B01-2

遺失值、離群值處理

吳漢銘

國立臺北大學 統計學系

本章大綱&學習目標

- 遺失值 (Missing Data)
- Visualization of missing values R package: VIM
  - Mammal Sleep Data
- Missingness Mechanism
  - Missing by Design，Missing Completely at Random，Missing at Random (MAR), Missing Not at Random (MNAR)，
- Missing Values in R
- Traditional Approaches to Handling Missing Data
- Advanced Imputation Methods
- R Packages for Dealing With Missing Values
  - MICE，Amelia，mi, Hmisc
- Outliers Detection
- Robust Statistical Methods

遺失值 (Missing Data)

- When data are missing for a variable for all cases, that particular variable is referred to as **latent** or **unobserved**.
  - In a survey context this would be a variable which is not present in the questionnaire.
- When data are missing for all variables for a given case, we have what is known as **unit non-response**.
  - In a survey context this would be an object (sample establishment) that did not complete and/or return the questionnaire at all.
- Missing data (missing values for certain variables for certain cases) are also referred to as **item non-response**.
  - i.e. when some of the questions in the questionnaire are left unanswered.

	A	B	C	D	E	F	G
1	ID	C	Y	X1	X2	X3	X4
2	s1	1	78.3	69.6	74.3	NA	5.22
3	s2	2	77	69.9	72.54	NA	3.98
4	s3	3	72.2	65.7	69.74	NA	4.89
5	s4	1	33.4	NA	30.97	NA	21.54
6	s5	2	32.65	28.35	30.54	NA	9.82
7	s6	3	35.45	28.5	32.01	NA	19.81
8	s7	1	424	378	403.55	NA	12.98
9	s8	2	NA	NA	NA	NA	NA
10	s9	3	355	312.5	339.96	NA	14.14
11	s10	1	18.2	15.5	17.19	NA	13.93
12	s11	2	18.3	15.3	16.38	NA	6.92
13	s12	3	16.1	13.9	14.92	NA	10.15
14	s13	1	23.75	20.2	22.19	NA	32.81

遺失值的處理

- The missing values may give clues to systematic aspects of the problem. Ignore the tuple, you cannot make use of the remaining values except the missing one.

- **How to deal with missing values:**
  - Filling the missing value manually is not applicable for large datasets.
  - Use a global constant to fill the value will misguide the mining process.
  - Use a measure for a central tendency for the attribute to fill the missing value for symmetric data distribution.
  - Use the attribute mean or median for all samples belonging to the same class as the given tuple.

- **補值 (Missing value imputation) (most popular):**
  - The missing data can be filled with data determined with regression, inference-based tool, such as Bayesian formalism or decision tree induction.

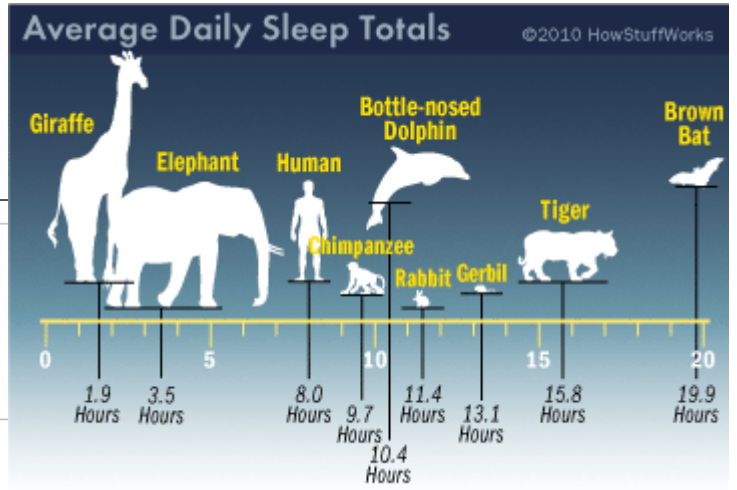
Mammal Sleep Data

- Allison and Cicchetti (1976): 62 mammal species on the interrelationship between sleep, ecological, and constitutional variables.
- Young et al. (2006) use it to illustrate visualization techniques for missing values in ViSta.
- Variables:
  - **Species**: Species of animal
  - **BodyWgt**: body weight (kg)
  - **BrainWgt**: brain weight (g)
  - **NonD**: slow wave ("nondreaming") sleep (hrs/day)
  - **Dream**: paradoxical ("dreaming") sleep (hrs/day)
  - **Sleep**: total sleep, sum of slow wave and paradoxical sleep (hrs/day)
  - **Span**: maximum life span (years)
  - **Gest**: gestation time (days)
  - **Pred**: predation index (1-5), 1 = minimum (least likely to be preyed upon); 5 = maximum (most likely to be preyed upon)
  - **Exp**: sleep exposure index (1-5), 1 = least exposed (e.g. animal sleeps in a well-protected den); 5 = most exposed
  - **Danger**: overall danger index (1-5) (based on the above two indices and other information), 1 = least danger (from other animals); 5 = most danger (from other animals)
- Analysis
  - Use multiple regression to predict the three sleep variables after suitably transforming the predictors and response. The response variable could be hours of sleep or the proportion of sleep spent dreaming.

Data file download: http://www.statsci.org/data/general/sleep.html
Also available at Mammal sleep data, mammalsleep {mice},
http://artax.karlin.mff.cuni.cz/r-help/library/mice/html/mammalsleep.html

Mammal Sleep Data

Image Source: http://www.stuff.co.nz/national/blogs/in-our-nature?page=2



Mammal Sleep Data

```
> data(sleep, package = "VIM")
> dim(sleep)
[1] 62 10
> head(sleep)
   BodyWgt BrainWgt NonD Dream Sleep Span Gest Pred Exp Danger
1 6654.000   5712.0   NA    NA   3.3 38.6  645    3   5      3
2    1.000      6.6  6.3   2.0   8.3  4.5   42    3   1      3
3    3.385     44.5   NA    NA  12.5 14.0   60    1   1      1
4    0.920      5.7   NA    NA  16.5   NA   25    5   2      3
5 2547.000   4603.0  2.1   1.8   3.9 69.0  624    3   5      4
6   10.550    179.5  9.1   0.7   9.8 27.0  180    4   4      4
```

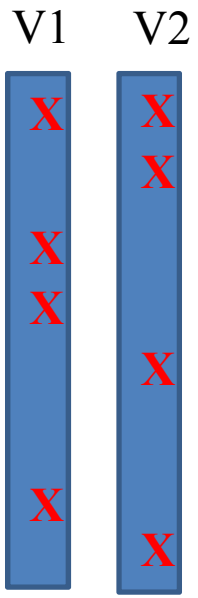
```
> summary(sleep)
    BodyWgt            BrainWgt             NonD            Dream            Sleep      
 Min.   :   0.005   Min.   :   0.14   Min.   : 2.100   Min.   :0.000   Min.   : 2.60  
 1st Qu.:   0.600   1st Qu.:   4.25   1st Qu.: 6.250   1st Qu.:0.900   1st Qu.: 8.05  
 Median :   3.342   Median :  17.25   Median : 8.350   Median :1.800   Median :10.45  
 Mean   : 198.790   Mean   : 283.13   Mean   : 8.673   Mean   :1.972   Mean   :10.53  
 3rd Qu.:  48.203   3rd Qu.: 166.00   3rd Qu.:11.000   3rd Qu.:2.550   3rd Qu.:13.20  
 Max.   :6654.000   Max.   :5712.00   Max.   :17.900   Max.   :6.600   Max.   :19.90  
                                      NA's   :14       NA's   :12      NA's   :4      
      Span             Gest             Pred            Exp            Danger     
 Min.   :  2.000   Min.   : 12.00   Min.   :1.000   Min.   :1.000   Min.   :1.000  
 1st Qu.:  6.625   1st Qu.: 35.75   1st Qu.:2.000   1st Qu.:1.000   1st Qu.:1.000  
 Median : 15.100   Median : 79.00   Median :3.000   Median :2.000   Median :2.000  
 Mean   : 19.878   Mean   :142.35   Mean   :2.871   Mean   :2.419   Mean   :2.613  
 3rd Qu.: 27.750   3rd Qu.:207.50   3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:4.000  
 Max.   :100.000   Max.   :645.00   Max.   :5.000   Max.   :5.000   Max.   :5.000  
 NA's   :4         NA's   :4                                                       
```

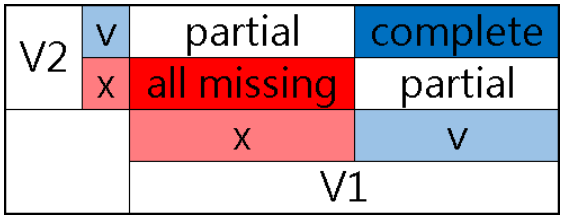
NOTE: Log-transformation of the **`BodyWgt, BrainWgt`** is necessary to obtain more symmetric distributions.

Visualization of missing values
R package: VIM

Visualization functions

- **barMiss**: Barplot with information about missing/imputed values
- **bgmap**: Backgound map
- **bubbleMiss**: Growing dot map with information about missing/imputed values
- **colormapMiss**: Colored map with information about missing/imputed values
- **colormapMissLegend**: Colored map with information about missing/imputed values
- **growdotMiss**: Growing dot map with information about missing/imputed values
- **histMiss**: Histogram with information about missing/imputed values
- **iimagMiss**: Matrix plot
- **initialise**: Initialization of missing values
- **mapMiss**: Map with information about missing/imputed values
- **marginmatrix**: Marginplot Matrix
- **marginplot**: Scatterplot with additional information in the margins
- **matrixplot**: Matrix plot
- **mosaicMiss**: Mosaic plot with information about missing/imputed values
- **pairsVIM**: Scatterplot Matrices
- **parcoordMiss**: Parallel coordinate plot with information about missing/imputed values
- **pbox**: Parallel boxplots with information about missing/imputed values
- **scattJitt**: Bivariate jitter plot
- **scattmatrixMiss**: Scatterplot matrix with information about missing/imputed values
- **scattMiss**: Scatterplot with information about missing/imputed values
- **spineMiss**: Spineplot with information about missing/imputed values





Imputation functions

- **hotdeck**: Hot-Deck Imputation
- **irmi**: Iterative robust model-based imputation
- **kNN**: k-Nearest Neighbour Imputation
- **regressionImp**: Regression Imputation

VIM: Visualization and Imputation of Missing Values
https://cran.r-project.org/web/packages/VIM/index.html
Matthias Templ , Andreas Alfons, Peter Filzmoser, 2012, Exploring incomplete data using visualization techniques, Advances in Data Analysis and Classification, 6(1), 29-47.

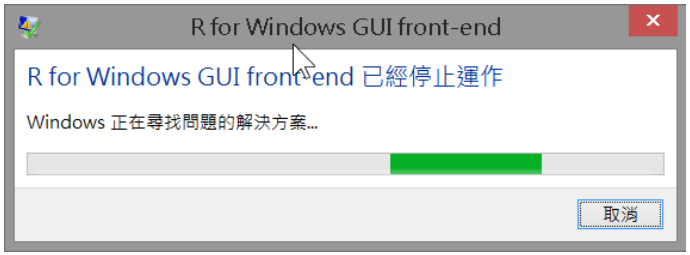
https://www.r-bloggers.com/graphical-presentation-of-missing-data-vim-package/

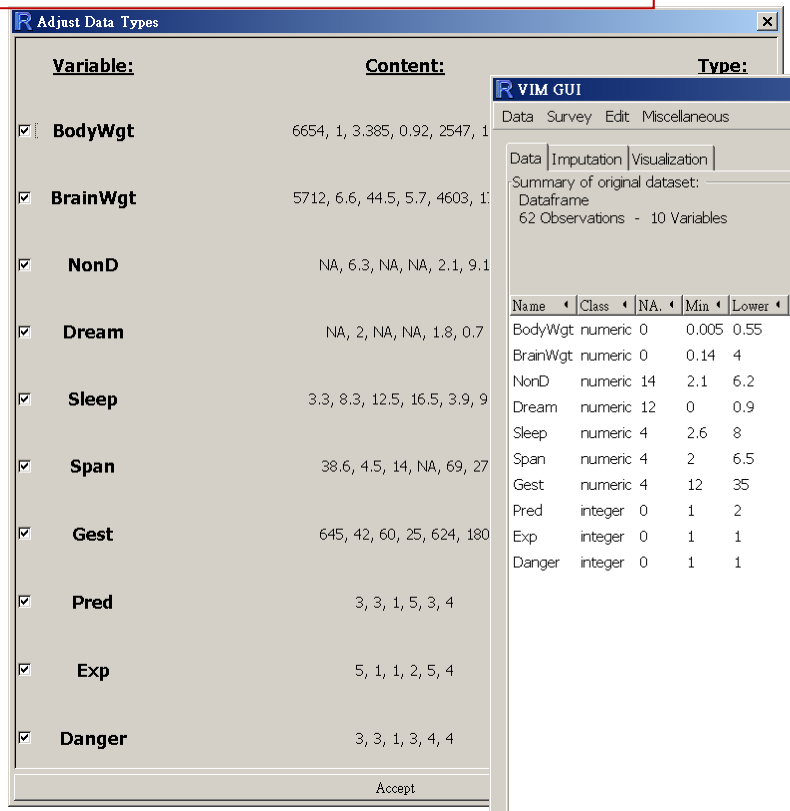
VIMGUI

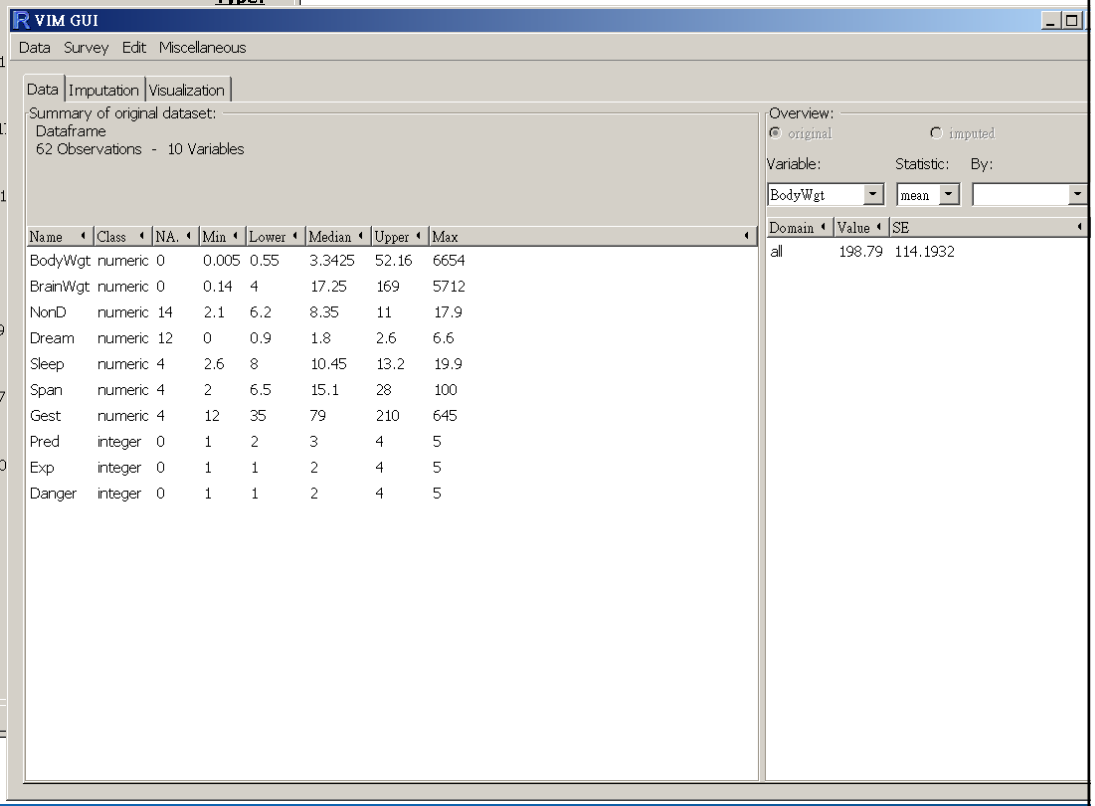
```
install.packages("VIM", "VIMGUI")
data(sleep, package = "VIM")
data(tao, package = "VIM")
data(SBS5242)
data(kola.background, package = "VIM")

library(VIMGUI)
VIMGUI()
```

