How to Identify and Impute Multiple Missing Values using R.

By Dr. Jon Starkweather
Research and Statistical Support consultant

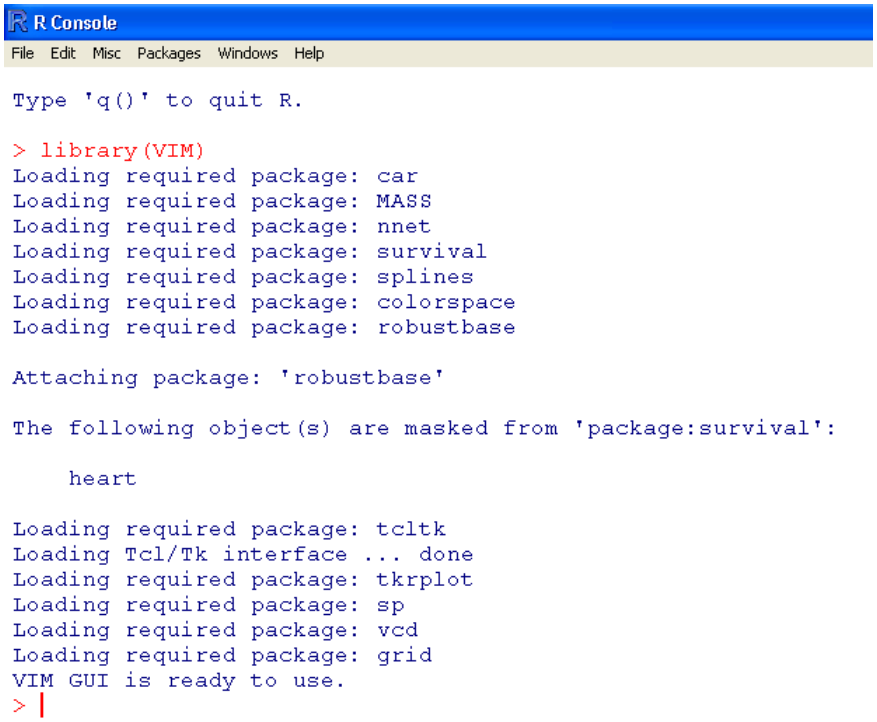
As with many functions or tasks in R, there are a great many ways to accomplish the goals of identifying, displaying, and imputing multiple missing values. The following article discusses only a few ways to identify, display, and impute missing values using three packages in the statistical software R. For those new to R, I would suggest reviewing the Research and Statistical Support (RSS) Do-it-Yourself (DIY) Introduction to R short course. A script file containing all the commands used in this article can be found here.

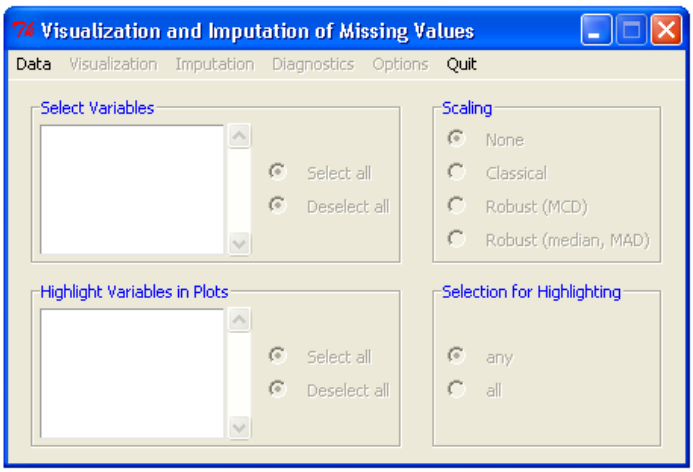
1. Identify and Display Missing Values.

Generally speaking, R identifies missing values with NA. So, running a simple summary(x) where 'x' is the data frame will provide the number of NA's (missing values) for the variable(s). Several examples of the 'summary' function are listed throughout this article.

1.1. The VIM package

The Visualization and Imputation of Missing values package (VIM; Templ, Alfons, & Kowarik, 2010a; Templ, Alfons, & Kowarik, 2010b), provides several functions for identifying and displaying missing data. It provides some very intuitive graphical displays which allow the user to easily identify missing data. Missing data is often displayed in bright red on otherwise grayscale or blue figures. When you load the package, you'll notice two things. First, it has several dependencies and second, it has its own Graphical User Interface (GUI). Generally, I do not use the GUI and instead rely on script which I simply prefer.

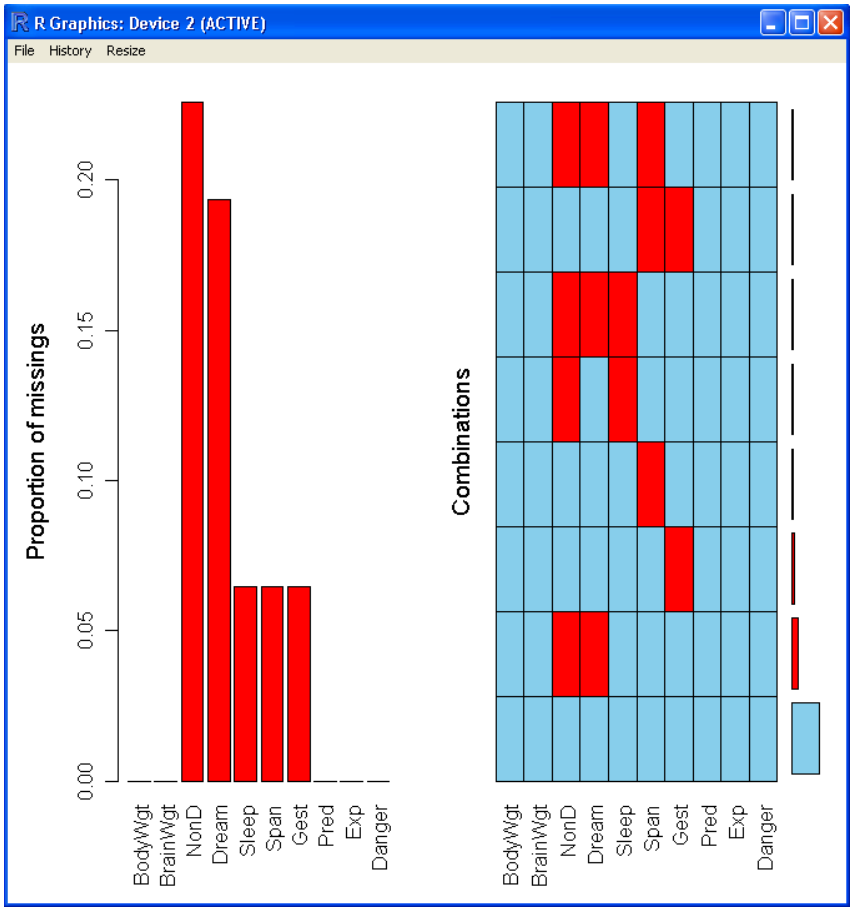




The function 'aggr' aggregates missing data and can be used to count or plot the *amount* of missing-ness for each variable as well as some combinations of variables. Use the examples provided in the documentation to replicate what is provided below (Templ, Alfons, & Kowarik, 2010a).

