model
fit <- lm(Sepal.Width ~ Sepal.Length + Petal.Width, data=iris)

#Model with listwise deletion
fit.del <- lm(Sepal.Width ~ Sepal.Length + Petal.Width, data=iris.del)
```

And we can compare the results.

```
# True model, i.e., no randomly excluded data
round(summary(fit)$'coefficients',3)
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)     1.926      0.321   6.002        0
## Sepal.Length    0.289      0.066   4.380        0
## Petal.Width    -0.466      0.072  -6.501        0
```

```
# Listwise deletion
round(summary(fit.del)$'coefficients',3)
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)     2.149      0.353   6.092        0
## Sepal.Length    0.260      0.072   3.596        0
## Petal.Width    -0.478      0.080  -5.961        0
```

```
# MICE
summary(fit.imp)
```

```
##
##  ## summary of imputation 1 :
##
## Call:
## lm(formula = Sepal.Width ~ Sepal.Length + Petal.Width)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.00834 -0.24160 -0.01687  0.24793  1.01097
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.08934    0.31963   6.537 9.69e-10 ***
## Sepal.Length  0.25821    0.06569   3.931  0.00013 ***
## Petal.Width  -0.43025    0.07108  -6.053 1.13e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3794 on 147 degrees of freedom
## Multiple R-squared:  0.213,  Adjusted R-squared:  0.2023
## F-statistic: 19.89 on 2 and 147 DF,  p-value: 2.262e-08
```

```
##
##
##  ## summary of imputation 2 :
##
## Call:
## lm(formula = Sepal.Width ~ Sepal.Length + Petal.Width)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.04514 -0.22893 -0.01233  0.23097  0.95000
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.92219    0.31035   6.194 5.59e-09 ***
## Sepal.Length  0.30262    0.06367   4.753 4.73e-06 ***
## Petal.Width  -0.49276    0.06879  -7.163 3.52e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3733 on 147 degrees of freedom
## Multiple R-squared:  0.2709, Adjusted R-squared:  0.261
## F-statistic: 27.31 on 2 and 147 DF,  p-value: 8.241e-11
##
##
##  ## summary of imputation 3 :
##
## Call:
## lm(formula = Sepal.Width ~ Sepal.Length + Petal.Width)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.02871 -0.23707 -0.01351  0.23440  0.97854
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.87734    0.32199   5.830 3.38e-08 ***
## Sepal.Length  0.30411    0.06635   4.583 9.70e-06 ***
## Petal.Width  -0.47331    0.07127  -6.641 5.65e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3786 on 147 degrees of freedom
## Multiple R-squared:  0.2398, Adjusted R-squared:  0.2294
## F-statistic: 23.18 on 2 and 147 DF,  p-value: 1.773e-09
##
##
##  ## summary of imputation 4 :
##
## Call:
## lm(formula = Sepal.Width ~ Sepal.Length + Petal.Width)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.01923 -0.24044 -0.00553  0.21648  0.99990
```

```
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.01203    0.30632   6.568 8.23e-10 ***
## Sepal.Length  0.27430    0.06307   4.349 2.54e-05 ***
## Petal.Width  -0.43860    0.06879  -6.376 2.22e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3712 on 147 degrees of freedom
## Multiple R-squared:  0.2255, Adjusted R-squared:  0.2149
## F-statistic: 21.39 on 2 and 147 DF,  p-value: 6.994e-09
##
##
##  ## summary of imputation 5 :
##
## Call:
## lm(formula = Sepal.Width ~ Sepal.Length + Petal.Width)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.01290 -0.23442 -0.00599  0.24417  1.00686
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.04100    0.30556   6.680 4.61e-10 ***
## Sepal.Length  0.26769    0.06304   4.246 3.84e-05 ***
## Petal.Width  -0.43424    0.06921  -6.275 3.72e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3694 on 147 degrees of freedom
## Multiple R-squared:  0.221,  Adjusted R-squared:  0.2104
## F-statistic: 20.85 on 2 and 147 DF,  p-value: 1.071e-08
```