狀況一

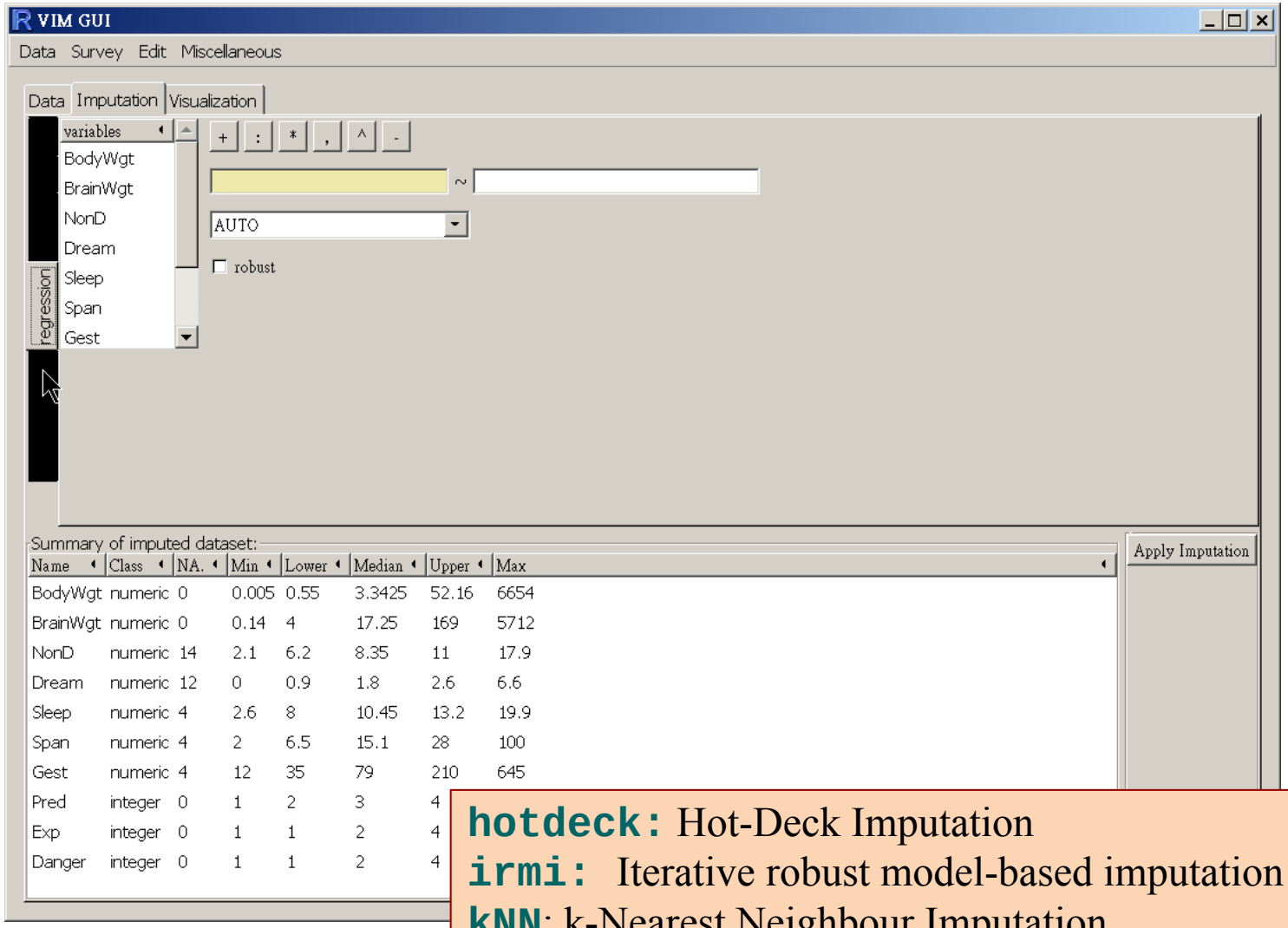
狀況二







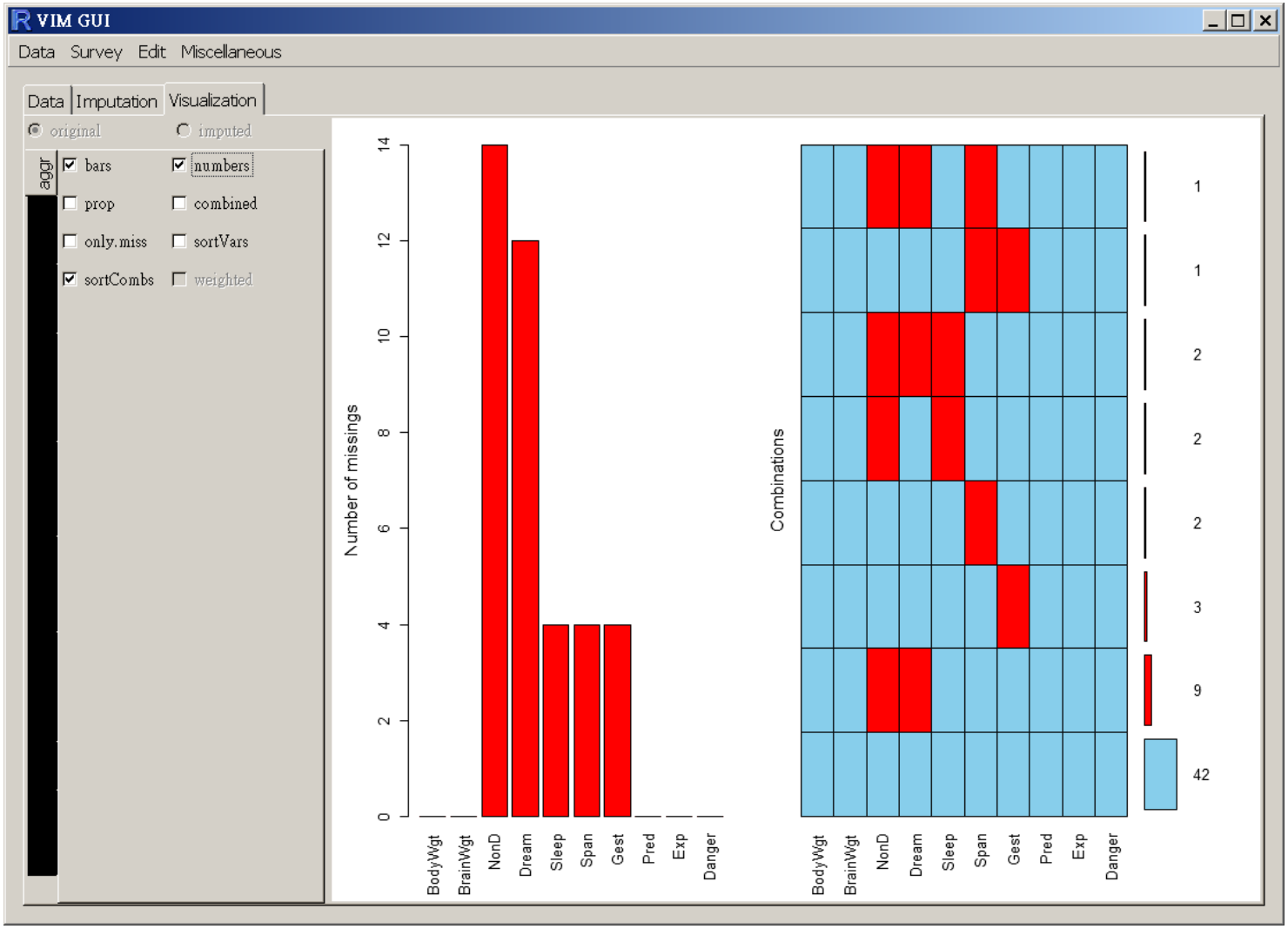
VIMGUI: Imputation tab



Name	Class	NA.	Min	Lower	Median	Upper	Max
BodyWgt	numeric	0	0.005	0.55	3.3425	52.16	6654
BrainWgt	numeric	0	0.14	4	17.25	169	5712
NonD	numeric	14	2.1	6.2	8.35	11	17.9
Dream	numeric	12	0	0.9	1.8	2.6	6.6
Sleep	numeric	4	2.6	8	10.45	13.2	19.9
Span	numeric	4	2	6.5	15.1	28	100
Gest	numeric	4	12	35	79	210	645
Pred	integer	0	1	2	3	4	
Exp	integer	0	1	1	2	4	
Danger	integer	0	1	1	2	4	

`hotdeck:` Hot-Deck Imputation
`irmi:` Iterative robust model-based imputation (IRMI)
`kNN`: k-Nearest Neighbour Imputation
`regressionImp:` Regression Imputation

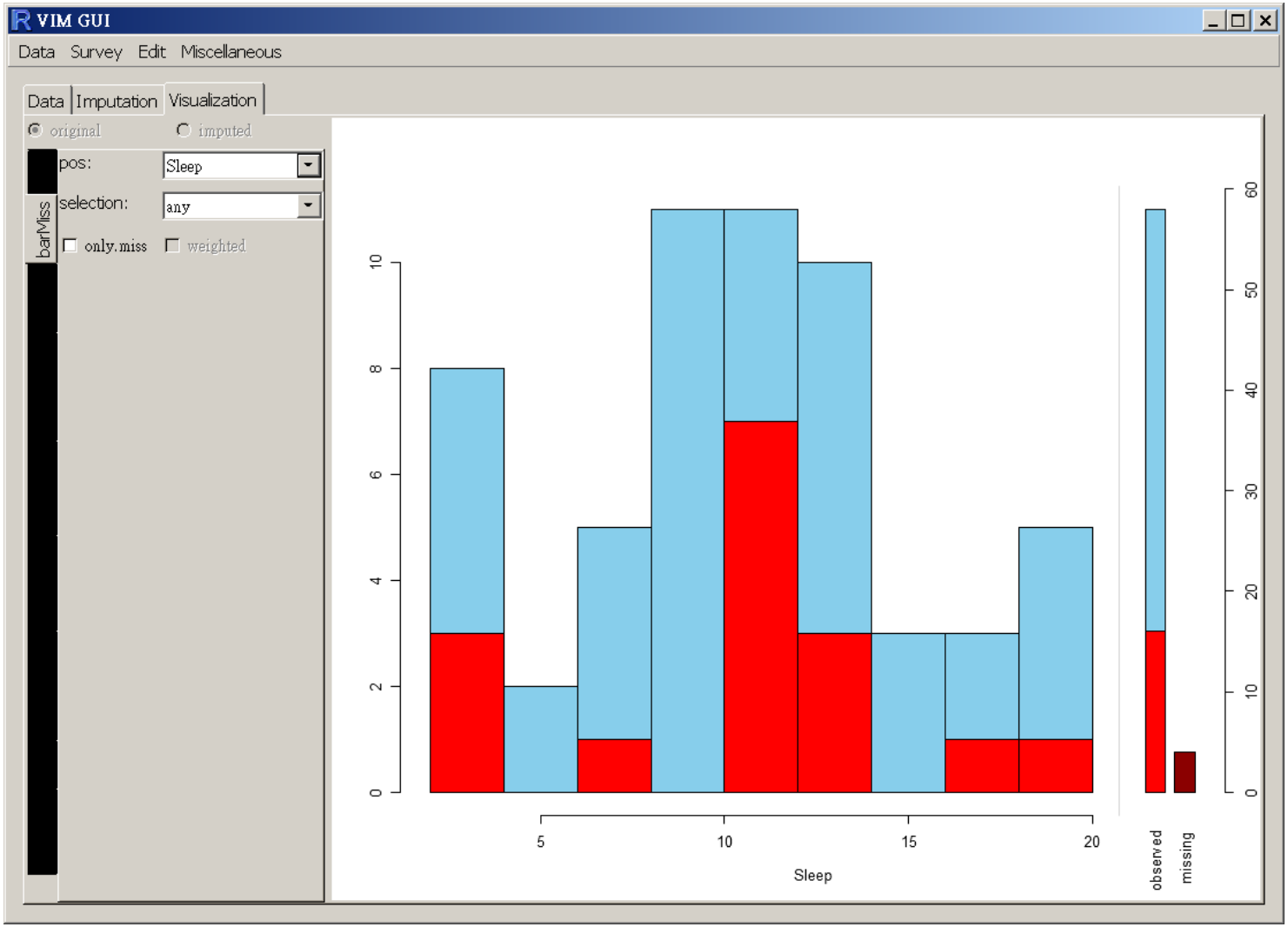
VIMGUI: Aggregation Plot



Study of dependence in missingness across variables.

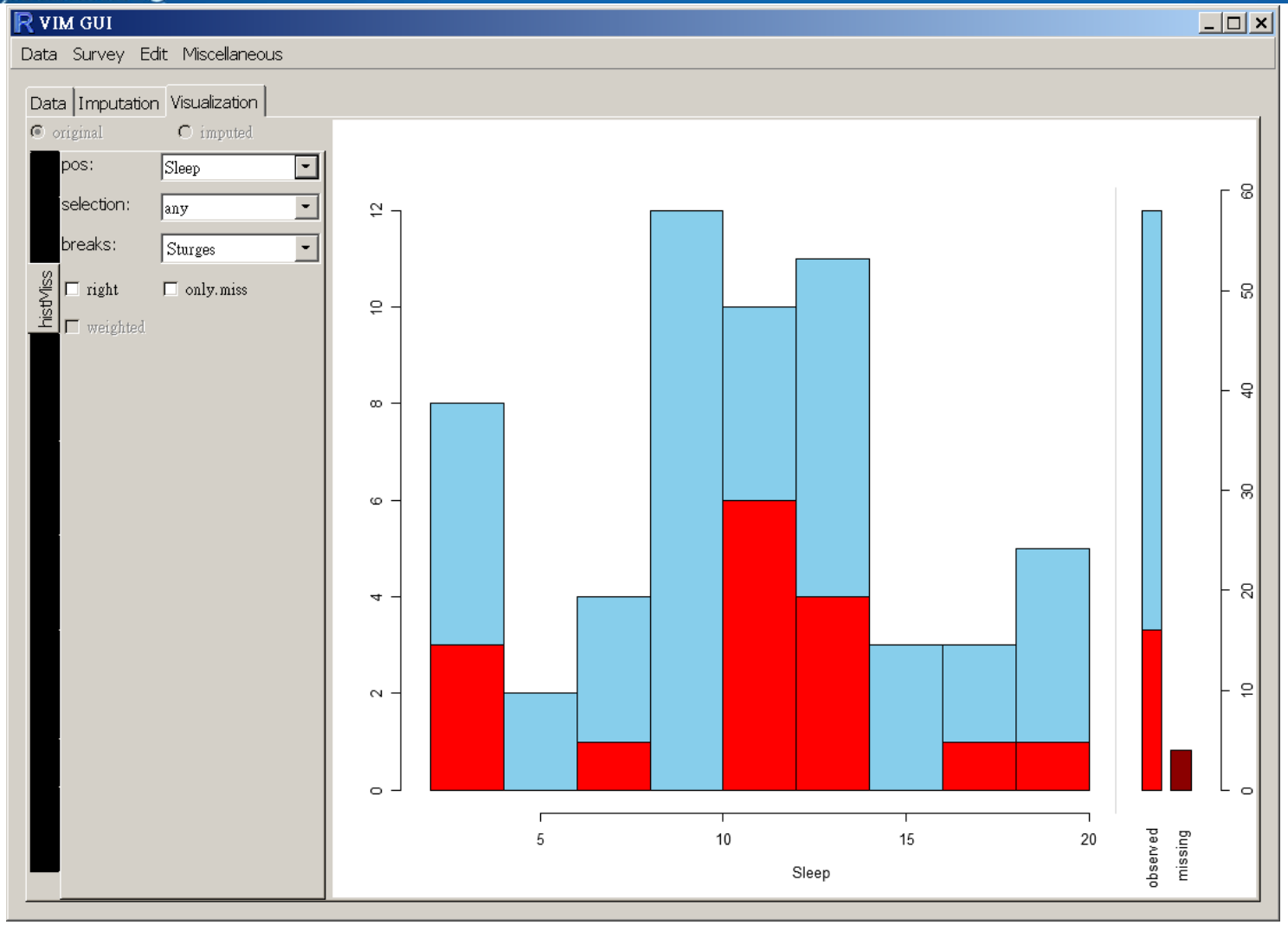
All existing combinations of missing (**red**) and non-missing (**light blue**) values in the observations. The frequencies of the combinations are visualized by small horizontal bars.

Barplot



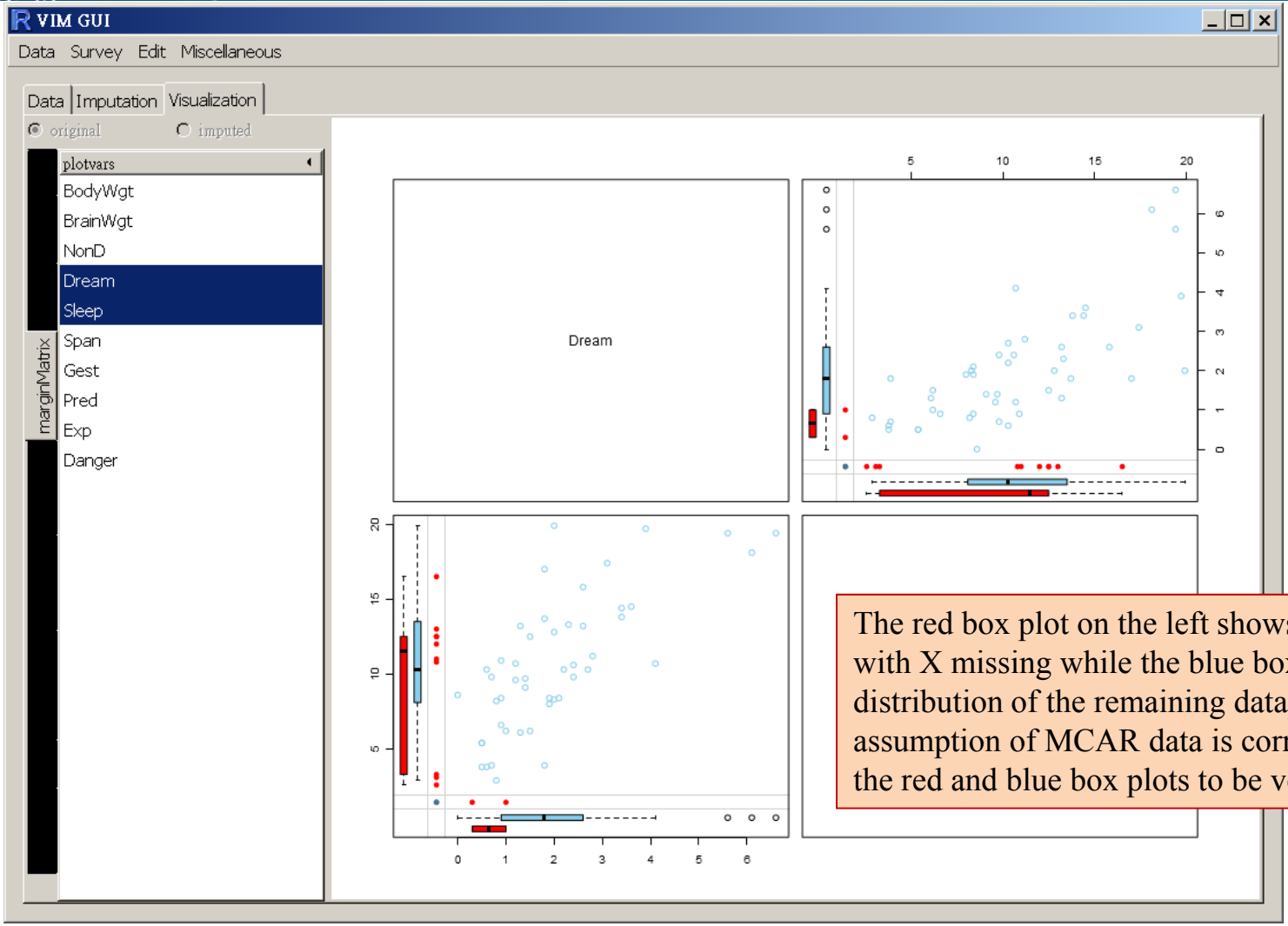
Wrong!