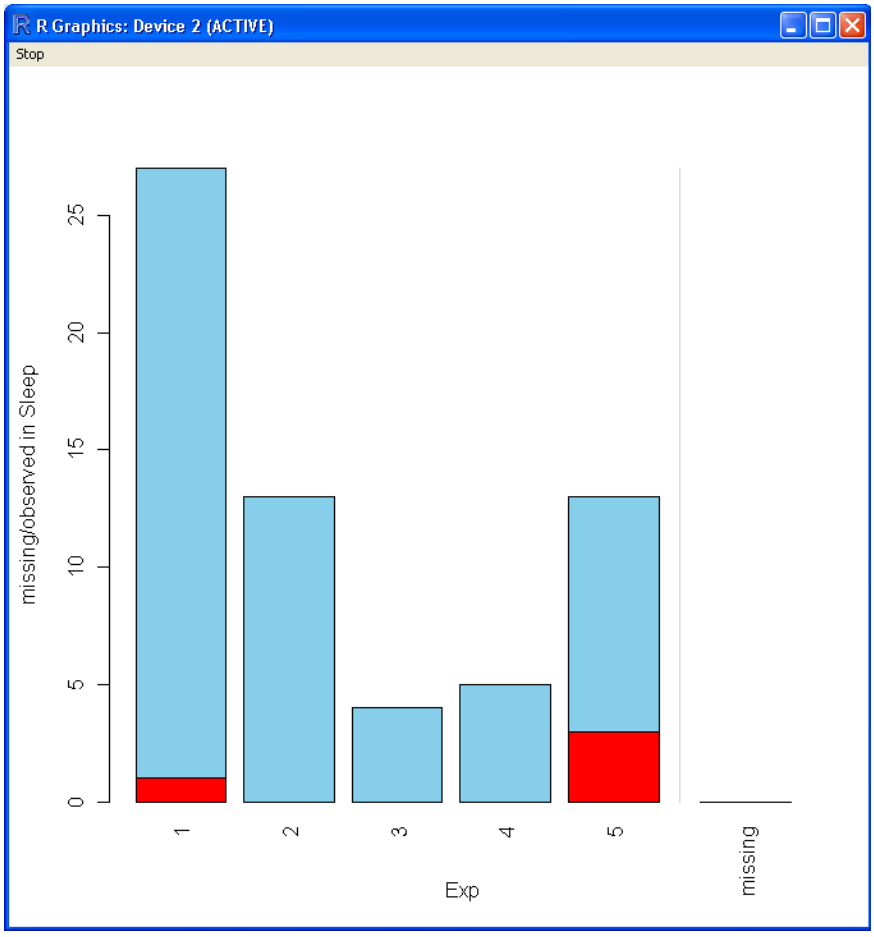
```
> data(sleep)
> a <- aggr(sleep)
> a

 Missings in variables:
 Variable Count
     NonD    14
    Dream    12
    Sleep     4
     Span     4
     Gest     4
>
```

You will also notice the graphical display which shows the proportion of missing-ness for each variable as well as some combinations (displayed below)



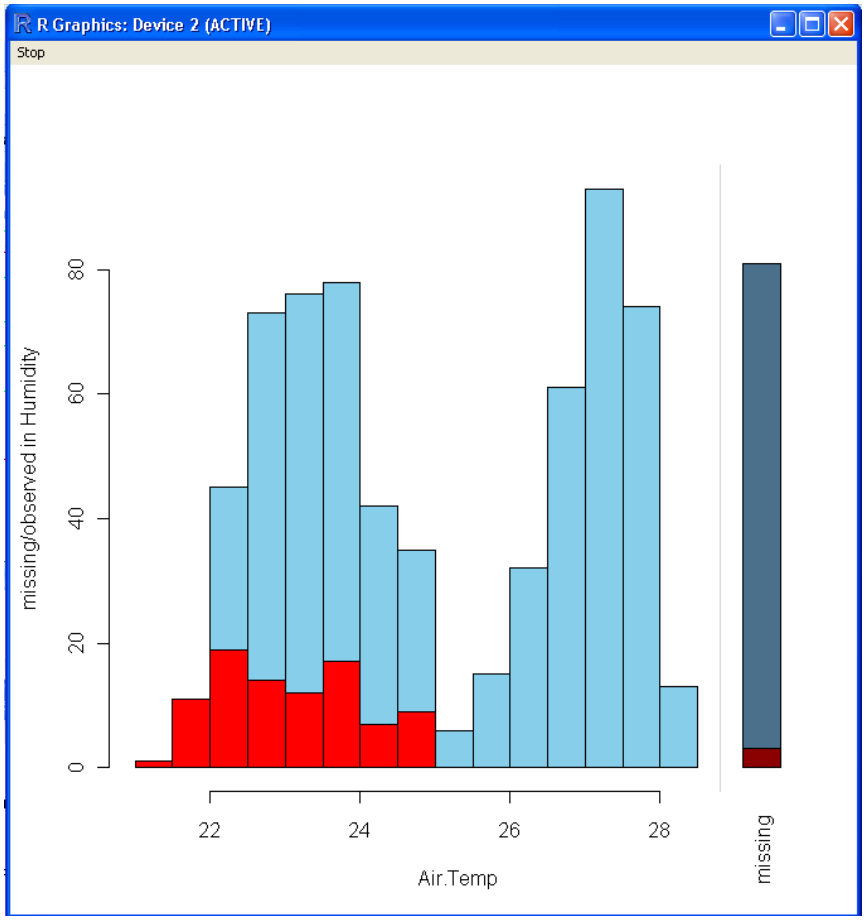
The function 'barMiss' can be used to produce bar charts which display the proportion of missing values of each variable using the color red in the lower part of each bar, the upper portions are displayed in blue.

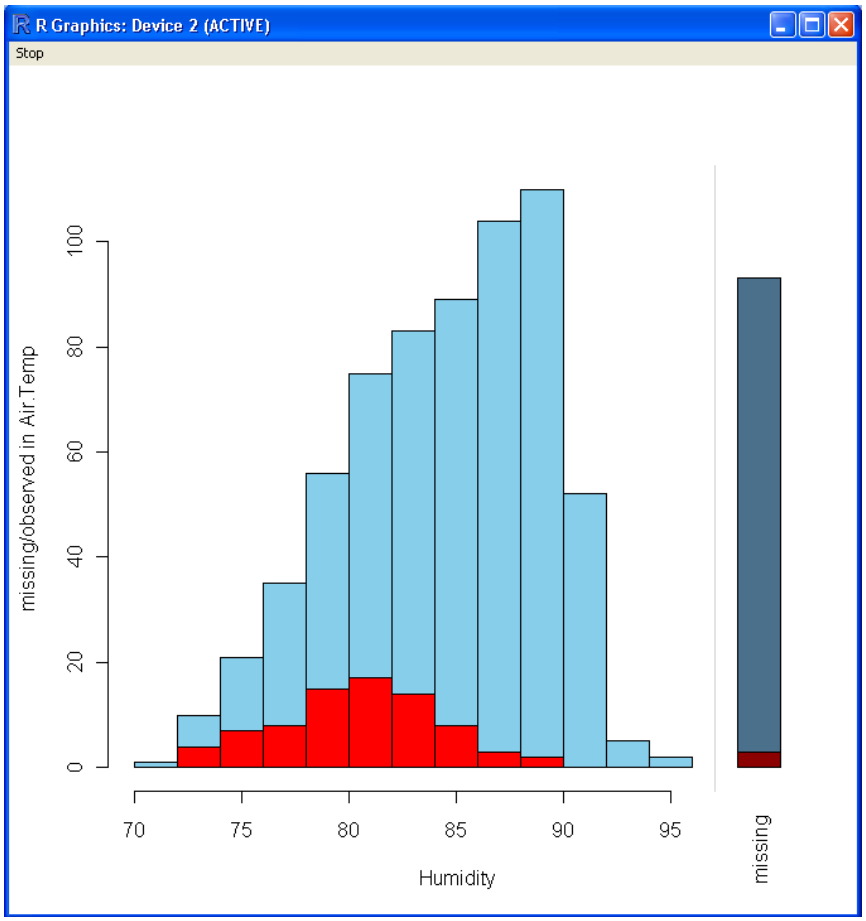
```
> data(sleep)
> x <- sleep[, c("Exp, "Sleep")]
> summary(x)
      Exp            Sleep
 Min.   :1.000   Min.   : 2.60
 1st Qu.:1.000   1st Qu.: 8.05
 Median :2.000   Median :10.45
 Mean   :2.419   Mean   :10.53
 3rd Qu.:4.000   3rd Qu.:13.20
 Max.   :5.000   Max.   :19.90
                 NA's   : 4.00
> barMiss(x)

Click in the left margin to switch to the previous variable or in the
right margin to switch to the next variable. To regain use of the VIM GUI
and the R console, click anywhere else in the graphics window.

>
```



The ‘histMiss’ function performs the same way the ‘barMiss’ function does but, obviously with histograms instead of bar graphs.

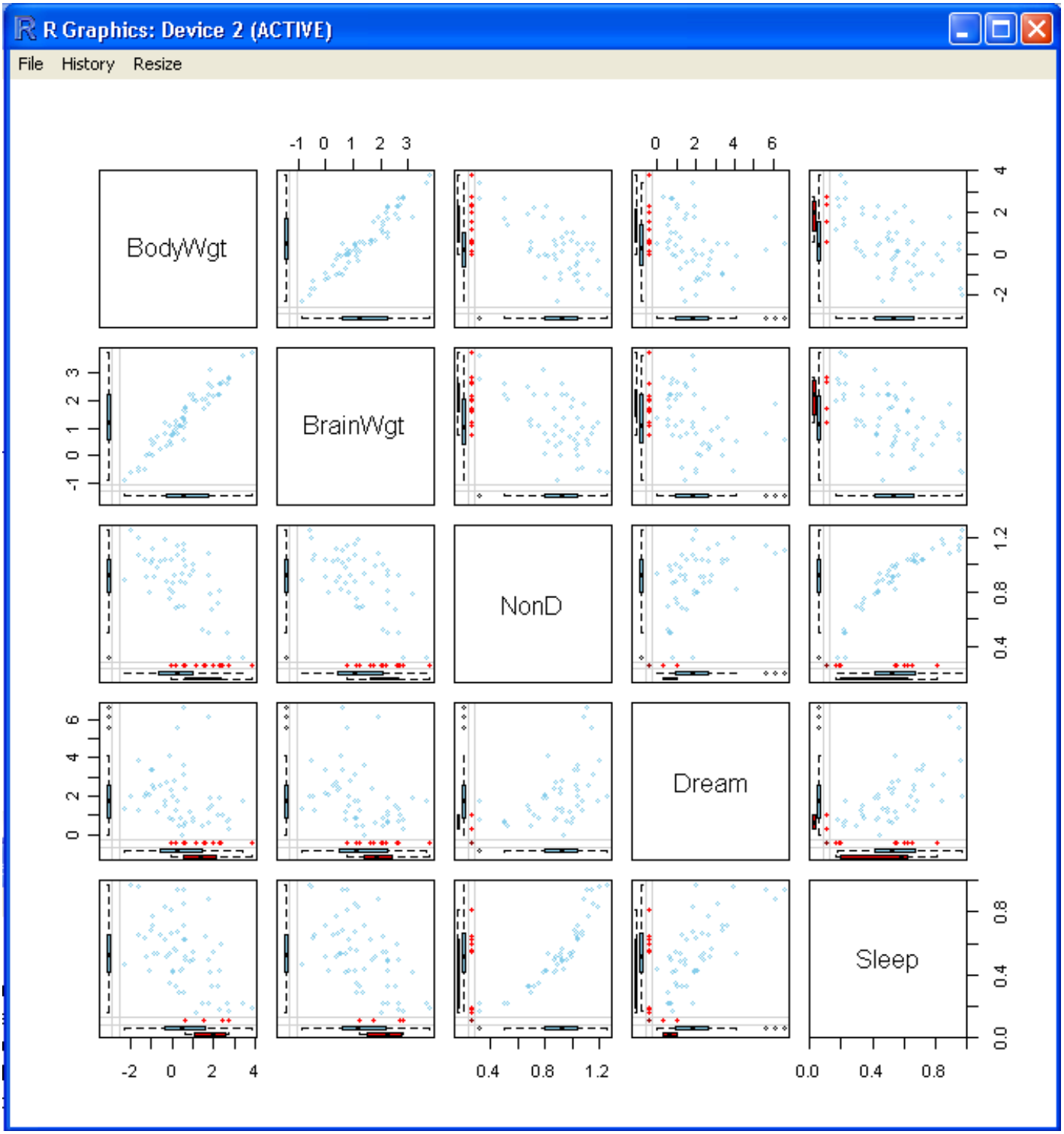
```
> data(tao)
> y <- tao[, c(“Air.Temp”, “Humidity”)]
> summary(y)
    Air.Temp        Humidity
 Min.   :21.42   Min.   :71.60
 1st Qu.:23.26   1st Qu.:81.30
 Median :24.52   Median :85.20
 Mean   :25.03   Mean   :84.43
 3rd Qu.:27.08   3rd Qu.:88.10
 Max.   :28.50   Max.   :94.80
 NA’s   :81.00   NA’s   :93.00
> histMiss(y)

Click in the left margin to switch to the previous variable or in the
right margin to switch to the next variable. To regain use of the VIM GUI
and the R console, click anywhere else in the graphics window.

>
```