Histogram



The amount of missing values in other variables can be displayed by splitting each bin into two parts.

Marginplot Matrix

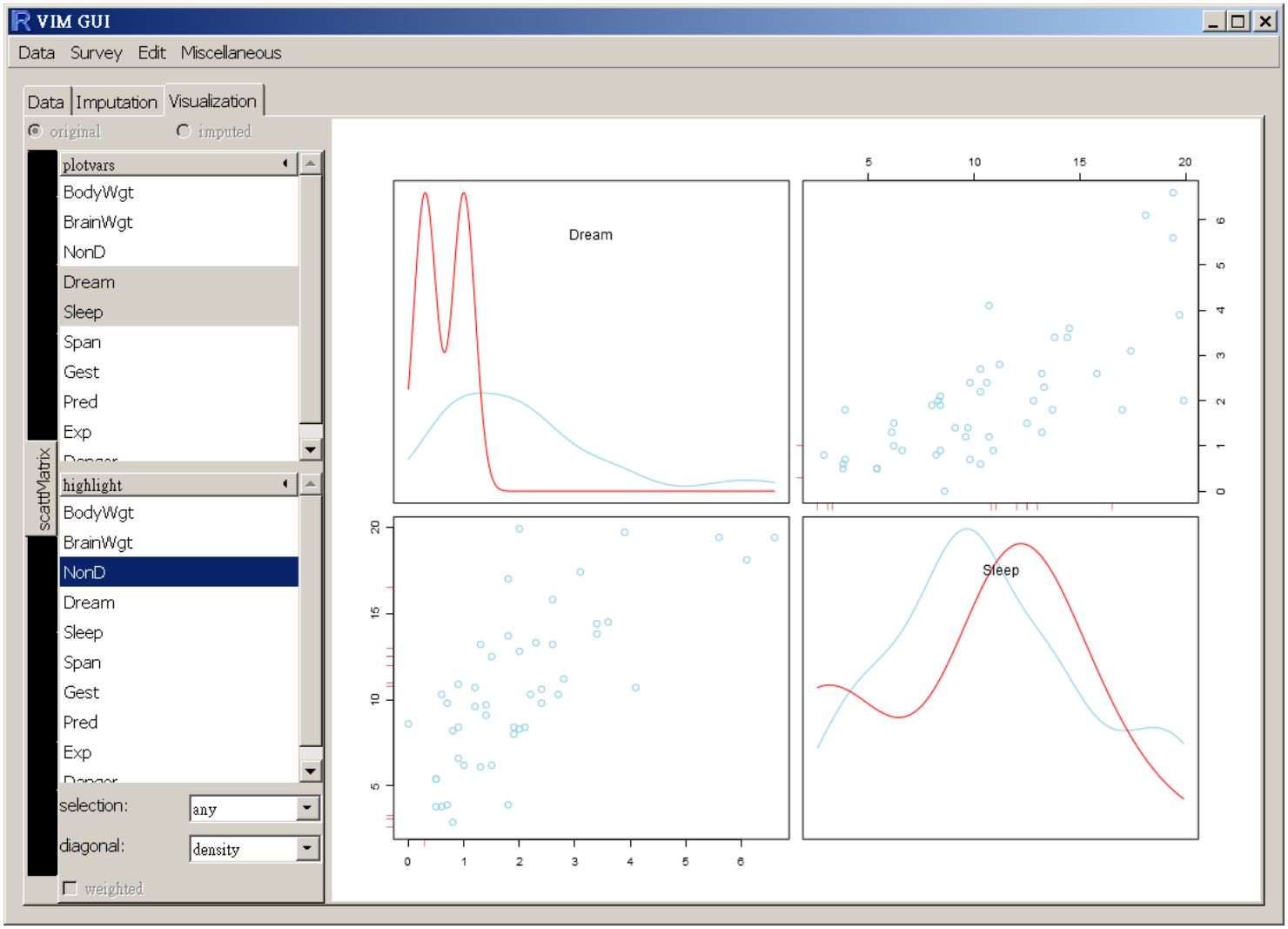


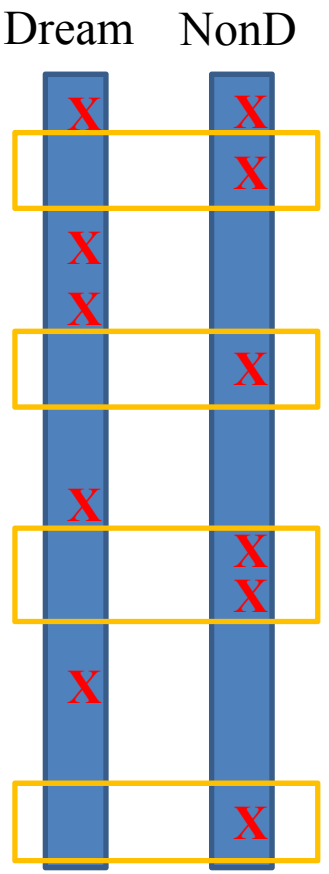
The observations with missing values in only one of the variables as univariate dot plots (sometimes also referred to as stripcharts or one-dimensional scatterplots) along the x- or y-axis.

The red box plot on the left shows the distribution of Y with X missing while the blue box plot shows the distribution of the remaining datapoints. If our assumption of MCAR data is correct, then we expect the red and blue box plots to be very similar.

The implementation in VIM also includes boxplots for available and missing data in the plot margins.
The frequencies of missing values in one or both variables are represented by numbers in the **lower left corner**.

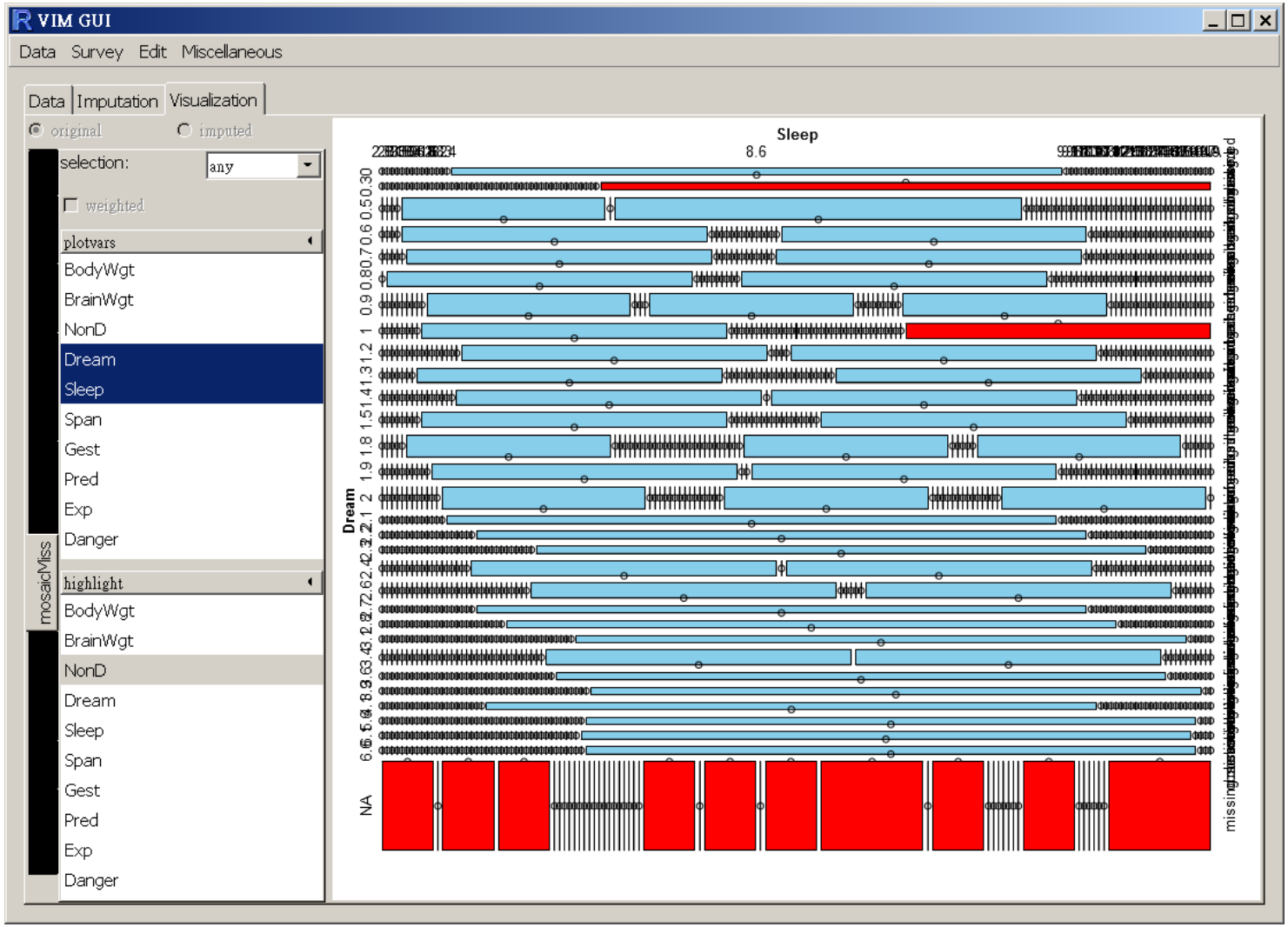
Scatterplot





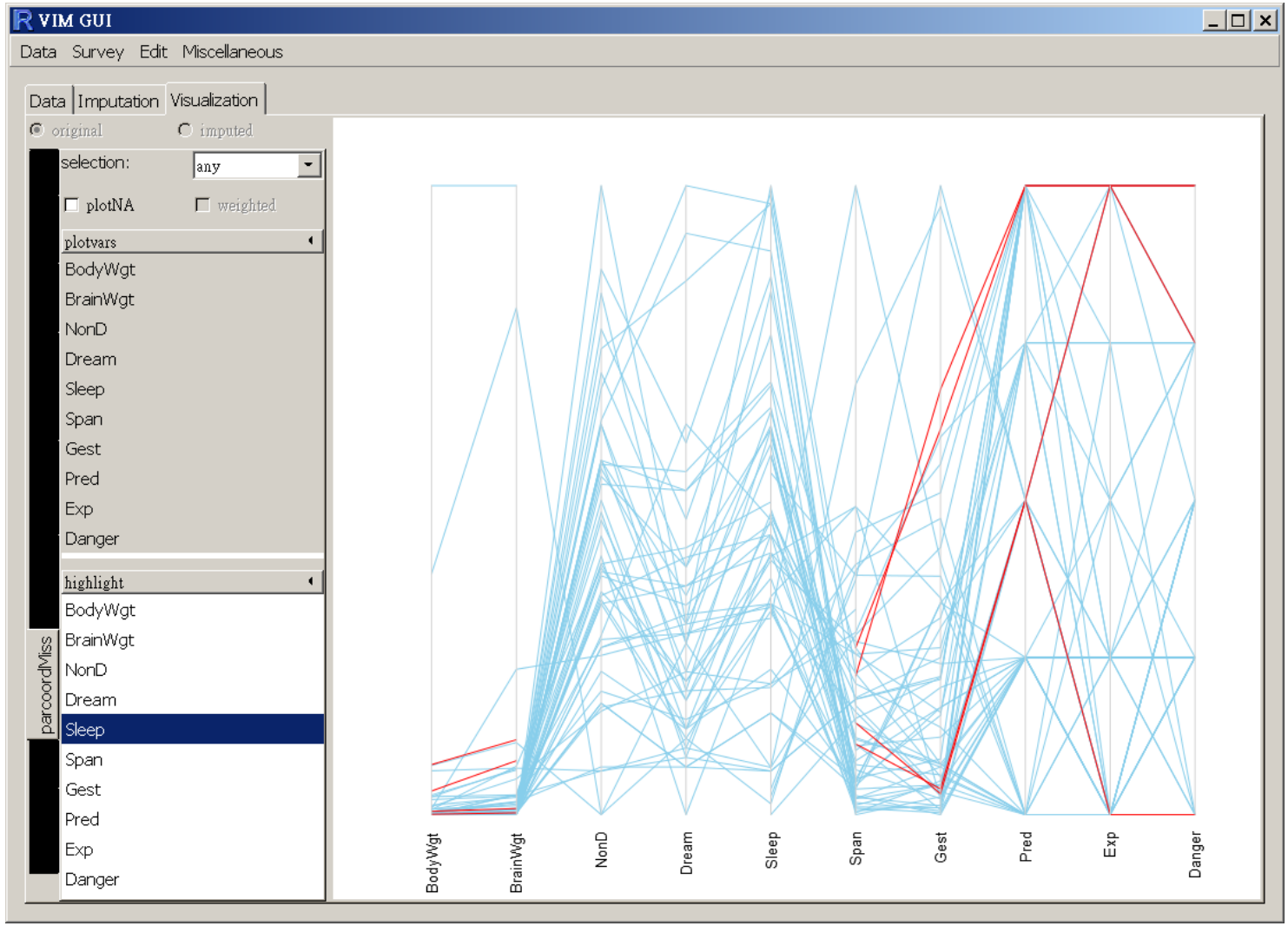
The observations with missing values in variable **NonD** are highlighted in the bivariate plots.

Mosaic Plot

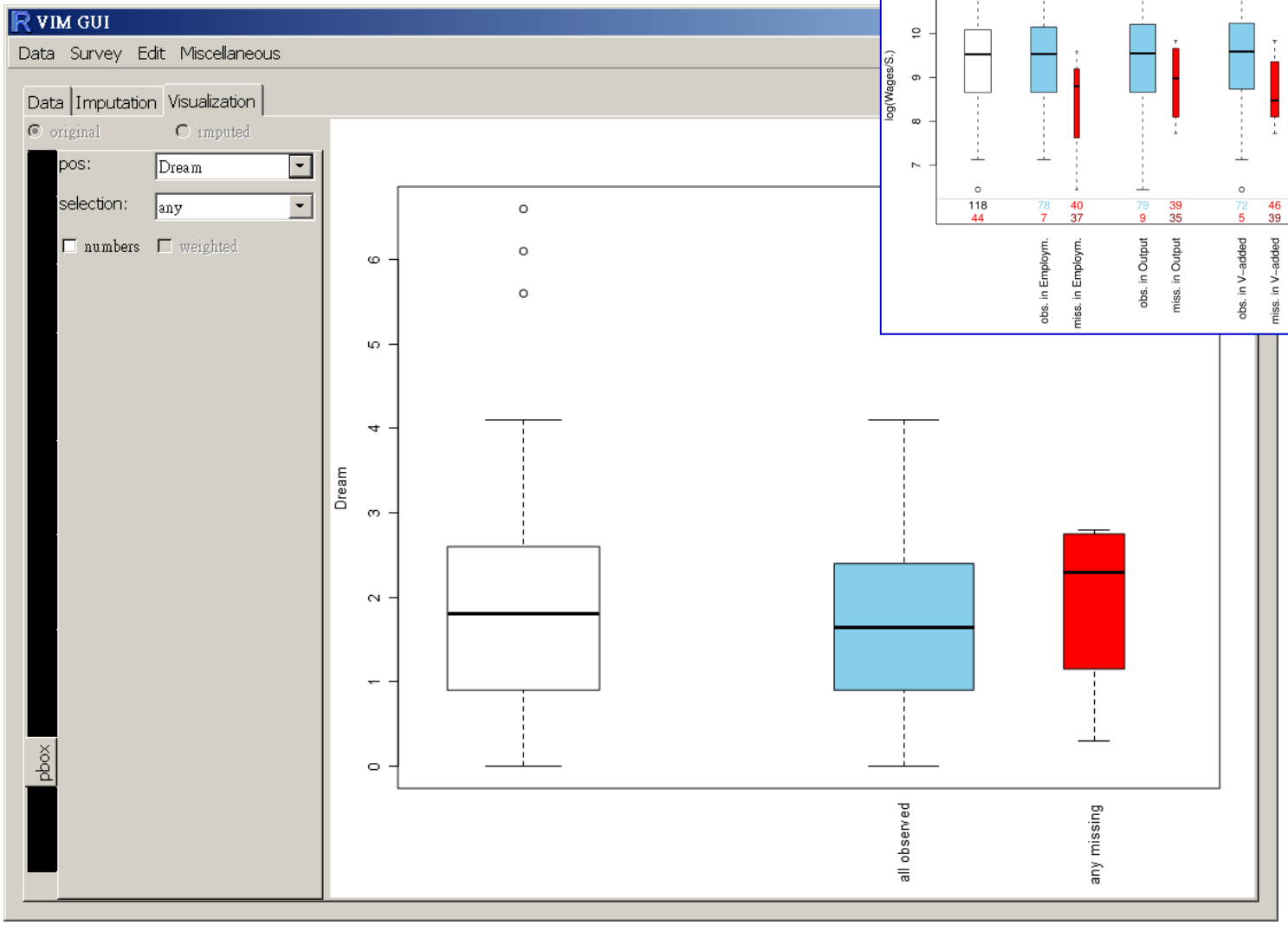


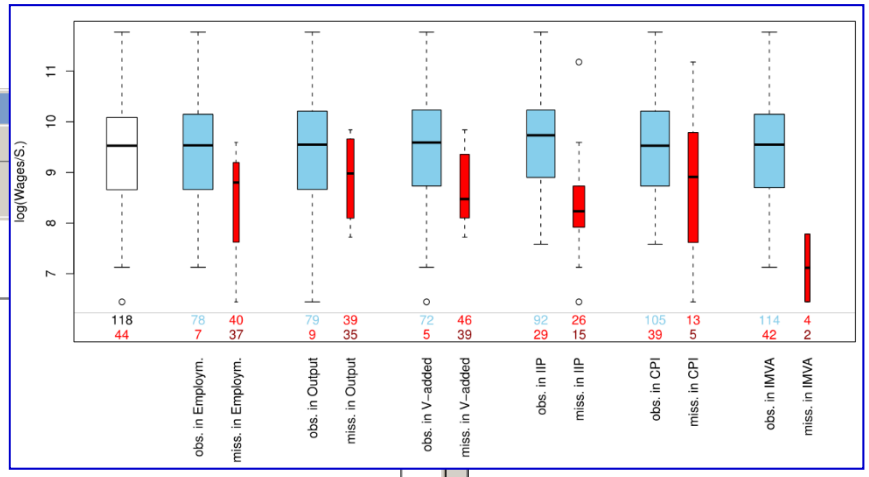
Google: Mosaic Plot: Visualization of "categorical data"