R Graphics: Device 2 (ACTIVE)
Stop
missing/observed in Humidity
0
20
40
60
80
22
24
26
28
Air.Temp
missing


R Graphics: Device 2 (ACTIVE)
Stop
missing/observed in Air.Temp
0
20
40
60
80
100
70
75
80
85
90
95
Humidity
missing

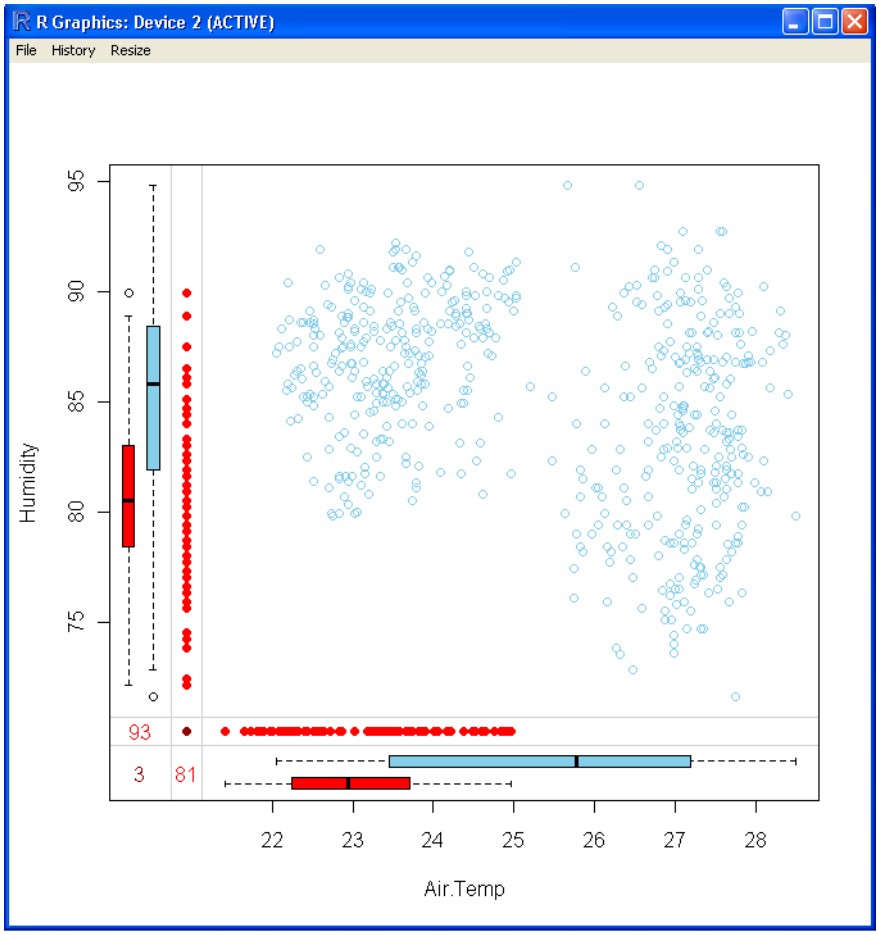
The function ‘marginmatrix’ creates a scatter plot matrix with information about missing values in the plot margins of each panel. In the margins box plots in blue represent the (non-missing) data. Single variable scatter plots and boxplots in red represent missing data and are located along the axis for each variable.

```
> data(sleep)
> z <- sleep[, 1:5]
> z[,c(1,2,3)] <- log10(z[,c(1,2,3)])
> summary(z)
     BodyWgt           BrainWgt          NonD            Dream      
 Min.   :-2.3010   Min.   :-0.8539   Min.   : 0.3222   Min.   : 0.000  
 1st Qu.:-0.2260   1st Qu.: 0.6263   1st Qu.: 0.7958   1st Qu.: 0.900  
 Median : 0.5240   Median : 1.2367   Median : 0.9217   Median : 1.800  
 Mean   : 0.5809   Mean   : 1.3638   Mean   : 0.8927   Mean   : 1.972  
 3rd Qu.: 1.6781   3rd Qu.: 2.2199   3rd Qu.: 1.0414   3rd Qu.: 2.550  
 Max.   : 3.8231   Max.   : 3.7568   Max.   : 1.2529   Max.   : 6.600  
                                     NA’s   :14.0000   NA’s   :12.000  
    Sleep      
 Min.   : 2.60  
 1st Qu.: 8.05  
 Median :10.45  
 Mean   :10.53  
 3rd Qu.:13.20  
 Max.   :19.90  
 NA’s   : 4.00  
> marginmatrix(z)
>
```



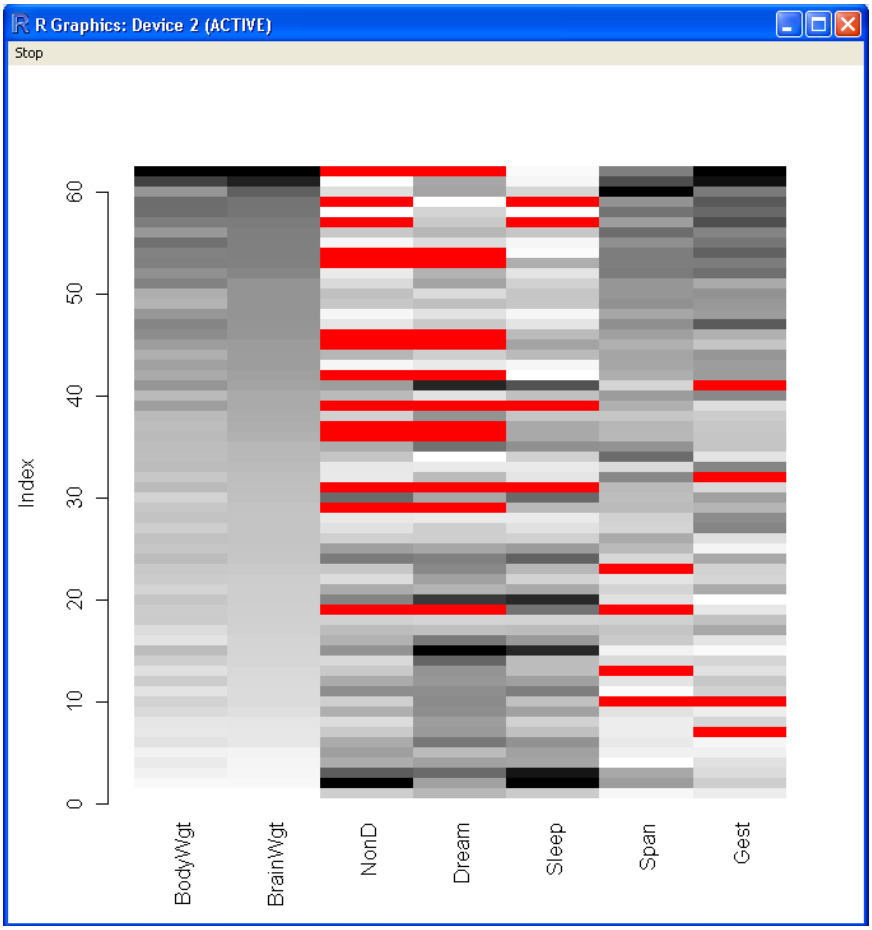
The function ‘marginplot’ performs essentially the same operation as ‘marginmatrix’ but for a standard two variable scatter plot – which makes it much easier to see and interpret. The red numbers (81 & 93) are the number of missing values for each variable; the single number in the lower right-most panel represents the number of cases which are missing values for both variables.

```
> data(tao)
> marginplot(tao[,c("Air.Temp", "Humidity")])
>
```



The function ‘matrixplot’ creates a color matrix plot in which the data cells are represented by a colored rectangle. Each cell is color coded along a continuum from white to black by default and missing data cells are given a clearly recognizable color (i.e. bright red by default). The data matrix plot can also be sorted by clicking inside the plot space on the variable’s column which you want to sort by.

```
> data(sleep)
> b <- sleep[, -(8:10)]
> b[,c(1,2,4,6,7)] <- log10(b[,c(1,2,4,6,7])
> matrixplot(b, sortby = “BrainWgt”)

Click in a column to sort by the corresponding variable.
To regain use of the VIM GUI and the R console, click outside the plot
region.

Warning message:
In matrixplot(b, sortby = “BrainWgt”) :
  variable ‘Dream’ contains infinite values
>
```



2. Imputation Techniques for Missing Values.

2.1 The VIM Package.

The VIM package can also be used to do multiple imputation using the 'irmi' function which does what it stands for, Iterative Robust Model-based Imputation. The function runs iterative regression analysis in which each iteration uses one variable as an outcome and the remaining variables as predictors. If the outcome has any missing values, the predicted values from the regression are imputed. Iterations end when all variables in the data frame have served as an outcome. Again, using the package documentation provided examples allows a brief introduction to the function (Templ, Alfons, & Kowarik, 2010a). Notice below, the variables Sea.Surface.Temp, Air.Temp, and Humidity all have missing values.

```
> data(tao)
> summary(tao)
      Year         Latitude       Longitude      Sea.Surface.Temp
 Min.   :1993   Min.   :-5.000   Min.   :-110.0   Min.   :21.60
 1st Qu.:1993   1st Qu.:-2.000   1st Qu.:-110.0   1st Qu.:23.50
 Median :1995   Median :-1.000   Median :-102.2   Median :26.55
 Mean   :1995   Mean   :-1.375   Mean   :-102.5   Mean   :25.86
 3rd Qu.:1997   3rd Qu.: 0.000   3rd Qu.: -95.0   3rd Qu.:28.21
 Max.   :1997   Max.   : 0.000   Max.   : -95.0   Max.   :30.17
                                                  NA's   : 3.00
    Air.Temp        Humidity         UWind           VWind
 Min.   :21.42   Min.   :71.60   Min.   :-8.100   Min.   :-6.200
 1st Qu.:23.26   1st Qu.:81.30   1st Qu.:-5.100   1st Qu.: 1.500
 Median :24.52   Median :85.20   Median :-3.900   Median : 2.900
```

```
 Mean   :25.03   Mean   :84.43   Mean   :-3.716   Mean   : 2.636
 3rd Qu.:27.08   3rd Qu.:88.10   3rd Qu.:-2.600   3rd Qu.: 4.100
 Max.   :28.50   Max.   :94.80   Max.   : 4.300   Max.   : 7.300
 NA's   :81.00   NA's   :93.00
> imputed.tao <- irmi(tao)
> summary(imputed.tao)
      Year         Latitude       Longitude      Sea.Surface.Temp
 Min.   :1993   Min.   :-5.000   Min.   :-110.0   Min.   :21.60
 1st Qu.:1993   1st Qu.:-2.000   1st Qu.:-110.0   1st Qu.:23.50
 Median :1995   Median :-1.000   Median :-102.2   Median :26.43
 Mean   :1995   Mean   :-1.375   Mean   :-102.5   Mean   :25.86
 3rd Qu.:1997   3rd Qu.: 0.000   3rd Qu.: -95.0   3rd Qu.:28.21
 Max.   :1997   Max.   : 0.000   Max.   : -95.0   Max.   :30.17
    Air.Temp         Humidity        UWind          VWind
 Min.   :21.42   Min.   :71.60   Min.   :-8.100   Min.   :-6.200
 1st Qu.:23.38   1st Qu.:81.60   1st Qu.:-5.100   1st Qu.: 1.500
 Median :25.11   Median :85.30   Median :-3.900   Median : 2.900
 Mean   :25.26   Mean   :84.71   Mean   :-3.716   Mean   : 2.636
 3rd Qu.:27.15   3rd Qu.:88.20   3rd Qu.:-2.600   3rd Qu.: 4.100
 Max.   :29.08   Max.   :95.89   Max.   : 4.300   Max.   : 7.300
>
```

2.2. The Amelia Package.

Another way of dealing with missing data is to use the Amelia package. The Amelia package (Honaker, King, & Blackwell, 2010a) is specifically designed to do multiple imputation on a variety of data types, as long as the data is in a matrix or data frame. The imputation function is the 'amelia' function, which creates new data sets which include multiple imputation of incomplete multivariate data values in place of missing values by running a bootstrapped EM algorithm. The 'amelia' function has a variety of optional arguments, including the ability to provide an initial priors matrix and bounds for missing values. Working with the documentation provided examples offers a brief introduction to the function (Honaker, et al., 2010a).