Parallel Coordinate Plot

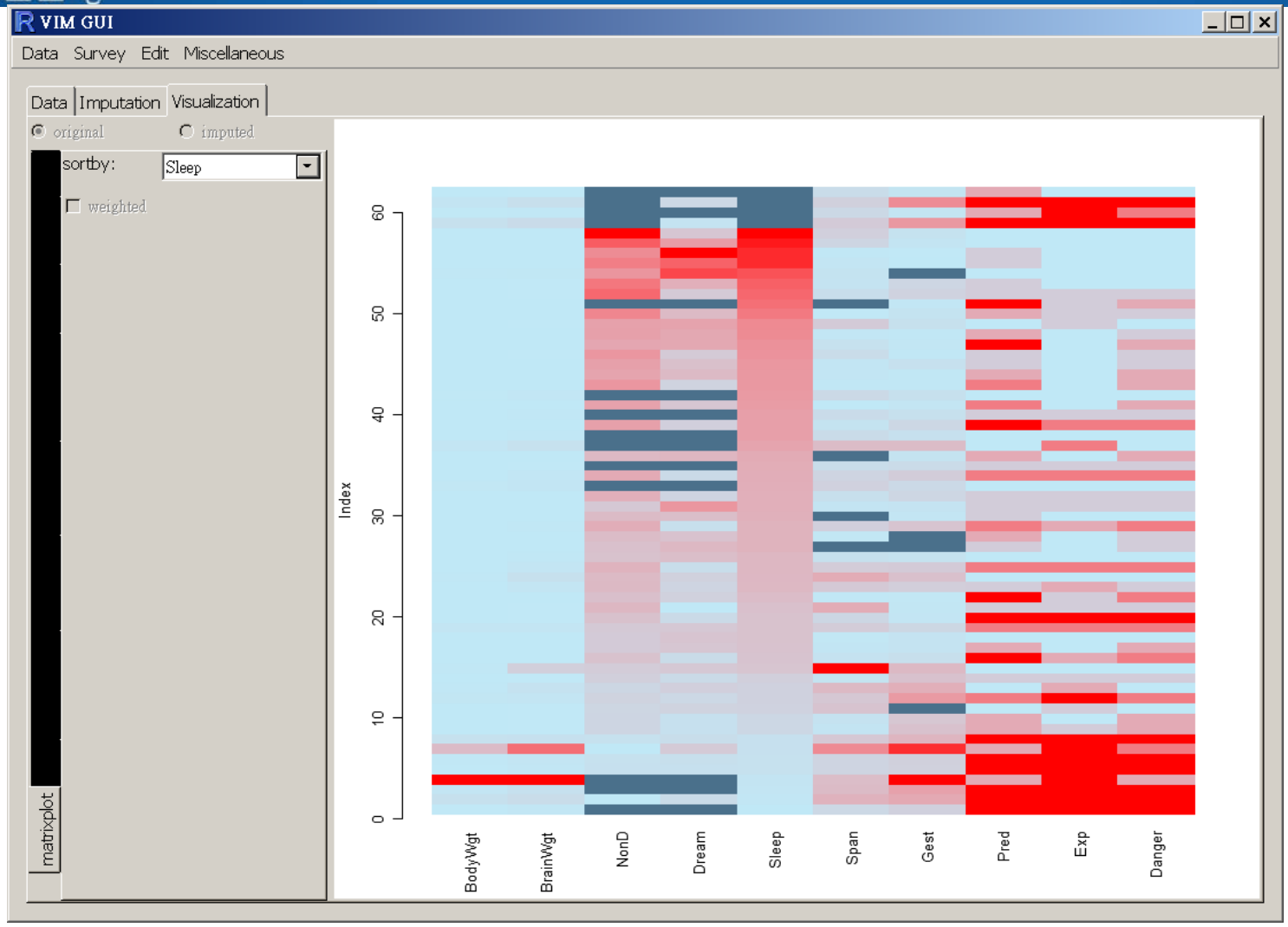


Parallel Boxplots





Matrix Plot



課堂練習: 利用**VIM**套件提供之指令，畫出以上各類圖形。

Other R Functions

```
missing.pattern.plot{mi}
missing.pattern.plot(data, y.order = FALSE, x.order = FALSE, clustered = TRUE,
                                    xlab = "Index", ylab = "Variable",
                                    main = NULL, gray.scale = FALSE,
                                    obs.col = "blue", mis.col = "red", ... )


# library(mi)
# data(CHAIN)
# missing.pattern.plot(CHAIN)
# missing.pattern.plot(scale(sleep))
```

```
ggmissing {ggplot2}
Missing values plot: Create a plot to illustrate patterns of missing values

ggmissing(data, avoid="stack", order=TRUE, missing.only = TRUE)
```

ps. 新版mi及ggplot2，已不提供!
See also: **aggr {MICE}**

Missingness Mechanism

- The presence of missing data can
  - effect the properties of the estimates
    (e.g. means, percentages, percentiles, variances, ratios, regression parameters, etc.).
  - affect inferences,
    (e.g., the properties of tests and confidence intervals.)

- **The missingness mechanism** (Little and Rubin, 1987)
  - The way in which the probability of an item missing depends on other observed or non-observed variables as well as on its own value.

- It helpful to classify missing values on the basis of the stochastic mechanism that produces them.

Missingness Mechanism

collected data

$$X = \{X_o, X_m\}$$

observed elements　　missing elements

The missingness indicator matrix R corresponds X,
and each element of R is 1 if the corresponding element of X is missing,
and 0 otherwise.

define the missingness mechanism as
the probability of R conditional on
the values of the observed and missing elements of X:

$$Pr(R|X_o, X_m)$$

(M1) Missing by Design

- **Excluded** some participants from the analysis because they are not part of the population under investigation.
  - Eg., valid skips: when a question is not answered because it is not applicable to the given unit.

- In many surveys different missingnes codes are applied indicating the reason why the respondent did not provide an answer:
  - (i) refused to answer; (ii) answered don't know; (iii) had a valid skip or (iv) was skipped by an enumerator error.
  - Depending on the code one can decide whether the corresponding values are to be imputed or not.

(M2) Missing Completely at Random

- **Missing Completely at Random (MCAR)**
  - missingness is independent of their own unobserved values and the observed data.
  - the pattern of missing values is totally random and does not depend on any variable, which may or may not be included in the analysis.

$$Pr(R|X) = Pr(R)$$

- *Example*: Miscoding or forgetting to log in answer
- For most data sets, the MCAR assumption is unlikely to be satisfied, one exception being the case when data are missing by design.
- For MCAR, **no bias** is introduced when omitting those observations with missing values (a lot of valuable information contained in these observations is then lost.)
- **Imputation methods** rely on the missingness being of the MCAR type.

(M3) Missing at Random (MAR)

- **MAR:** missingness does not depend on their unobserved value but does dependent on the observed data.

$$Pr(R|X) = Pr(R|X_o)$$

- ***Example***: male participants (observed data) are more likely to refuse to fill out the depression survey, but it does not depend on the level of their depression (unobserved value).
- ***Example***: if men are more likely to tell you their weight than women, **weight is MAR**.

- MAR can never be tested on any given data set because it can be that some unobserved variables are causing the missing pattern.
- MCAR is a special case of MAR, i.e. if the data are MCAR, they are also MAR.
- We can ignore missing data (= omit missing observations) if we have MAR or MCAR.

Ignorable

- The missing-data mechanism is said to be *ignorable* if the data are MAR and the parameters governing the missing-data mechanism are distinct from the parameters in the model to be estimated.

- Usually, MAR and "ignorability" are used interchangeably.

- If the missing-data mechanism is ignorable, then it is possible to obtain valid, optimal estimates of parameters without directly modeling the missing-data mechanism (Allison, 2001).

(M4) Missing Not at Random (MNAR)

- Missing Not at Random (MNAR)
  - The MAR assumption is violated. In general MNAR means that an unknown process is generating the missing values.
  - ***Example***: question about **income**, where the high rate of missing values (usually 20%~50%) is related to the value of the income itself (very high and very low values will not be answered).
- **MNAR data is a more serious issue.**
  - Check the data gathering process further and try to understand why the information is missing.
  - Eg., if most of the people in a survey did not answer a certain question, why did they do that? Was the question unclear?
- MNAR can appear in one of the two versions (or a combination):
  - Missingness that depends on unobserved predictors
  - Missingness that depends on the missing value itself.
- MNAR: the missing-data mechanism is **not ignorable**, and a valid estimation requires the missing-data mechanism to be modeled as part of the estimation process. The results can be very sensitive to the model choice (Little and Rubin, 1987).

Some Notes

- Assuming data is **MCAR**, too much missing data can be a problem.
  - Usually a safe maximum threshold is **5%** of the total for large datasets.
  - If missing data for a certain feature or sample is more than 5% then you probably should leave that feature or sample out.
- If some variable is missing almost **25%** of the datapoints.
  - Consider either dropping it from the analysis or gather more measurements.
  - Keep the other variables are below the **5%** threshold.
- For samples, missing just one feature (column) leads to a **25%** missing data per sample (row). Samples that are missing 2 or more features (**>50%**), should be dropped if possible.
- For categorical variables, replacing categorical variables is usually not advisable. Some common practice include replacing missing categorical variables with the mode of the observed ones (questionable).

Missing Values in R

- **NA**: a missing value ("not available"), **"NA"**: a string.
- **x[1]== NA** is not a valid logical expression and will not return **FALSE** as one would expect but will return **NA**.

```
> myvector <- c(10, 20, NA, 30, 40)
> myvector
[1] 10 20 NA 30 40
> mycountry <- c("Austria", "Australia", NA, NA, "Germany", "NA")
> mycountry
[1] "Austria"   "Australia" NA          NA          "Germany"   "NA"
> is.na(myvector)
[1] FALSE FALSE  TRUE FALSE FALSE
> which(is.na(myvector))
[1] 3
> x <- c(1, 4, 7, 10)
> x[4] <- NA # sets the 4th element to NA
> x
[1]  1  4  7 NA
> is.na(x) <- 1 # sets the first element to NA
> x
[1] NA  4  7 NA
```

```
#Recoding Values to Missing
mydata$v1[mydata$v1==99] <- NA
```

NOTE: **NULL** denotes something which never existed and cannot exist at all.

NA and Factor

- When a vector is used to create a factor by default, the missing value **NA** will be excluded from factor levels.
- In order to create a factor that includes missing values from a numeric variable, use **exclude = NULL**.
- The **table()** function will not create a factor level for **NA** which could be achieved by **exclude = NULL**.

```
> (x <- c("A", "B", NA, "C", "A", "A", "B", NA))
[1] "A" "B" NA  "C" "A" "A" "B" NA
> (x1 <- factor(x))
[1] A    B    <NA> C    A    A    B    <NA>
Levels: A B C
> is.na(x1)
[1] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE
> (table(x1))
x1
A B C
3 2 1
> (table(x1, exclude=NULL))
x1
   A    B    C <NA>
   3    2    1    2
```

```
> (x2 <- factor(x, exclude = NULL))
[1] A    B    <NA> C    A    A    B    <NA>
Levels: A B C <NA>
> is.na(x2)
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> (table(x2))
x2
   A    B    C <NA>
   3    2    1    2
> (table(x2, exclude=NULL))
x2
   A    B    C <NA>
   3    2    1    2
```

NA as a Level

```
> ## suppose you want "NA" as a level, and to allow missing values.
> is.na(x2)[2] <- TRUE
> x2
[1] A    <NA> <NA> C    A    A    B    <NA>
Levels: A B C <NA>
> is.na(x2)
[1] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
>
> airquality$Month
  [1] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6
 [41] 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
 [81] 7 7 7 7 7 7 7 7 7 7 7 7 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
[121] 8 8 8 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
> table(addNA(airquality$Month))

   5    6    7    8    9 <NA>
  31   30   31   31   30    0
> table(addNA(airquality$Month, ifany = TRUE))

 5  6  7  8  9
31 30 31 31 30
```

NA in Summary Functions

- Most of the statistical summary functions (**mean, var, sum, min, max**, etc.) accept an argument called **na.rm**, which can be set to **TRUE** if you want missing values to be removed before the summary is calculated. (default : **FALSE**)

- For functions that don't provide such an argument, the negation operator (**!**) can be used in an expression like **x[!is.na(x)]** to create a vector which contains only the nonmissing values in **x**.

```
> x <- c(1, 4, NA, 10)
>  summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
    1.0     2.5     4.0     5.0     7.0    10.0       1
>  mean(x)
[1] NA
>  sd(x)
[1] NA
>  mean(x, na.rm=TRUE)
[1] 5
>  sd(x, na.rm=TRUE)
[1] 4.582576
> x[!is.na(x)]
[1]  1  4 10
```

NA in Modeling Functions

- The statistical modeling functions (`lm, glm, gam`, etc.) all have an argument called `na.action=`, which allows you to specify a function that will be applied to the data frame specified by the `data=` argument before the modeling function processes the data.
  - `na.fail()` - issue an error if the object contains missing values.
  - `na.omit()` - exclude the missing values and return the rest of the object.
    (The `complete.cases` function may also be useful to achieve the same task)
  - `na.exclude()` - same as `na.omit()` but will result in different behavior of some functions (like `napredict()` and `naresid()`)
  - `na.pass()` - return also the missing values (the object remains unchanged)

NA in Modeling Functions

```
> mydata <- as.data.frame(matrix(sample(1:20, 8), ncol = 2))
> mydata[4, 2] <- NA
> names(mydata) <- c("y", "x")
> mydata
   y  x
1  1 19
2  6 12
3 10  2
4  4 NA
> lm(y~x, data = mydata)

Call:
lm(formula = y ~ x, data = mydata)

Coefficients:
(Intercept)            x
    11.3927      -0.5205

> lm(y~x, data = mydata, na.action = na.omit)

Call:
lm(formula = y ~ x, data = mydata, na.action = na.omit)

Coefficients:
(Intercept)            x
    11.3927      -0.5205

> lm(y~x, data = mydata, na.action = na.fail)
Error in na.fail.default(list(y = c(1L, 6L, 10L, 4L), x = c(19L, 12L,  :
  missing values in object
```

Other Special Values in R

- **NaN** : "not a number" which can arise for example when we try to compute the undeterminate 0/0.
- **Inf** which results from computations like 1/0.
- Using the functions **is.finite()** and **is.infinite()** we can determine whether a number is finite or not.

```
> x <- c(1, 0, 10)
> x/x
[1]   1 NaN   1
> is.nan(x/x)
[1] FALSE  TRUE FALSE
```

```
> 1/x
[1] 1.0 Inf 0.1
> is.finite(1/x)
[1]  TRUE FALSE  TRUE
>
> -10/x
[1]  -10 -Inf   -1
> is.infinite(-10/x)
[1] FALSE  TRUE FALSE
```

```
> exp(-Inf)
[1] 0
> 0/Inf
[1] 0
> Inf - Inf
[1] NaN
> Inf/Inf
[1] NaN
```

Traditional Approaches to Handling Missing Data

- If missing values do occur by chance among a set of replicates, the observed members of the set can stand in for the missing, albeit with some loss of statistical precision.
- Traditional Approaches to Handling Missing Data

(T1) List-wise deletion

(T2) Pairwise deletion

(T3) Non-response weighting

(T4) Mean substitution

(T5) Regression substitution.

(T6) Last value carried forward.

(T7) Using information from related observations.

(T8) Dummy variable adjustment

(T9) Deterministic imputation.

(T1) List-wise Deletion

- Also called the **complete case analysis**.
- All units with missing data for a variable are removed and the analysis is performed with the remaining units (complete cases).
- This is the default approach in most statistical packages.
- The use of this method is only justified if the missing data generation mechanism is **MCAR**.
- In R, using the function **`na.omit()`** or extract complete observations using the function **`complete.cases()`.**

```
> mdata <- matrix(rnorm(15), nrow=5)
> mdata[sample(1:15, 4)] <- NA
> mdata <- as.data.frame(mdata)
> mdata
           V1         V2          V3
1 -0.62222501  1.0807983          NA
2  0.07124865  0.5216675 -0.08334454
3  1.70707399  0.1004917  0.88197789
4          NA -0.6595201 -0.08387860
5          NA  1.6138847          NA
> (x1 <- na.omit(mdata))
          V1        V2          V3
2 0.07124865 0.5216675 -0.08334454
3 1.70707399 0.1004917  0.88197789
> (x2 <- mdata[complete.cases(mdata),])
          V1        V2          V3
2 0.07124865 0.5216675 -0.08334454
3 1.70707399 0.1004917  0.88197789
> mdata[!complete.cases(mdata),]
         V1         V2         V3
1 -0.622225  1.0807983         NA
4        NA -0.6595201 -0.0838786
5        NA  1.6138847         NA
```

快速分析一下，得知資料大概狀況

(T2) Pairwise Deletion

- To compute a covariance matrix, each two cases will be used for which the values of both corresponding variables are available. In R,
  - **use="everything"** (default): use all observations will result in a covariance matrix most likely consisting of NAs.
  - **use="all.obs"**: the presence of missing observations will produce an error.
  - **use="complete.obs"**: missing values are handled by list-wise deletion (and if there are no complete cases, an error appears).
  - **use="pairwise.complete.obs"**: the covariance between each pair of variables is computed using all complete pairs of observations on those variables.
- This can result in covariance or correlation matrices which are not positive semi-definite, as well as NA entries if there are no complete pairs for the given pair of variables.

```
> cov(mdata)
   V1        V2 V3
V1 NA        NA NA
V2 NA 0.7694197 NA
V3 NA        NA NA
> cov(mdata, use = "all.obs")
Error in cov(mdata, use = "all.obs") :
missing observations in cov/cor
> cov(mdata, use = "complete.obs")
           V1          V2         V3
V1  1.3379623 -0.34448500  0.7895494
V2 -0.3444850  0.08869452 -0.2032852
V3  0.7895494 -0.20328521  0.4659237
```

```
> cov(mdata, use = "na.or.complete")
           V1          V2         V3
V1  1.3379623 -0.34448500  0.7895494
V2 -0.3444850  0.08869452 -0.2032852
V3  0.7895494 -0.20328521  0.4659237
> cov(mdata, use = "pairwise")
           V1          V2         V3
V1  1.4304107 -0.56002326 0.78954945
V2 -0.5600233  0.76941970 0.05468712
V3  0.7895494  0.05468712 0.31078774
```

(T4) Mean Substitution

- A very simple but popular approach is to substitute means for the missing values.
- The method preserves sample size and does not reduce the statistical power associated with sample size in comparison with list-wise or pairwise deletion.
- This method produces biased estimates and can severely distort the distribution of the variable in which missing values are substituted.
- This results in underestimates of the standard deviations and **distorts relationships between variables** (estimates of the correlation are pulled toward zero).

Due to these **distributional problems**, it is often recommended to ignore missing values rather than impute values by mean substitution (Little and Rubin, 1989.)

```
mean.subst <- function(x) {
   x[is.na(x)] <- mean(x, na.rm = TRUE)
   x
}
```

```
> mdata
           V1         V2          V3
1 -0.62222501  1.0807983          NA
2  0.07124865  0.5216675 -0.08334454
3  1.70707399  0.1004917  0.88197789
4          NA -0.6595201 -0.08387860
5          NA  1.6138847          NA
> mdata.mip <- apply(mdata, 2, mean.subst)
> mdata.mip
               V1         V2          V3
[1,] -0.62222501  1.0807983  0.23825158
[2,]  0.07124865  0.5216675 -0.08334454
[3,]  1.70707399  0.1004917  0.88197789
[4,]  0.38536588 -0.6595201 -0.08387860
[5,]  0.38536588  1.6138847  0.23825158
```

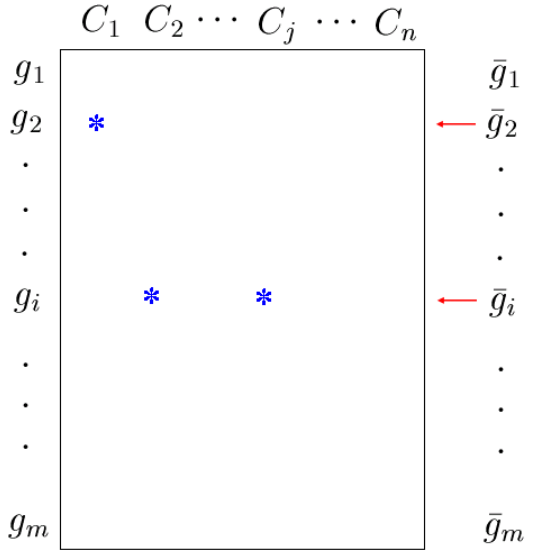
Mean Substitution for Microarray Data

- Missing log2 transformed data are replaced by *zeros* or by an *average* expression over the row ("row average").

- **Row average** assumes that the expression of a gene in one of the experiments is similar to its expression in a different experiment, which is often not true in microarray experiments.

- **Main weakness:**
  - do not model the connection of the missing values to the observed data.
  - do not consider the correlation structure of the data.
  - not very effective (Troyanskaya et al, 2001)

- **Useful**
  - where an initial imputation is required an iterative imputation method.



$$\widehat{C_j(g_i)} = \bar{g}_i$$

$i = 1, 2, \cdots, m.$

$j = 1, 2, \cdots, n.$

zero's
row average
row median

(T6) Last Value Carried Forward

- For **longitudinal data** (repeated measures are taken per subject) one could apply the Last Value Carried Forward (LVCF or LOCF) technique.
- The last observed value is used to fill in missing values in subsequent observations assuming that the most recent observation is the best guess for subsequent missing values (i.e. that the response remains constant for the last observed value).
- Unfortunately this assumption could be biased.
- R package `zoo` with function `na.locf()`: Generic function for replacing each NA with the most recent non-NA prior to it.

zoo {zoo}: Z's Ordered Observations

```
> bz <- zoo(c(2,NA,1,4,5,2))
> na.locf(bz)
1 2 3 4 5 6
2 2 1 4 5 2
> cz <- zoo(c(NA,9,NA,2,3,2))
> cz
 1  2  3  4  5  6
NA  9 NA  2  3  2
> na.locf(cz)
2 3 4 5 6
9 9 2 3 2
> z <- zoo(c(0.073, 0.590, 0.810, 0.078, 0.475),
+      as.Date(c("2016-01-13", "2016-01-09", "2016-01-16",
+                   "2016-01-23", "2016-01-18")))
> z
2016-01-09 2016-01-13 2016-01-16 2016-01-18 2016-01-23
     0.590      0.073      0.810      0.475      0.078
> g <- seq(start(z), end(z), "day")
> g
 [1] "2016-01-09" "2016-01-10" "2016-01-11" "2016-01-12" "2016-01-13" "2016-01-14"
 [7] "2016-01-15" "2016-01-16" "2016-01-17" "2016-01-18" "2016-01-19" "2016-01-20"
[13] "2016-01-21" "2016-01-22" "2016-01-23"
> na.locf(z, xout = g)
2016-01-09 2016-01-10 2016-01-11 2016-01-12 2016-01-13 2016-01-14 2016-01-15
     0.590      0.590      0.590      0.590      0.073      0.073      0.073
2016-01-16 2016-01-17 2016-01-18 2016-01-19 2016-01-20 2016-01-21 2016-01-22
     0.810      0.810      0.475      0.475      0.475      0.475      0.475
2016-01-23
     0.078
```

(T7-T9) Traditional Approaches

- (T7) Using Information from Related Observations
  - In official Statistics, the imputation of missing values with a donor from the underlying data (hot-deck imputation) or with a donor from external data (cold-deck imputation) is still popular.
  - `hotdec()` from the R package VIM.

R.R. Andridge and R.J.A. Little, A Review of Hot Deck Imputation for Survey Non-response. Int Stat Rev. 2010 Apr; 78(1): 40–64.

- (T8) Dummy Variable Adjustment
  - leads to biases in the estimated regression parameters and the standard errors, even if the data are MCAR, thus making it unacceptable (Jones, 1996).

- (T9) Deterministic Imputation
  - The deterministic imputation identifies cases in which there is only one possible solution (based on logical rules) and thus allows the record to satisfy the rules.

Advanced Imputation Methods

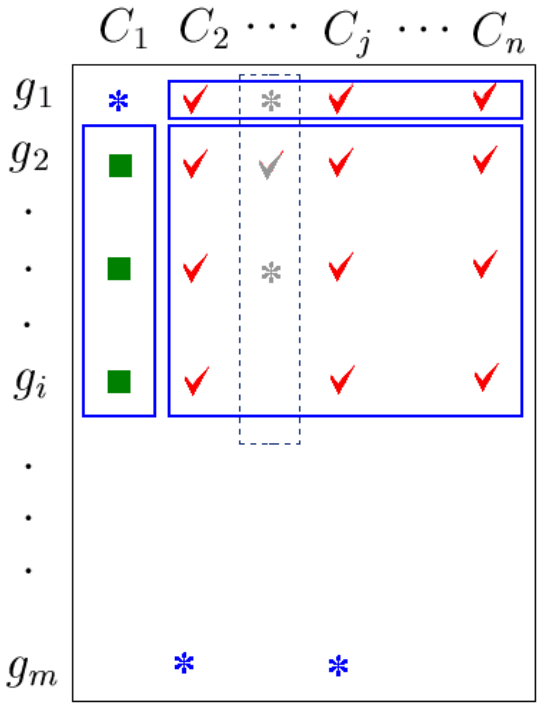
- Univariate methods: column-wise (conditional) mean imputation.

- Multivariate methods: using the linear dependencies between variables.
  - **data-ordering and distance-based** imputation methods such as hot-deck methods and k-nearest neighbour imputation.
  - **covariance-based methods** such as the approaches by Verboven et al (2007) or Serneels and Verdonck (2008), and
  - **model-based methods** approaches such as regression imputation (Raghunathan et al, 2001; Templ et al, 2010) or depth-based imputation (B´ eguin and Hulliger, 2004).

- The assumption of elliptical distributions is necessary for all covariance-based methods, but not for depth-based ones.

(A1) K-Nearest Neighbour Imputation

- The k-nearest neighbour imputation searches for the k-nearest observations (respective to the observation which has to be imputed) and replaces the missing value with the mean of the found *k* observations.
- It is recommended to use the (weighted) median instead of the arithmetic mean.
- **KNN** minimize data modeling assumptions and take advantage of the correlation structure of the data.



KNNimpute

Model:

$$\{g_{(k)}, k = 1, 2, \cdots, K\} = \underset{k}{\text{args}} \max_{i \in C} \text{Corr}(g_1, g_i)$$

$$\{g_{(k)}, k = 1, 2, \cdots, K\} = \underset{k}{\text{args}} \min_{i \in C} \text{Dist}(g_1, g_i)$$

C: Observed C_i's without missing values

Imputation:

$$\text{Average} \quad \widehat{C_1(g_1)} = \frac{1}{K} \sum_{k=1}^{K} C_1(g_k)$$

$$\text{Weighted Average} \quad \widehat{C_1(g_1)} = \frac{\sum_{k=1}^{K} w_k C_1(g_k)}{\sum_{k=1}^{K} w_k}$$

$$w_k = \frac{1}{\sum_{j \in C} [C_j(g_k) - C_1(g_1)]^2}$$

KNN Imputation for Microarray Data

- Results are adequate and relatively insensitive to values of k between 10 and 20. (Troyanskaya et al, 2001)
- Euclidean distance appeared to be a sufficiently accurate norm.
- Euclidean distance measure is often sensitive to outliers, which could be present in microarray data.
- Log-transformed data seems to sufficiently reduce the effect of outliers on genes similarity determination.

knnImputation {DMwR}, ce.impute {dprep}

```
> pMiss <- function(x){sum(is.na(x))/length(x)*100}
>
> # DMwR: Functions and data for "Data Mining with R"
> # knnImputation(data, k = 10, scale = T, meth = "weighAvg", distData = NULL)
> library(DMwR)
> data(algae) # head(algae)
> summary(algae)
    season       size       speed          mxPH            mnO2                Cl
 autumn:40   large :45   high  :84   Min.   :5.600   Min.   : 1.500   Min.   :  0.222
...
 Max.   :44.400   Max.   :77.600   Max.   :31.600
>
> colid <- which(unlist(lapply(algae, is.numeric)))
> apply(algae[, colid], 2, pMiss)
mxPH mnO2   Cl  NO3  NH4 oPO4  PO4 Chla   a1   a2   a3   a4   a5   a6   a7
 0.5  1.0  5.0  1.0  1.0  1.0  1.0  6.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
> algae.comp <- knnImputation(algae)
```

```
> # dprep: Data Pre-Processing and Visualization Functions for Classification
> # ce.impute {dprep}: Imputation in supervised classification
> library(dprep)
> # ce.impute(data, method = c("mean", "median", "knn"), atr, nomatr = rep(0, 0), k1 = 10)
> data(hepatitis)
> summary(hepatitis)
       V1              V2              V3              V4              V5
 Min.   :1.000   Min.   : 7.0   Min.   :1.000   Min.   :1.000   Min.   :1.000
...
> hepa.imputed <- ce.impute(hepatitis, "knn", k1=10)
```

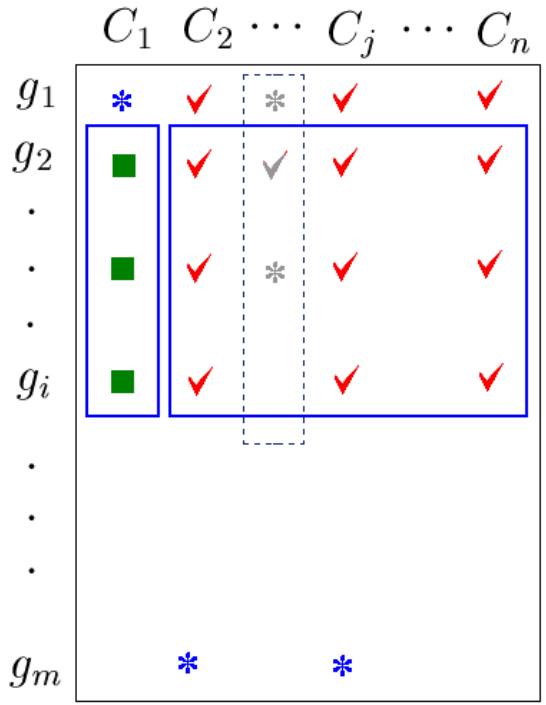
impute.knn{impute}

```
> source("https://bioconductor.org/biocLite.R")
> biocLite(c("impute", "pcaMethods", "Biobase"))
```

```
> library(impute)
> data(khanmiss); dim(khanmiss)
[1] 2309   65
> khanmiss[1:4, 1:5]
     X                        X1      sample1      sample2      sample3
1                                         EWS          EWS          EWS
2 GENE1  "\\"catenin (cadherin-a"   0.773343723 -0.078177781 -0.084469157
3 GENE2   "farnesyl-diphosphate"   -2.438404816 -2.415753791 -1.649739209
4 GENE3  "\\"phosphofructokinase"  -0.482562158  0.412771683 -0.241307522
> khan.expr <- khanmiss[-1, -(1:2)]
> dim(khan.expr)
[1] 2308   63
> khan.expr[1:4, 1:5]
       sample1      sample2      sample3      sample4     sample5
2  0.773343723 -0.078177781 -0.084469157  0.965614087 0.075663904
..
5 -2.721135441 -2.825145973  -2.87528612 -1.741256487  0.27269533
> if(exists(".Random.seed")) rm(.Random.seed) #  First example
> khan.imputed <- impute.knn(as.matrix(khan.expr))
> sum(as.integer(is.na(khan.expr)))
[1] 1282
> sum(as.integer(!is.na(khan.expr)))
[1] 144122
> sum(as.integer(is.na(khan.imputed)))
[1] 0
> sum(as.integer(!is.na(khan.imputed)))
[1] 145404
> attr(khan.imputed, "rng.seed") # should be 362436069
> attr(khan.imputed, "rng.state") # should be NULL
> set.seed(12345) #  Second example
> khan.imputed <- impute.knn(as.matrix(khan.expr))
> save(khan.imputed, file="khanimputation.Rda")
```

(A2) Regression Methods

- Using fitted regression values to replace missing values.
- The model must be chosen so that it does not yields invalid fitted values.
  e.g., negative values.
- This technique might be more accurate than simply substituting a measure of central tendency, since the imputed value is based on other input variables.
- This technique underestimates standard errors by underestimating the variance in x.