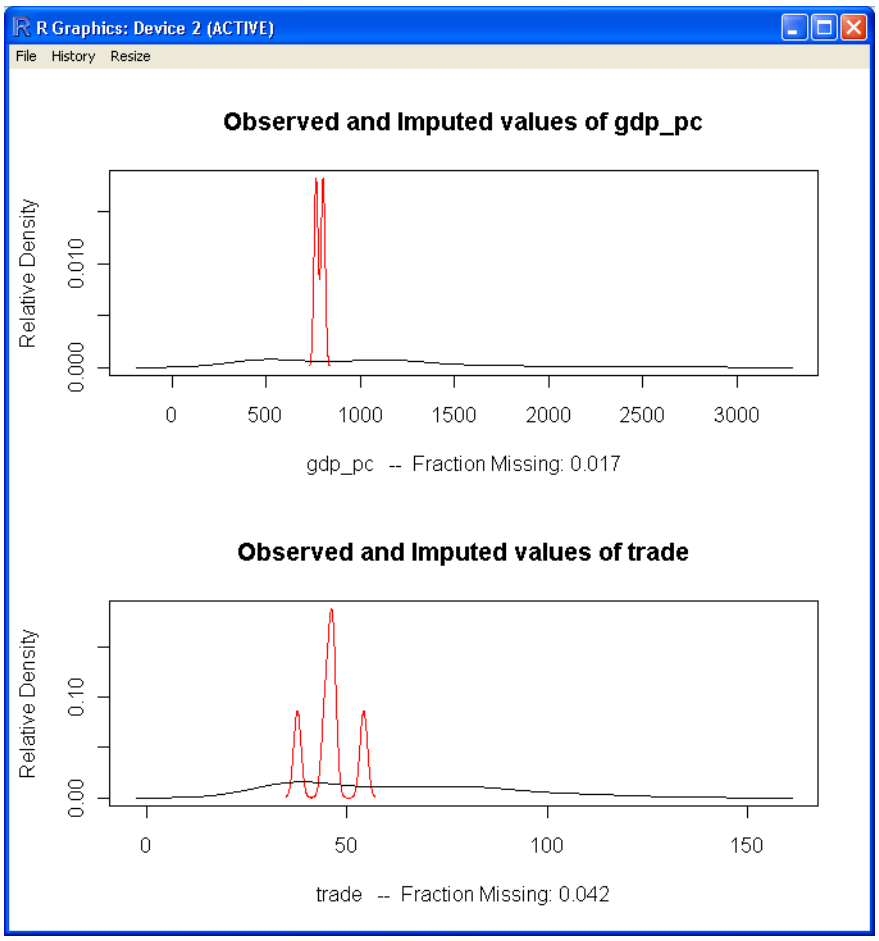
```
> library(Amelia)
Loading required package: foreign
##
## Amelia II: Multiple Imputation
## (Version 1.2-18, built: 2010-11-04)
## Copyright (C) 2005-2010 James Honaker, Gary King and Matthew Blackwell
## Refer to http://gking.harvard.edu/amelia/ for more information
##
> data(africa)
> summary(africa)
      year              country       gdp_pc            infl
 Min.   :1972   Burkina Faso:20   Min.   : 376.0   Min.   : -8.400
 1st Qu.:1977   Burundi     :20   1st Qu.: 513.8   1st Qu.:  4.760
 Median :1982   Cameroon    :20   Median :1035.5   Median :  8.725
 Mean   :1982   Congo       :20   Mean   :1058.4   Mean   : 12.753
 3rd Qu.:1986   Senegal     :20   3rd Qu.:1244.8   3rd Qu.: 13.560
 Max.   :1991   Zambia      :20   Max.   :2723.0   Max.   :127.890
                                  NA's   :   2.0
     trade            civlib          population
 Min.   : 24.35   Min.   :0.0000   Min.   : 1332490
 1st Qu.: 38.52   1st Qu.:0.1667   1st Qu.: 4332190
```

```
 Median : 59.59   Median :0.1667   Median : 5853565
 Mean   : 62.60   Mean   :0.2889   Mean   : 5765594
 3rd Qu.: 81.16   3rd Qu.:0.3333   3rd Qu.: 7355000
 Max.   :134.11   Max.   :0.6667   Max.   :11825390
 NA's   :  5.00
>
```

Next, we can use the 'amelia' function to create the new data set(s). Notice the summary (below) tells us there were "5 imputed datasets" created. We could increase the number of data sets created by changing 'm=5' (default) to whatever number of data sets we wanted; however, Honaker, King, and Blackwell (2010b) state "unless the rate of missing-ness is very high, *m* = 5 (the program default) is probably adequate" (p. 4).

```
> a.out <- amelia(x=africa,m=5,cs="country",ts="year",logs="gdp_pc")
-- Imputation 1 --

 1  2  3

-- Imputation 2 --

 1  2  3

-- Imputation 3 --

 1  2

-- Imputation 4 --

 1  2  3  4

-- Imputation 5 --

 1  2

> summary(a.out)

Amelia output with 5 imputed datasets.
Return code:  1
Message:  Normal EM convergence.

Chain Lengths:
--------------
Imputation 1:  3
Imputation 2:  3
Imputation 3:  2
Imputation 4:  4
Imputation 5:  2

Rows after Listwise Deletion:  115
Rows after Imputation:  120
Pattern of missingness in the data:  3

Fraction Missing for individual variables:
-----------------------------------------
```

```
           Fraction Missing
year             0.00000000
country          0.00000000
gdp_pc           0.01666667
infl             0.00000000
trade            0.04166667
civlib           0.00000000
population       0.00000000
> plot(a.out)
>
```



Next, we can write the data sets created and store them (by default) in our working directory. The following function 'write.amelia' takes all the imputed data sets created using the 'amelia' function and writes them as new data files. In this case, specified with the names: "africa.outdata1.csv", "africa.outdata1.csv", "africa.outdata3.csv", "africa.outdata4.csv", and "africa.outdata5.csv". The 'csv' extension refers to *comma separated values* which is a form of text (.txt) data file with values separated by commas.

```
> write.amelia(a.out,"africa.outdata",extension=NULL,format="csv")
>
```