Regression

Model:

$$C_1 = \beta_0 + \sum_{j \in C} \beta_j C_j$$

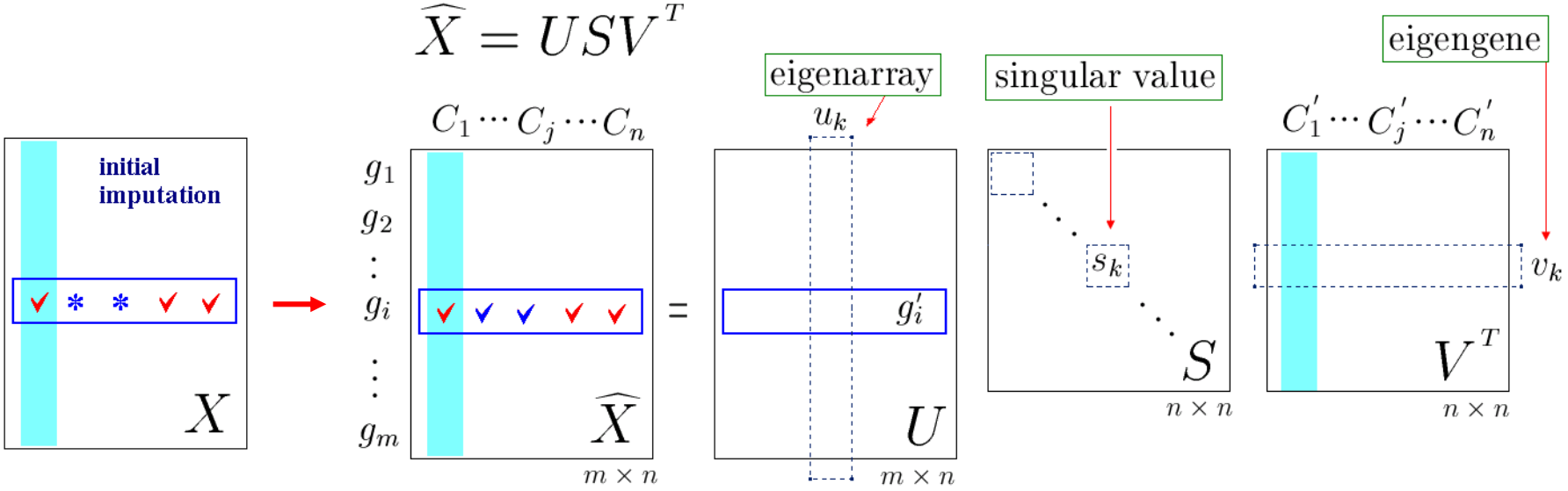
C: Observed C_i's without missing values

Imputation:

$$\widehat{C_1(g_1)} = \hat{\beta}_0 + \sum_{j \in C} \hat{\beta}_j C_j(g_1)$$

(A3) Singular Value Decomposition Imputation

$$\widehat{X} = USV^T$$



SVDimpute

Model:

$$g_i(\mathrm{C}) = \beta_0 + \sum_{k=1}^{K} \beta_k v_{(k)}(\mathrm{C})$$

C: Observed C_i's without missing values

Imputation:

$$g_i(\widehat{\sim \mathrm{C}}) = \hat{\beta}_0 + \sum_{k=1}^{K} \hat{\beta}_k v_{(k)}(\sim \mathrm{C})$$

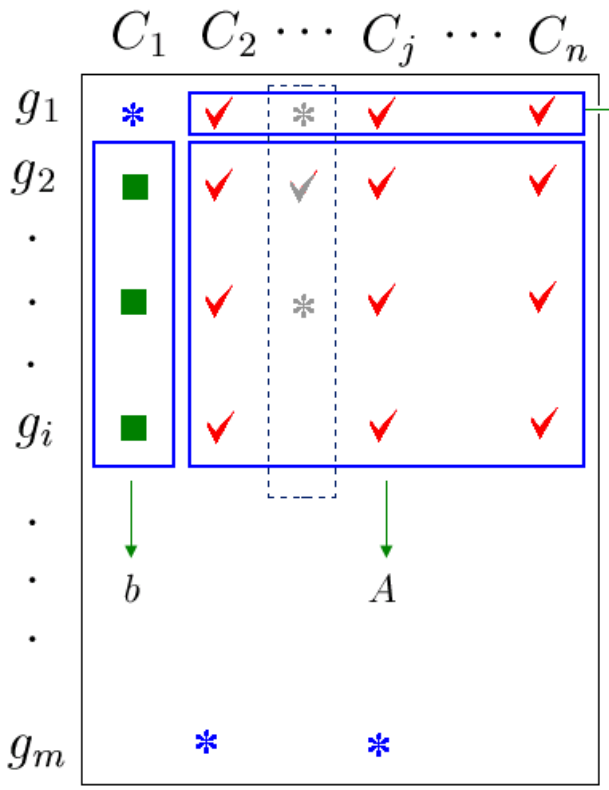
Could Extend to Iterative approach.

- Troyanskaya O, Cantor M, Sherlock G, Brown P, Hastie T, Tibshirani R, Botstein D, Altman RB. (2001), Missing value estimation methods for DNA microarrays. Bioinformatics 17(6), 520-525.
- Trevor Hastie , Robert Tibshirani, Gavin Sherlock , Michael Eisen , Patrick Brown , David Botstein. (1999). Imputing Missing Data for Gene Expression Arrays, Technical Report.

範例: svdImpute{pcaMethods}

```
> library(pcaMethods)
> library(Biobase)

> data(sample.exprSet.1)
> exSet <- sample.exprSet.1
> strc(exSet)

> exSetNa <- exSet
> dim(exSetNa@exprs)
[1] 500  26
> exSetNa@exprs[sample(500*26, 200)] <- NA
> lost <- is.na(exSetNa@exprs)
> sum(lost)
>
> #  PPCA Impute
> impExSet <- asExprSet(pca(exSetNa, nPcs = 2, method = "ppca"), exSetNa)
>
> #  or SVD Impute
>impExSet <- asExprSet(pca(exSetNa, nPcs = 5, method = "svdImpute"), exSetNa)

> index <- sum((exSet@exprs[lost]-impExSet@exprs[lost])^2)/sum(exSet@exprs[lost]^2)
> index
```

(A4) Local Least Square Imputation



LLSimpute

$$\{g_{(k)}, k = 1, 2, \cdots, K\} = \underset{k}{\text{args}} \max_{i \in \text{C}} \text{Corr}(g_1, g_i)$$

$$\{g_{(k)}, k = 1, 2, \cdots, K\} = \underset{k}{\text{args}} \min_{i \in \text{C}} \text{Dist}(g_1, g_i)$$

C: Observed C_i's without missing values

$$v = \arg\min_x \|A'x - w\|_2$$

Imputation:

$$\widehat{C_1(g_1)} = b'v$$

- Bo TH, Dysvik B, Jonassen I. LSimpute: accurate estimation of missing values in microarray data with least squares methods. Nucleic Acids Res. 2004 Feb 20;32(3):e34.
- Hyunsoo Kimy, Gene H. Golubz, and Haesun Parky. (2004). Missing Value Estimation for DNA Microarray Gene Expression Data: Local Least Squares Imputation, Bioinformatics Advance Access published August 27, 2004.

Methods Related to Maximum likelihood and Multiple Imputation

- Maximum likelihood (ML) and multiple imputation (MI) are currently considered the "state of the art" and are the recommended missing data techniques in the methodological literature.

- Advantages:
  - Have a strong **theoretical framework**.
  - Supported by a large number of empirical studies in **different domains**.
  - The popularity of these methods has risen recently due to the available implementations in a variety of both commercial and free statistical **software programs**.

(A5) Expectation Maximisation (EM)

- EM imputation method (Dempster et al, 1977) assumes that the underlying model for the observed data is Gaussian.
- This method is able to deal with MCAR and MAR missing values mechanism.
- EM algorithm starts with some initial values for the mean and the covariance matrix and iterates through imputing missing values (imputation step) and re-estimating the mean and the covariance matrix from the complete data set (estimation step).

```
> library(norm)
> data(mdata)
> tail(mdata, 4)
   ageh agew edu   inc kid
22   34   36  12 85000   1
23   40   35  16    NA  NA
24   38   38  18 95000   2
25   41   37  12    NA   2
> s <- prelim.norm(mdata) #do preliminary manipulations
> thetahat <- em.norm(s, showits=FALSE) #find the mle
> rngseed(1234567) #set random number generator seed
> ximp <- imp.norm(s, thetahat, mdata) #impute missing data under the MLE
> tail(ximp, 4)
   ageh agew edu      inc      kid
22   34   36  12 85000.00 1.000000
23   40   35  16 37936.22 2.680135
24   38   38  18 95000.00 2.000000
25   41   37  12 59358.20 2.000000
```

NOTE: **MCMC algorithm** can be applied to models other than the multivariate normal model.

(A6) Predictive Mean Matching (PMM)

- **Predictive Mean Matching (PMM)**: for each observation in a variable with missing value, we find observation (from available values) with the closest **predictive mean** to that variable. The observed value from this "match" is then used as imputed value.

- PMM imputes missing values by means of the nearest-neighbor donor with distance based on the expected values of the missing variables conditional on the observed covariates.

- The PMM method ensures that imputed values are plausible; it might be more appropriate than the regression method (which assumes a joint multivariate normal distribution) if the normality assumption is violated (Horton and Lipsitz 2001, p. 246).

- PMM works well for continuous and categorical (binary & multi-level) without the need for computing residuals and maximum likelihood fit.

PMM Algorithm

Let $Y=(Y_{\text{obs}}, Y_{\text{mis}})$ be an incomplete semicontinuous variable with n sample units, where Y_{obs} and Y_{mis} denote the observed values and the missing values in Y, respectively.

Further, $X=(X_1, \ldots, X_j)$ is a set of j fully observed covariates, where X_{obs} and X_{mis} correspond to the observed missing parts in Y.

$$Y_i = X_i^T \beta + \epsilon_i,$$

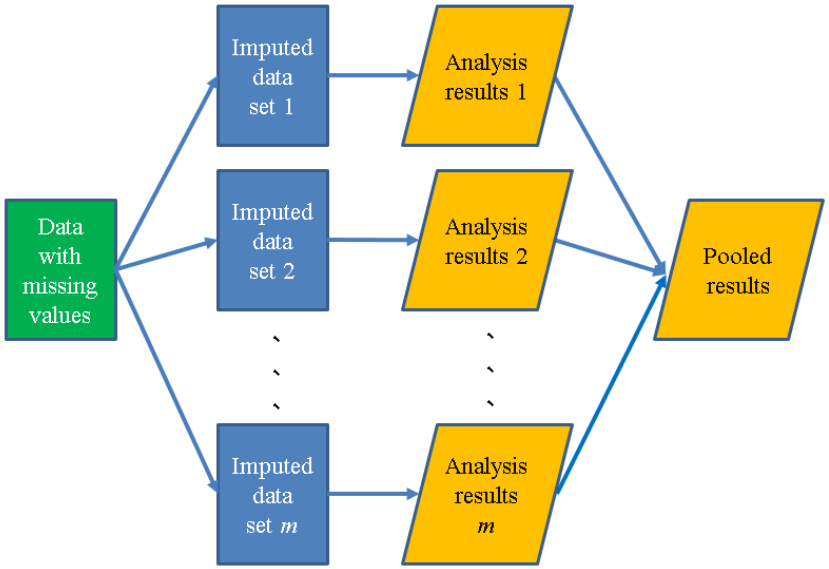
Multiply imputing Y_{mis} by means of PMM is performed by the following algorithm:

1. Use linear regression of Y_{obs} given X_{obs} to estimate $\hat{\beta}$, $\hat{\sigma}$, and $\hat{\varepsilon}$ by means of ordinary least squares.
2. Draw σ^{2*} as $\sigma^{2*} = \hat{\varepsilon}^T \hat{\varepsilon}/A$, where A is a χ^2 variate with $n_{\text{obs}} - r$ degrees of freedom.
3. Draw β^* from a multivariate normal distribution centered at $\hat{\beta}$ with covariance matrix $\sigma^{2*}\left(X_{\text{obs}}^T X_{\text{obs}}\right)^{-1}$.
4. Calculate $\hat{Y}_{\text{obs}} = X_{\text{obs}}\,\hat{\beta}$ and $\hat{Y}_{\text{mis}} = X_{\text{mis}}\,\beta^*$.
5. For each $\hat{Y}_{\text{mis},i}$, find $\Delta = |\hat{Y}_{\text{obs}} - \hat{Y}_{\text{mis},i}|$.
6. Randomly sample one value from $(\Delta^{(1)}, \Delta^{(2)}, \Delta^{(3)})$, where $\Delta^{(1)}$, $\Delta^{(2)}$, and $\Delta^{(3)}$ are the three smallest elements in Δ, respectively, and take the corresponding $Y_{\text{obs},i}$ as the imputation.
7. Repeat steps 1–6 m times, each time saving the completed dataset.

Gerko Vink, Laurence E. Frank, Jeroen Pannekoek and Stef van Buuren, 2014, Predictive mean matching imputation of semicontinuous variables, Statistica Neerlandica, 68(1), 61–90.

(A7) Multiple Imputation

- Multiple imputation requires three steps
  - **Imputation**: impute the missing entries of the incomplete data sets *m* times. Imputed values are drawn for a distribution (that can be different for each missing entry). This step results is *m* complete data sets.
  - **Analysis**: Analyze each of the *m* completed data sets. This step results in *m* analyses.
  - **Pooling**: Integrate the *m* analysis results into a final result.



- Rubin (1987) has shown that if the method to create imputations is inferences will be statistically valid.

Multiple Imputation Online:
www.multiple-imputation.com

Rubin, D.B. (1987), Multiple Imputation for Nonresponse in Surveys, New York: John Wiley & Sons, Inc.
Little, R.J.A. and Rubin, D.B. (1987), Statistical Analysis with Missing Data, New York: John Wiley & Sons, Inc.

Comparison

表 8.1　遺漏資料的插補技術比較

插補方法	優點	缺點	最佳使用時機
只利用有效資料的插補			
完整個案分析	• 最容易執行 • 許多統計軟體的預設方法	• 最容易受到非隨機過程的影響 • 樣本數的損失最多 • 較低的統計檢定力	• 較大的樣本數 • 變數之間有較強的關係 • 資料的遺漏程度較低
所有可用資料分析	• 有效資料的最大利用 • 在不替代數值的之下儘可能地將樣本數極大化	• 每一個變數插補的樣本數不一樣 • 在相關和特徵值的計算可能產生「超出範圍」的數值	• 資料的遺漏程度相對較低 • 變數之間是中等相關
利用已知的替代值插補			
個案替代	• 提供真實的替代數值而不是計算得到的數值（例如另一個實際的觀察值）	• 必須有不在原始樣本內的其他個案 • 必須定義相似性的測量，以找到適當的替代個案	• 其他的個案可以取得 • 能夠確認適當的替代個案
熱卡 / 冷卡插補	• 從最相似的個案或最佳的已知數值取得實際數值來替代遺漏資料	• 必須定適合的相似個案或適當的外部數值	• 確定替代的數值是已知的，或在相似性的基礎上，透過遺漏資料的處理找出適當的變數

劉正山, 莊文忠, 2012, 項目無反應資料的多重插補分析, 第八章, 臺灣選舉與民主化調查(TEDS)方法論之回顧與前瞻 (黃紀 主編。) pp. 276-305.

Comparison

MLE, EM

插補方法	優點	缺點	最佳使用時機
隨機性遺漏資料處理的插補			
模型基礎法	• 能處理非隨機和隨機的遺漏資料過程 • 是有最小偏差之數值的原始分布的最佳代表	• 研究者才能詳細說明的複雜模型 • 需要專業的軟體 • 一般不是可以直接由軟體程式中取得（SPSS 的 EM 方法除外）	• 可以解決非隨機遺漏資料過程的唯一方法 • 資料的遺漏程度為高度且需要最小偏差的方法，以確保可通則化程度
利用計算的替代值插補			
平均值替代	• 易於了解及執行 • 提供所有的個案有完整的資料	• 減少分布的變異 • 扭曲資料的分布 • 削弱已觀察到的相關	• 資料的遺漏程度相對較低 • 變數之間有較強的關係
迴歸插補	• 利用變數之間的真實關係 • 以觀察個案在其他變數上所得到的數值為基礎計算替代數值 • 每一個有遺漏資料的變數可以使用一組獨特的預測變數	• 強化既有的關係和減少可通則化程度 • 變數之間必須有充分的關係才能產生有效的預測數值 • 除非將誤差項納入替代數值，否則會低估變異性 • 替代數值可能「超出合理範圍」	• 資料的遺漏程度為中度或高度 • 變數間的關係必須充分確立，才不致於影響到可通則化程度 • 軟體的可取得性

資料來源：Hairs et al. (2010, 55)

劉正山, 莊文忠, 2012, 項目無反應資料的多重插補分析, 第八章, 臺灣選舉與民主化調查(TEDS)方法論之回顧與前瞻 (黃紀 主編。) pp. 276-305.

R Packages for Dealing With Missing Values

- **Amelia (Amelia II)**: A Program for Missing Data
- **hot.deck:** Multiple Hot-Deck Imputation https://cran.r-project.org/web/packages/package-name/
- **HotDeckImputation**: Hot Deck Imputation Methods for Missing Data
- **impute**: (Bioconductor) Imputation for Microarray Data
- **mi**: Missing Data Imputation and Model Checking
- **mice**: Multivariate Imputation by Chained Equations
- **missForest:** Nonparametric Missing Value Imputation using Random Forest
- **missMDA:** Handling Missing Values with Multivariate Data Analysis (e.g., imputePCA, imputeMCA,)
- **mitools**: Tools for Multiple Imputation of Missing Data
- **norm:** Analysis of Multivariate Normal Datasets with Missing Values
- **VIM**: Visualization and Imputation of Missing Values
- R packages support for missing values imputation.
  - **Hmisc:** Harrell Miscellaneous
  - **survey**: analysis of complex survey samples
  - **Zelig**: Everyone's Statistical Software
  - **rfImpute{randomForest}**: Imputations by randomForest
  - **imputation{rminer}**: Data Mining Classification and Regression Methods, Missing data imputation (e.g. substitution by value or hotdeck method).
  - **impute.svd{bcv}**: Cross-Validation for the SVD (Bi-Cross-Validation), Missing value imputation via a low-rank SVD approximation estimated by the EM algorithm.
  - **mlr**: Machine Learning in R provides several imputation methods.
    https://mlr-org.github.io/mlr-tutorial/release/html/index.html

    Package "**imputation**" was removed from the CRAN. (Archived on 2014-01-14)

R Package: MICE

- `mice`: Multivariate Imputation by Chained Equations in R by Stef van Buuren.

- Imputing missing values on mixed data.
  - **Continuous data**: Predictive mean matching, Bayesian linear regression, Linear regression ignoring model error, Unconditional mean imputation etc.
  - **Binary data**: Logistic Regression, Logistic regression with bootstrap
  - **Categorical data** (More than 2 categories) - Polytomous logistic regression, Proportional odds model etc.
  - **Mixed data** (Can work for both Continuous and Categorical) - CART, Random Forest, Sample (Random sample from the observed values).

- (NOTE: while straightforward, seemed very slow.)

Source: http://www.listendata.com/2015/08/missing-imputation-with-mice-package-in.html

Imputation using MICE Package

```
> mydata <- airquality
> mydata[4:10,3] <- rep(NA,7)
> mydata[1:5,4] <- NA
>
> #Use numerical variables as examples here.
> #Ozone is the variable with the most missing datapoints.
> data <- mydata[-c(5,6)]
> summary(mydata)
     Ozone           Solar.R          Wind             Temp           Month            Day
 Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :57.00   Min.   :5.000   Min.   : 1.0
 1st Qu.: 18.00   1st Qu.:115.8   1st Qu.: 7.400   1st Qu.:73.00   1st Qu.:6.000   1st Qu.: 8.0
 Median : 31.50   Median :205.0   Median : 9.700   Median :79.00   Median :7.000   Median :16.0
 Mean   : 42.13   Mean   :185.9   Mean   : 9.806   Mean   :78.28   Mean   :6.993   Mean   :15.8
 3rd Qu.: 63.25   3rd Qu.:258.8   3rd Qu.:11.500   3rd Qu.:85.00   3rd Qu.:8.000   3rd Qu.:23.0
 Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00   Max.   :9.000   Max.   :31.0
 NA's   :37       NA's   :7       NA's   :7        NA's   :5
>
> #Check the missing percentages for features (columns) and samples (rows)
> pMiss <- function(x){sum(is.na(x))/length(x)*100}
> apply(mydata, 2, pMiss)
    Ozone   Solar.R      Wind      Temp     Month       Day
24.183007  4.575163  4.575163  3.267974  0.000000  0.000000
> apply(mydata, 1, pMiss)
  [1] 16.66667 16.66667 16.66667 33.33333 66.66667 33.33333 16.66667 16.66667 16.66667
33.33333 16.66667  0.00000
...
[145]  0.00000  0.00000  0.00000  0.00000  0.00000 16.66667  0.00000  0.00000  0.00000
```

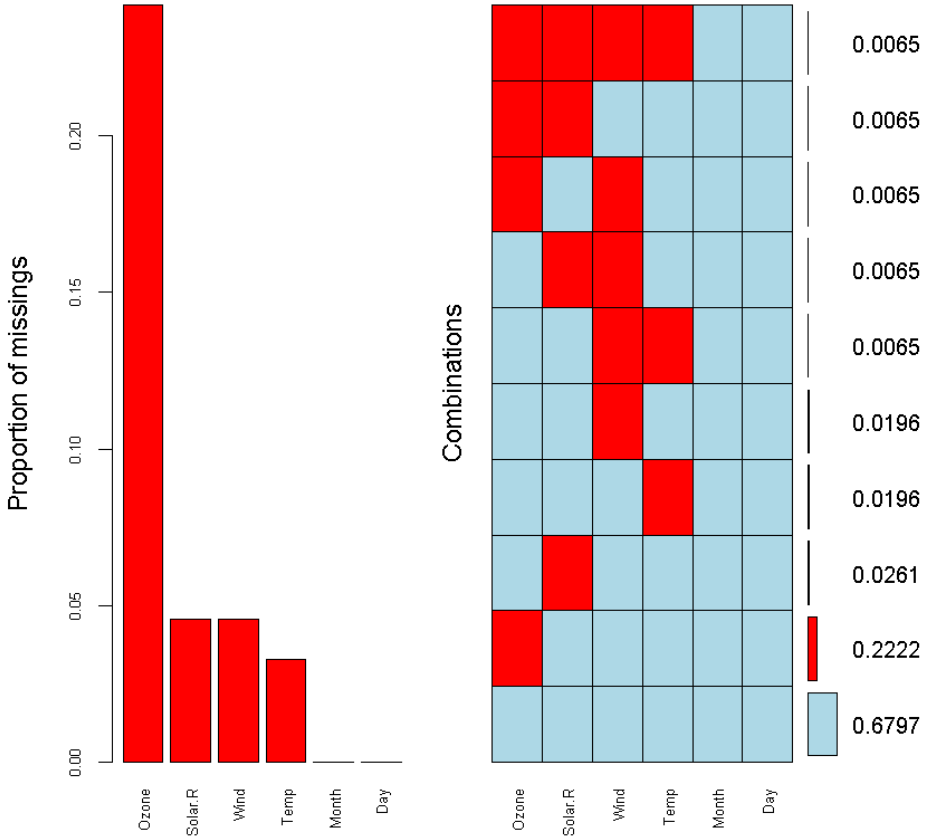
Sourec: http://www.r-bloggers.com/imputing-missing-data-with-r-mice-package/

Visualizing the Pattern of Missing Data

```
> library(mice)
> md.pattern(mydata)
    Month Day Temp Solar.R Wind Ozone
104     1   1    1       1    1     1  0
 34     1   1    1       1    1     0  1
  4     1   1    1       0    1     1  1
  3     1   1    1       1    0     1  1
  3     1   1    0       1    1     1  1
  1     1   1    1       0    1     0  2
  1     1   1    1       1    0     0  2
  1     1   1    1       0    0     1  2
  1     1   1    0       1    0     1  2
  1     1   1    0       0    0     0  4
        0   0    5       7    7    37 56
```

```
> library(VIM)
> mydata.aggrplot <- aggr(mydata,
col=c('lightblue','red'), numbers=TRUE,
prop = TRUE, sortVars=TRUE,
labels=names(mydata), cex.axis=.7, gap=3)

 Variables sorted by number of missings:
 Variable      Count
    Ozone 0.24183007
  Solar.R 0.04575163
     Wind 0.04575163
     Temp 0.03267974
    Month 0.00000000
      Day 0.00000000
```

#104 samples are complete, 34 samples miss only the Ozone measurement, 4 samples miss only the Solar.R value and so on.



Number of Observations Per Patterns for All Pairs of Variables

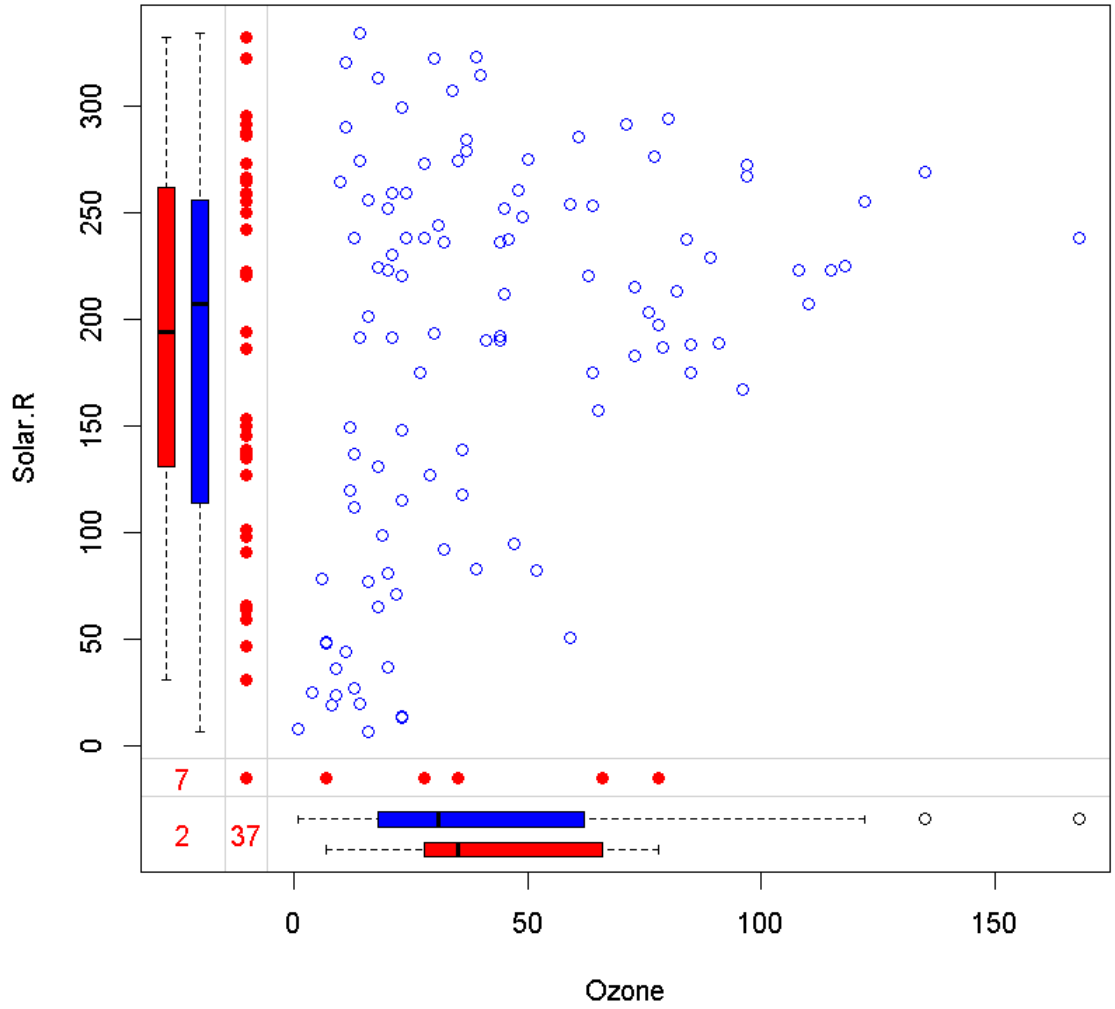

```
> md.pairs(mydata)
$rr
        Ozone Solar.R Wind Temp Month Day
Ozone     116     111  111  112   116 116
Solar.R   111     146  141  142   146 146
Wind      111     141  146  143   146 146
Temp      112     142  143  148   148 148
Month     116     146  146  148   153 153
Day       116     146  146  148   153 153

$rm
        Ozone Solar.R Wind Temp Month Day
Ozone       0       5    5    4     0   0
Solar.R    35       0    5    4     0   0
Wind       35       5    0    3     0   0
Temp       36       6    5    0     0   0
Month      37       7    7    5     0   0
Day        37       7    7    5     0   0
```

- `rr`: response-response, both variables are observed
- `rm`: response-missing, row observed, column missing
- `mr`: missing-response, row missing, column observed
- `mm`: missing-missing, both variables are missing

```
$mr
        Ozone Solar.R Wind Temp Month Day
Ozone       0      35   35   36    37  37
Solar.R     5       0    5    6     7   7
Wind        5       5    0    5     7   7
Temp        4       4    3    0     5   5
Month       0       0    0    0     0   0
Day         0       0    0    0     0   0

$mm
        Ozone Solar.R Wind Temp Month Day
Ozone      37       2    2    1     0   0
Solar.R     2       7    2    1     0   0
Wind        2       2    7    2     0   0
Temp        1       1    2    5     0   0
Month       0       0    0    0     0   0
Day         0       0    0    0     0   0
```

Marginplot

```
> marginplot(mydata[,c("Ozone", "Solar.R")], col = c("blue", "red"))
```



- The blue box plot located on the left and bottom margins shows the distribution of the non-missing datapoints.
- The red box plot on the left shows the distribution of Solar.R with Ozone missing while
- Likewhise for the Ozone box plots at the bottom of the graph.
- If our assumption of MCAR data is correct, then we expect the red and blue box plots to be very similar.

Generates Multivariate Imputations by Chained Equations (MICE)

```
mice(data, m = 5, method = vector("character", length = ncol(data)),
  predictorMatrix = (1 - diag(1, ncol(data))),
  visitSequence = (1:ncol(data))[apply(is.na(data), 2, any)],
  form = vector("character", length = ncol(data)),
  post = vector("character", length = ncol(data)), defaultMethod = c("pmm",
  "logreg", "polyreg", "polr"), maxit = 5, diagnostics = TRUE,
  printFlag = TRUE, seed = NA, imputationMethod = NULL,
  defaultImputationMethod = NULL, data.init = NULL, ...)
```

```
> methods(mice)
 [1] mice.impute.2l.norm      mice.impute.2l.pan       mice.impute.2lonly.mean
 [4] mice.impute.2lonly.norm  mice.impute.2lonly.pmm   mice.impute.cart
 [7] mice.impute.fastpmm      mice.impute.lda          mice.impute.logreg
[10] mice.impute.logreg.boot  mice.impute.mean         mice.impute.norm
[13] mice.impute.norm.boot    mice.impute.norm.nob     mice.impute.norm.predict
[16] mice.impute.passive      mice.impute.pmm          mice.impute.polr
[19] mice.impute.polyreg      mice.impute.quadratic    mice.impute.rf
[22] mice.impute.ri           mice.impute.sample       mice.mids
[25] mice.theme
see '?methods' for accessing help and source
Warning message:
In .S3methods(generic.function, class, paren
  function 'mice' appears not to be S3 generic; fou
```

Method	Description	Scale type	Default
pmm	Predictive mean matching	numeric	Y
norm	Bayesian linear regression	numeric	
norm.nob	Linear regression, non-Bayesian	numeric	
mean	Unconditional mean imputation	numeric	
2L.norm	Two-level linear model	numeric	
logreg	Logistic regression	factor, 2 levels	Y
polyreg	Multinomial logit model	factor, >2 levels	Y
polr	Ordered logit model	ordered, >2 levels	Y
lda	Linear discriminant analysis	factor	
sample	Random sample from the observed data	any	

PMM (Predictive Mean Matching) – For numeric v

Impute Missing Values

```
> mydata.ip <- mice(mydata, m=5, maxit=50, meth='pmm', seed=500)

 iter imp variable
  1   1  Ozone  Solar.R  Wind  Temp
  1   2  Ozone  Solar.R  Wind  Temp
...
  50   4  Ozone  Solar.R  Wind  Temp
  50   5  Ozone  Solar.R  Wind  Temp
> summary(mydata.ip)
Multiply imputed data set
Call:
mice(data = mydata, m = 5, method = "pmm", maxit = 50, seed = 500)
Number of multiple imputations:  5
Missing cells per column:
  Ozone Solar.R    Wind    Temp   Month     Day
     37       7       7       5       0       0
Imputation methods:
  Ozone Solar.R    Wind    Temp   Month     Day
  "pmm"   "pmm"   "pmm"   "pmm"   "pmm"   "pmm"
VisitSequence:
  Ozone Solar.R    Wind    Temp
      1       2       3       4
PredictorMatrix:
        Ozone Solar.R Wind Temp Month Day
Ozone       0       1    1    1     1   1
Solar.R     1       0    1    1     1   1
Wind        1       1    0    1     1   1
Temp        1       1    1    0     1   1
Month       0       0    0    0     0   0
Day         0       0    0    0     0   0
Random generator seed value:  500
```

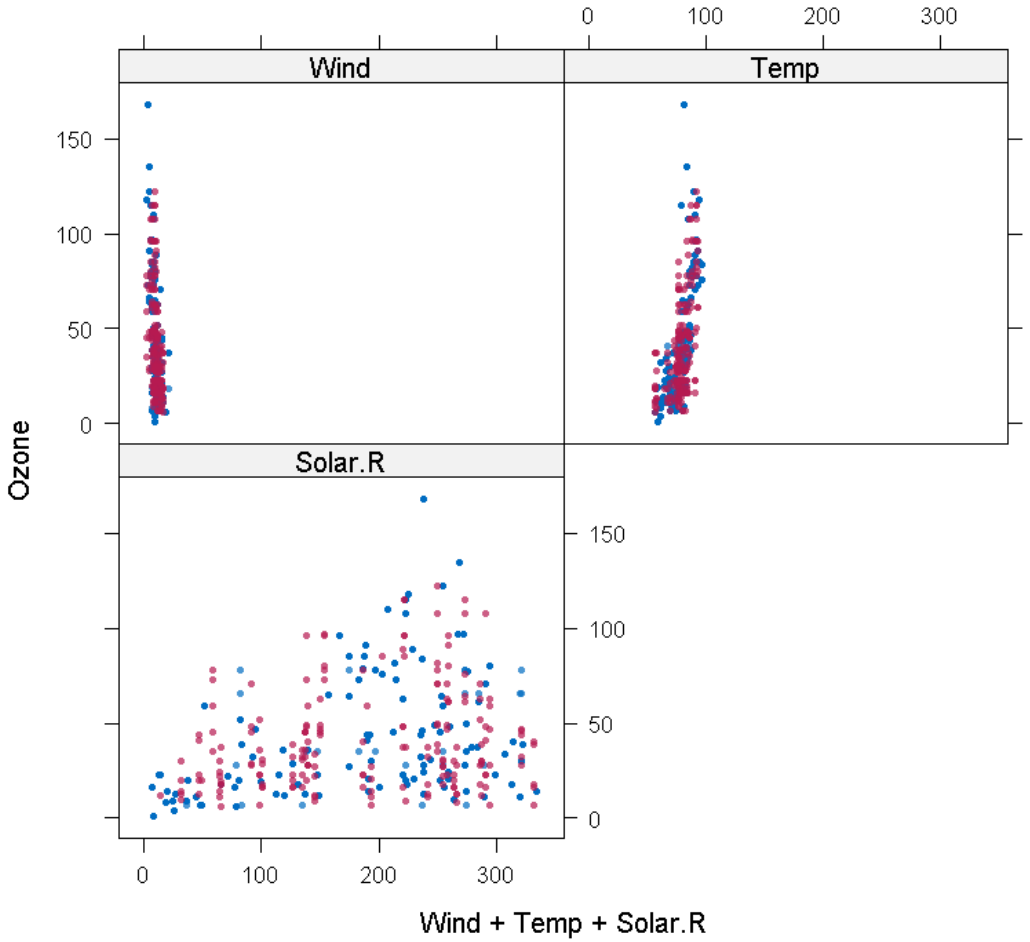
```
> mydata.ip$imp$Ozone
>  1   2   3   4   5
5    59  85  20 108  18
10   11   7  27  14  21
...
150   9  34  27  12  22
```

The output shows the imputed data for each observation (first column left) within each imputed dataset (first row at the top).

```
> # get back the first completed dataset out of 5
> mydata.completed <- complete(mydata.ip, 1)
```