Now we can load any of these 5 new data sets into R from the working directory. Generally, the last iteratively produced data set offers the best estimates of the missing values/data because; it is based on the previous estimates (i.e. priors).

```
> a.out5 <- read.table("C:/Documents and Settings/user/Desktop/WorkStuff/
            africa.outdata5",header=TRUE,sep=",",na.strings="NA",dec=".",
            strip.white=TRUE)
>
```

Now, we can perform a summary to take a look at how the missing values may have changed the central tendency and or distribution of the variables.

```
> summary(a.out5)
>
       X                year              country       gdp_pc
 Min.   :  1.00   Min.   :1972   Burkina Faso:20   Min.   : 376.0
 1st Qu.: 30.75   1st Qu.:1977   Burundi     :20   1st Qu.: 511.8
 Median : 60.50   Median :1982   Cameroon    :20   Median :1015.5
 Mean   : 60.50   Mean   :1982   Congo       :20   Mean   :1048.6
 3rd Qu.: 90.25   3rd Qu.:1986   Senegal     :20   3rd Qu.:1232.2
 Max.   :120.11   Max.   :1991   Zambia      :20   Max.   :2723.0
      infl              trade            civlib         population
 Min.   : -8.400   Min.   : 24.35   Min.   :0.0000   Min.   : 1332490
 1st Qu.:  4.760   1st Qu.: 38.41   1st Qu.:0.1667   1st Qu.: 4332190
 Median :  8.725   Median : 58.84   Median :0.1667   Median : 5853565
 Mean   : 12.753   Mean   : 61.52   Mean   :0.2889   Mean   : 5765594
 3rd Qu.: 13.560   3rd Qu.: 80.79   3rd Qu.:0.3333   3rd Qu.: 7355000
 Max.   :127.890   Max.   :134.11   Max.   :0.6667   Max.   :11825390
>
```

2.3. The mvnmle Package.

Another way of dealing with missing data involves using the 'mvnmle' package (Gross & Bates, 2009) to create a complete variance/covariance matrix which will include maximum likelihood estimates for missing values. Notice, this is very different from the previous two methods. The previous methods were concerned with retrieving a new (imputed) data file. The mvnmle method is concerned only with a complete variance/covariance matrix based on maximum likelihood values imputed where previously missing values existed. This can be useful for some multivariate analysis (e.g. structural equation modeling, principal components analysis, etc.). Again we will be using the examples provided in the package documentation (Gross & Bates, 2009).

```
> library(mvnmle)
> data(apple)
> summary(apple)
      size           worms
 Min.   : 4.00   Min.   :27.00
 1st Qu.: 6.50   1st Qu.:38.75
 Median :12.50   Median :44.00
 Mean   :14.72   Mean   :45.00
 3rd Qu.:21.25   3rd Qu.:53.75
 Max.   :40.00   Max.   :59.00
                 NA's   : 6.00
```

Take a look at the covariance matrix for 'apple'.

```
> cov(apple)
         size worms
size  94.8065    NA
worms      NA    NA
>
```

Notice that because of the 6 missing values on the variable 'worms' we get 'NA' for 3 of the 4 entries of the variance/covariance matrix. We can conduct the multiple imputation using the 'mlest' function, which applies maximum likelihood estimates for missing values so that the variance/covariance matrix can be computed.

```
> mlest(apple)
$muhat
```

```
[1] 14.72227 49.33325

$sigmahat
          [,1]      [,2]
[1,]  89.53415 -90.69653
[2,] -90.69653 114.69470

$value
[1] 148.4350

$gradient
[1]  4.996200e-06  2.891530e-06  9.105833e-07  1.684765e-05  -1.073488-04

$hessian
NULL

$stop.code
[1] 1

$iterations
[1] 34

>
```

To extract only the variance/covariance matrix and assign it a name (imputed.cov.apple):

```
> imputed.cov.apple <- mlest(apple)$sigmahat
> imputed.cov.apple
          [,1]      [,2]
[1,]  89.53415 -90.69653
[2,] -90.69653 114.69470
```

Then, this matrix can be sent to another function for the primary analysis.

2.4. The SeqKnn and rrcovNA Packages.

Finally, another way of dealing with missing data is the *k* nearest neighbor (knn) approach. This method is quite simple in principle but is effective and often preferred over some of the more sophisticated methods described above. Nearest neighbors are records that have similar completed data patterns; the average of the *k*-nearest neighbor's completed data are used to impute the value for a variable that is missing it's value (where *k* can be set by the analyst or R user). Hastie, et al., (1999) have shown a *k* ranging from 5 to 10 is adequate. The advantage of the knn approach is that it assumes data are missing at random (MAR) meaning, missing data only depends on the observed data; which in turn means, the knn approach is able to take advantage of multivariate relationships in the completed data. The disadvantage of this approach is it does not include a component to model random variation; consequently uncertainty in the imputed value is underestimated. As an example of the simplicity of the knn approach, consider the following:

Data frame:

```
-----------------------------------------------
case v1 v2 v3 v4 v5 v6
 1    3  3  4  3  4  4 -|
 2    3  3  4  3  4  4  |- 4 nearest
 3    3  2  4  4  4  4  |    neighbors
 4    3  2  4  4  4  4 -|
 5    3  2  4  ?  4  4 --- missing v3
```

```
-----------------------------------------------
5    3  2  4  ?  4  4 before imputation
5    3  2  4 3.5 4  4 after imputation
                |
          imputed value
```

To implement the knn approach in R, Kim and Yi (2009) have made available the 'SeqKnn' package, which performs a sequential knn procedure using the 'SeqKnn' function. Again, using the example provided in the package documentation offers a quick introduction to the function. It is a simple function which simply uses the data name (matrix or data frame) and *k* = the user defined number of nearest neighbors (*k* = 10 below).

```
> library(SeqKnn)
> data(khan05)
> imputed.k05 <- SeqKNN(khan05,10)
2208
>
```

Summaries were not included above because; the khan05 dataset has 64 variables and the summary outputs would fill an unnecessary amount of space in this article. To get the summaries for comparison, simply type:
summary(khan05)
summary(imputed.k05)

The package 'rrcovNA' (Todorov, 2010) also has a function for conducting sequential nearest neighbor imputation ('impSeq'), as well as a function ('impSeqRob') which is a robust variant of the former. Similar to the 'SeqKNN' in terms of simplicity, the function 'impSeq' simply requires the data in matrix or data frame format. The difference between the 'impSeq' function and the 'SeqKNN' from above is the manner in which distances between neighboring cases are determined. The 'SeqKNN' function uses Euclidean distances while 'impSeq' uses statistical measures of distance (mean & covariance). In the case of 'impSeqRob' the distances are determined by robust estimates of location and scatter. The 'rrcovNA' package requires several other packages (listed below in the output). Again, using the examples provided in the library documentation shows how easy it is to use these functions.

```
> library(rrcovNA)
Loading required package: rrcov
Loading required package: robustbase
Loading required package: pcaPP
Loading required package: mvtnorm
Scalable Robust Estimators with High Breakdown Point (version 1.1-00)
Loading required package: norm
Scalable Robust Estimators with High Breakdown Point for
Incomplete Data (version 0.3-00)
> data(bush10)
> summary(bush10)
       V1              V2              V3              V4
 Min.   : 78.00   Min.   : 66.0   Min.   : 10.0   Min.   :110.0
 1st Qu.: 88.00   1st Qu.:108.5   1st Qu.:185.5   1st Qu.:200.0
 Median : 94.00   Median :137.0   Median :260.5   Median :215.0
 Mean   : 99.85   Mean   :130.3   Mean   :278.1   Mean   :230.6
 3rd Qu.:112.00   3rd Qu.:155.2   3rd Qu.:378.5   3rd Qu.:246.0
 Max.   :146.00   Max.   :181.0   Max.   :577.0   Max.   :344.0
 NA's   :  5.00   NA's   :  2.0   NA's   :  6.0   NA's   :  5.0
       V5
 Min.   :188.0
 1st Qu.:260.0
 Median :273.0
```

```
 Mean   :284.1
 3rd Qu.:301.0
 Max.   :380.0
```

Below is an example of the standard ‘impSeq’ function.

```
> imputed.b10 <- impSeq(bush10)
> summary(imputed.b10)
       V1               V2              V3              V4
 Min.   : 78.00   Min.   : 66.0   Min.   : 10.0   Min.   :110.0
 1st Qu.: 88.25   1st Qu.:105.5   1st Qu.:187.2   1st Qu.:193.1
 Median :100.50   Median :137.0   Median :252.9   Median :213.5
 Mean   :102.92   Mean   :129.7   Mean   :288.9   Mean   :227.8
 3rd Qu.:113.00   3rd Qu.:155.0   3rd Qu.:379.5   3rd Qu.:246.0
 Max.   :146.00   Max.   :181.0   Max.   :599.7   Max.   :344.0
       V5
 Min.   :188.0
 1st Qu.:260.5
 Median :274.5
 Mean   :286.6
 3rd Qu.:301.0
 Max.   :380.0
```

Below is an example of the robust sequential imputation, ‘impSeqRob’ function with the default value of alpha shown. Also notice when retrieving the imputed data from the output of the ‘impSeqRob’ function, you must apply a dollar sign and x to the name you provided (dataname$x).

```
> rob.imputed.b10 <- impSeqRob(bush10, alpha=0.9)
> summary(rob.imputed.b10$x)
       V1               V2              V3              V4
 Min.   : 73.39   Min.   : 66.0   Min.   : 10.0   Min.   :110.0
 1st Qu.: 88.00   1st Qu.:105.5   1st Qu.:189.3   1st Qu.:192.7
 Median : 97.00   Median :137.0   Median :255.6   Median :213.5
 Mean   :102.73   Mean   :129.2   Mean   :288.9   Mean   :227.5
 3rd Qu.:113.00   3rd Qu.:155.0   3rd Qu.:379.5   3rd Qu.:246.0
 Max.   :156.85   Max.   :181.0   Max.   :589.4   Max.   :344.0
       V5
 Min.   :188.0
 1st Qu.:260.5
 Median :274.5
 Mean   :286.6
 3rd Qu.:301.0
 Max.   :380.0
>
```

3. Conclusions

Keep in mind; the techniques discussed in this article represent a very small percentage of the available methods for identifying, displaying, and imputing missing values. A *partial* list of packages implementing various functions to handle missing values and missing value imputations is given below (below the References and Resources section). Additionally, the CRAN Multivariate Task View (Hewson, 2010) has a listing of several packages and what they can do for missing data. Also notice that most of the packages discussed above contain more functions than the ones reviewed here. Lastly, there are some limitations to the techniques discussed above. Most assume the data are multivariate normal. Also, the mlest function is limited to 50 variables or less.

Until next time; you may say I’m a dreamer, but I’m not the only one…

4. References and Resources

Gross, K., & Bates, D. (2009). Package ‘mvnmle’. Available at:
http://cran.r-project.org/web/packages/mvnmle/mvnmle.pdf

Hastie, T., Tibshirani, R., Sherlock, G., Eisen, M., Brown, P. and Botstein, D., Imputing Missing Data for Gene Expression Arrays, Stanford University Statistics Department Technical report (1999),
http://www-stat.stanford.edu/~hastie/Papers/missing.pdf

Hewson, P. (2010). CRAN Task View: Multivariate Statistics. Available at:
http://cran.r-project.org/web/views/Multivariate.html

Honaker, J., King, G., & Blackwell, M. (2010a). Package ‘Amelia’. Available at:
http://cran.r-project.org/web/packages/Amelia/Amelia.pdf

Honaker, J., King, G., & Blackwell, M. (2010b). Package ‘Amelia’ vignette. Available at:
http://cran.r-project.org/web/packages/Amelia/vignettes/amelia.pdf

Kim, K., & Yi, G. (2009). Package ‘SeqKnn’. Available at:
http://cran.r-project.org/web/packages/SeqKnn/SeqKnn.pdf

Templ, M., Alfons, A., & Kowarik, A. (2010a). Package ‘VIM’. Available at:
http://cran.r-project.org/web/packages/VIM/VIM.pdf

Templ, M., Alfons, A., & Kowarik, A. (2010a). Package ‘VIM’ vignette. Available at:
http://cran.r-project.org/web/packages/VIM/vignettes/VIM-EU-SILC.pdf

Templ, M. (2010). CRAN Task View: Official Statistics & Survey Methodology. Available at:
http://cran.r-project.org/web/views/OfficialStatistics.html

Todorov, V. (2010). Package ‘rrcovNA’. Available at:
http://cran.r-project.org/web/packages/rrcovNA/rrcovNA.pdf

5. Packages implementing various functions to handle missing values and missing value imputations (note: this is only a partial list):

Amelia
arrayImpute
bcv
cat
crank
CVThresh
crank
compositions
Design
dprep
eigenmodel
EMV
FAwR
Hmisc

impute
imputeMDR
MADAM
mclust
Mfuzz
mi
mitools
mice
missMDA
mimR
mix
mix
MImix
MIfuns
monomvn
mvnmle
norm
nnc
optmatch
pan
pcaMethods
prabclus
rama
randomForest
rconfifers
relaimpo
robCompositions
rrp
scrime
SDisc
simsalabim
VIM
vmv
yaImpute