Compare the Distributions of Original and Imputed data

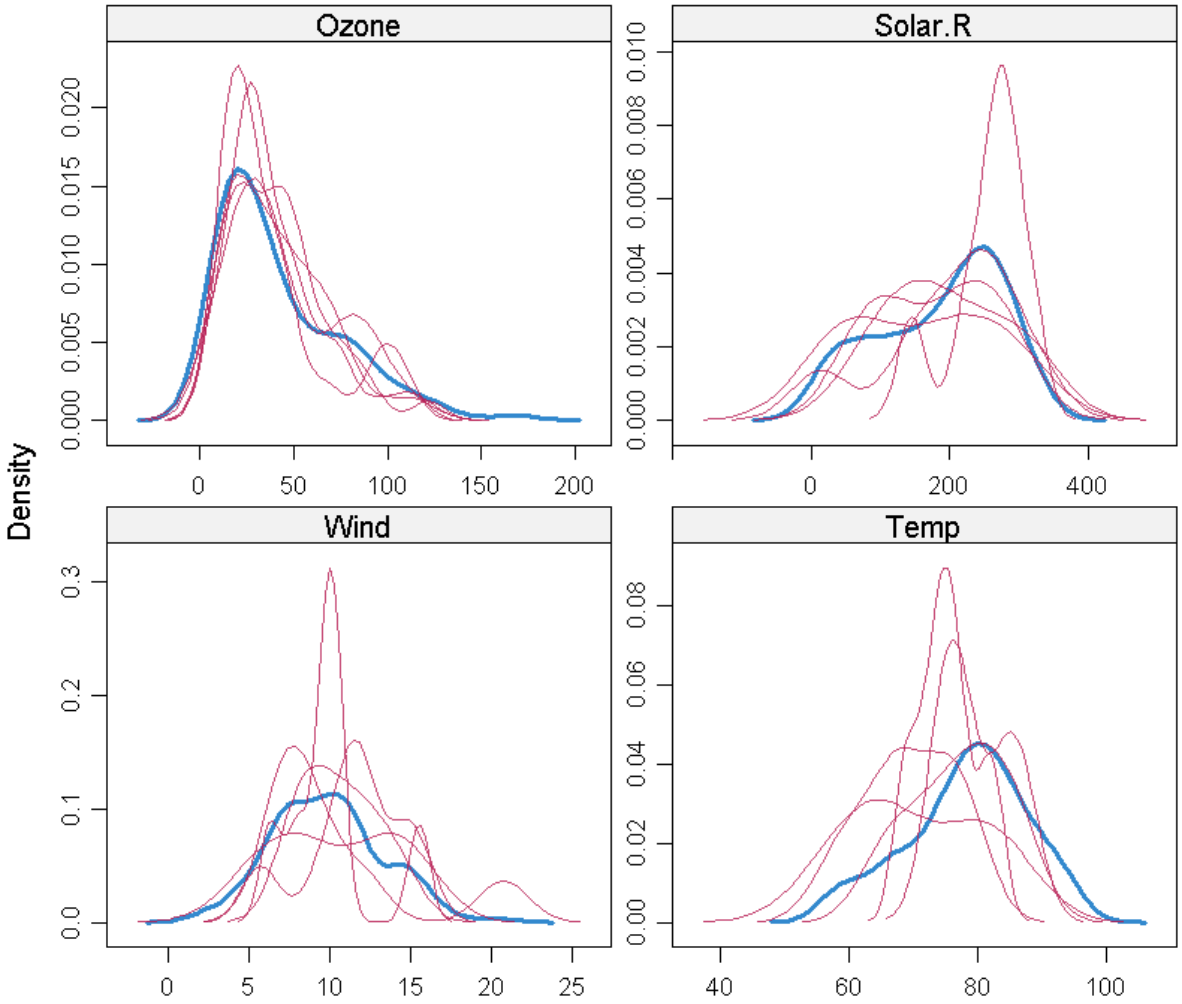

```
> library(lattice)
> xyplot(mydata.ip, Ozone ~ Wind + Temp + Solar.R, pch=16, cex=0.5)
```



- Check if the shape of the imputed points (magenta) matches the shape of the observed (blue) ones (observed).
- The matching shape means the imputed values are indeed "plausible values".

Density Plot

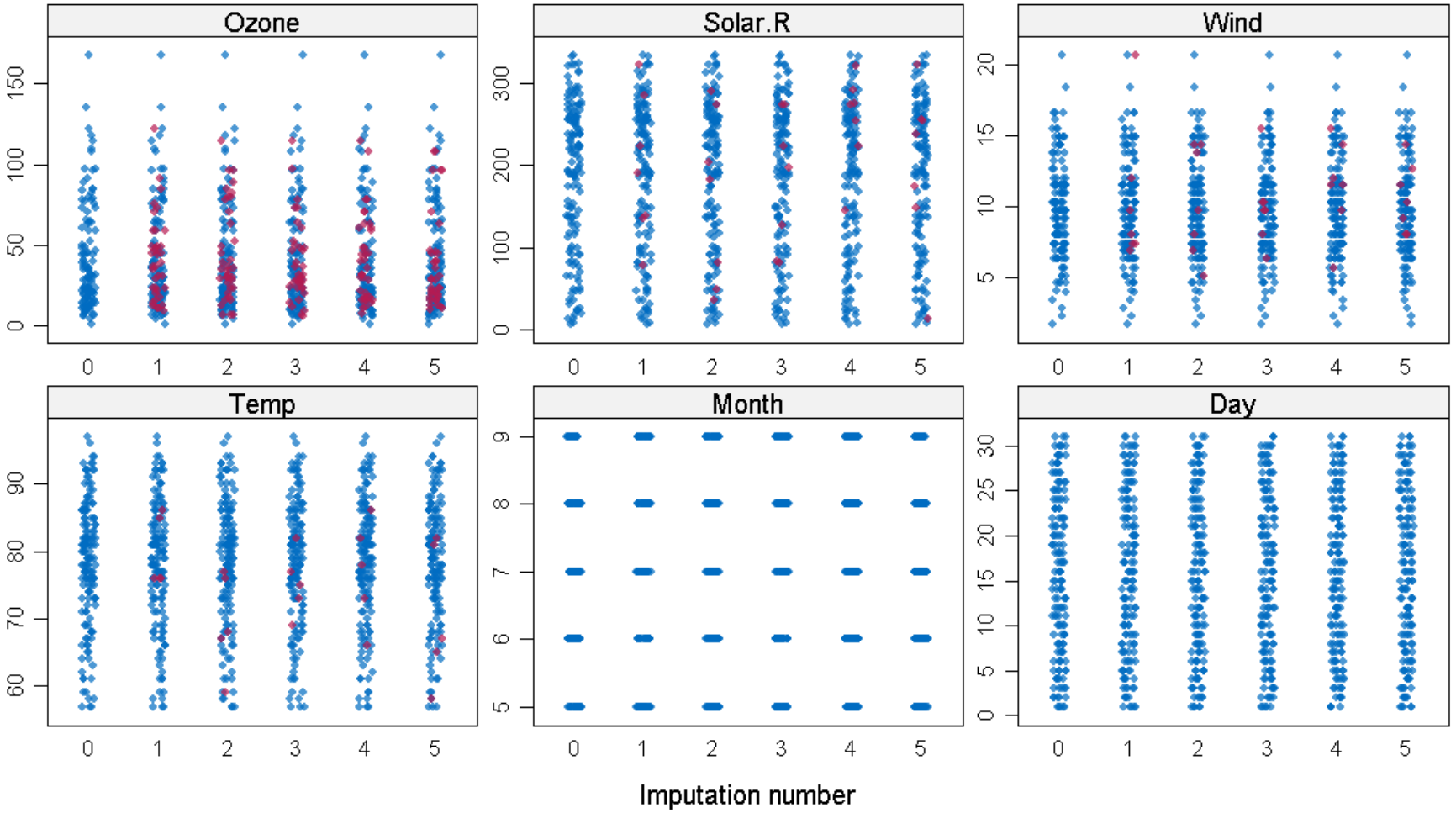
```
> densityplot(mydata.ip)
```



The density of the imputed data for each imputed dataset is showed in magenta while the density of the observed data is showed in blue. Under MCAR, we expect the distributions to be similar.

Shows the Distributions of the Variables as Individual Points

```
> stripplot(mydata.ip, pch = 16, cex = 0.6)
```



Pooling

- Next step: fit a linear model to the data.
- **mice** fit a model to each of the imputed dataset and then pool the results together.

```
> # linear regression for each imputed data set - 5 regression are run
> modelFit1 <- with(mydata.ip, lm(Temp~ Ozone + Solar.R+Wind))
> # pool coefficients and standard errors across all 5 regression models
> summary(pool(modelFit1))
                   est          se          t       df     Pr(>|t|)        lo 95
(Intercept) 71.11418579 2.840129171 25.0390674 85.04465 0.000000e+00 65.467290906
Ozone        0.17412083 0.025108183  6.9348239 72.90551 1.383136e-09  0.124079199
Solar.R      0.01004273 0.007163085  1.4020115 87.03503 1.644683e-01 -0.004194599
Wind        -0.21504110 0.222484210 -0.9665454 61.98616 3.375274e-01 -0.659782671
                 hi 95 nmis       fmi    lambda
(Intercept) 76.76108067   NA 0.1459648 0.1261138
Ozone        0.22416246   37 0.1734348 0.1510666
Solar.R      0.02428005    7 0.1418215 0.1223252
Wind         0.22970047    7 0.2026905 0.1773735
```

To reduce the effect of the random seed initialization, we can impute a higher number of dataset, by changing the default **m=5** parameter in the **mice()** function.

```
mydata.ip2 <- mice(mydata, m=50, seed=245435)
modelFit2 <- with(mydata.ip2,lm(Temp ~ Ozone + Solar.R + Wind))
summary(pool(modelFit2))
```

Quick Tutorial on MICE Package

```
> # Generate 10% missing values at Random
> iris.mis <- prodNA(iris, noNA = 0.1)  # library(missForest)
> # Check missing values introduced in the data
> summary(iris.mis)
> iris.mis <- subset(iris.mis, select = -c(Species))
> summary(iris.mis)
>
> #  A tabular form of missing value present in each variable
> library(mice)
> md.pattern(iris.mis)
> # Visualization
> library(VIM)
> 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"))

> #  Imputation
> imputed_Data <- mice(iris.mis, m=5, maxit = 50, method = 'pmm', seed = 500)
> summary(imputed_Data)
> # Check imputed values
> imputed_Data$imp$Sepal.Width
> # Get complete data ( 2nd out of 5)
> completeData <- complete(imputed_Data,2)
> #  Build predictive model
> fit <- with(data = imputed_Data, exp = lm(Sepal.Width ~ Sepal.Length + Petal.Width))
> #  Combine results of all 5 models
> combine <- pool(fit)
> summary(combine)
```

Source: http://www.analyticsvidhya.com/blog/2016/03/tutorial-powerful-packages-imputing-missing-values/

R Package: Amelia

- The package is named after Amelia Earhart, a famous American woman aviator who went missing over the ocean. Amelia is also a name of a birth defect of lacking one or more limbs.

愛蜜莉亞·艾爾哈特

愛蜜莉亞·艾爾哈特約1935年照片

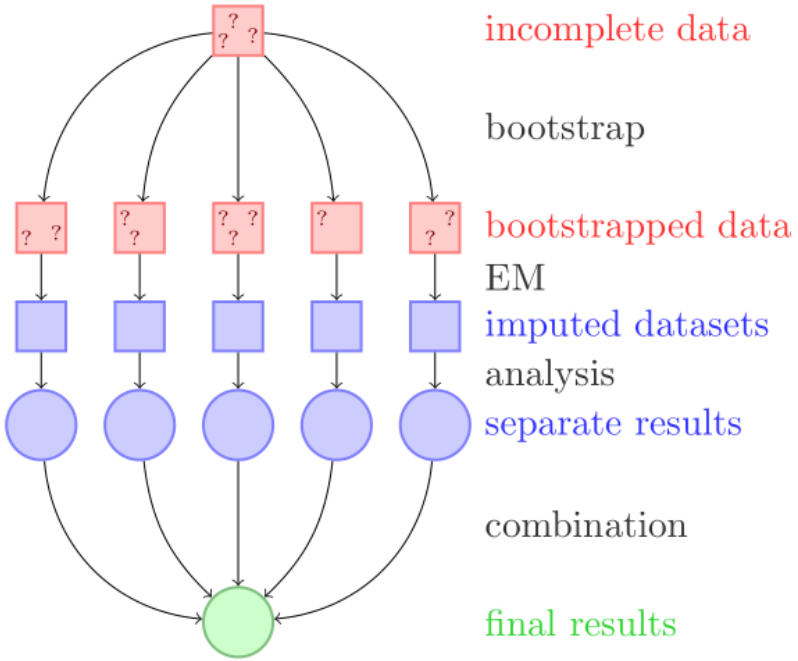
https://en.wikipedia.org/wiki/Amelia_Earhart

- **Properties:**
  - Performs **multiple imputation** (generate imputed data sets).
  - **Bootstrap based EMB** (expectation-maximization with bootstrapping) algorithm makes it faster and robust to impute many variables including cross sectional, time series data etc.
  - Parallel imputation feature using multicore CPUs.

- **Assumptions:**
  - All variables in a data set have **Multivariate Normal Distribution** (**MVN**). It uses means and covariances to summarize data.
  - Missing data is random in nature (Missing at Random)

Amelia: Multiple Imputation with the EMB Algorithm

- Take ***m*** bootstrap samples and apply EMB algorithm to each sample.
- The ***m*** estimates of mean and variances will be different.
- The first set of estimates are used to impute first set of missing values using regression, then second set of estimates are used for second set and so on.



James Honaker, Gary King, Matthew Blackwell, 2011, Amelia II: A Program for Missing Data, Journal of Statistical Software, December 2011, Volume 45, Issue 7.

Quick Tutorial on Amelia Package

- Comparing with **MICE**, MVN lags on some crucial aspects such as:
  - **MICE** imputes data on variable by variable basis whereas MVN uses a joint modeling approach based on multivariate normal distribution.
  - **MICE** is capable of handling different types of variables whereas the variables in MVN need to be normally distributed or transformed to approximate normality.
  - **MICE** can manage imputation of variables defined on a subset of data whereas MVN cannot.

```
> library(Amelia)
> # Seed 10% missing values
> ir is.mis <- prodNA(iris, noNA = 0.1) # library(missForest)
> summary(iris.mis)
> # Specify columns and run amelia
> amelia_fit <- amelia(iris.mis, m=5, parallel = "multicore", noms =
"Species")
> # Access imputed outputs
> st r(amelia_fit)
> amelia_fit$imputations[[1]]
> amelia_fit$imputations[[2]]
> # Check a particular column in a data set
> am elia_fit$imputations[[5]]$Sepal.Length
> # Export the outputs to csv files
> write.amelia(amelia_fit, file.stem = "imputed_data_set")
```

R Package: mi

- **mi** (Multiple imputation with diagnostics) : multiple imputation using predictive mean matching method.

- **Some unique characteristics**:
  - It allows **graphical diagnostics** of imputation models and convergence of imputation process.
  - It uses **bayesian** version of regression models to handle issue of separation.
  - Imputation model specification is similar to regression output in R.
  - It automatically detects **irregularities** in data such as high collinearity among variables.
  - It adds noise to imputation process to solve the problem of additive constraints.

```
> library(Hmisc)
> library(mi)
> # seed missing values ( 10% )
> iris.mis <- prodNA(iris, noNA = 0.1)
> summary(iris.mis)
> # Imputing missing value with mi
> mi_data <- mi(iris.mis, seed = 335)
> summary(mi_data)
```

Source: http://www.analyticsvidhya.com/blog/2016/03/tutorial-powerful-packages-imputing-missing-values/

R Package: Hmisc

- Hmisc is useful for data analysis, high–level graphics, imputing missing values, advanced table making, model fitting & diagnostics etc.
  - `impute()`: imputes missing value using user defined statistical method (mean, max, mean).
  - `aregImpute()`: allows mean imputation using additive regression, bootstrapping, and predictive mean matching.

- **How it works:**
  - Different bootstrap resamples are used for each of multiple imputations.
  - A flexible additive model (non parametric regression method) is fitted on samples taken with replacements from original data and missing values (acts as dependent variable) are predicted using non-missing values (independent variable).
  - It uses **predictive mean matching** (default) to impute missing values.

- **Some properties:**
  - Assume linearity in the variables being predicted.
  - Fisher's optimum scoring method is used for predicting categorical variables.

Source: http://www.analyticsvidhya.com/blog/2016/03/tutorial-powerful-packages-imputing-missing-values/

NOTE: `Hmisc` should be your first choice of missing value imputation followed by `missForest` and `MICE`.

Imputation using Hmisc Package

```
> library(Hmisc)
> # Seed missing values ( 10% )
> iris.mis <- prodNA(iris, noNA = 0.1)
> summary(iris.mis)
>
> # Impute with mean value (or min, max, median)
> iris.mis$imputed_age <- with(iris.mis, impute(Sepal.Length, mean))
> # Impute with random value
> iris.mis$imputed_age2 <- with(iris.mis, impute(Sepal.Length, 'random'))
> # Using argImpute
> impute_arg <- aregImpute(~ Sepal.Length + Sepal.Width + Petal.Length +
Petal.Width + Species, data = iris.mis, n.impute = 5)
>
> # argImpute() automatically identifies the variable type and treats them
accordingly.
> impute_arg
> #  Higher R² values, better are the values predicted.
>
> # Check imputed variable Sepal.Length
> impute_arg$imputed$Sepal.Length
```

Source: http://www.analyticsvidhya.com/blog/2016/03/tutorial-powerful-packages-imputing-missing-values/

Other Online Examples: Example 6 Multiple Imputation& Missing Data, coreysparks,March 2, 2015
https://rpubs.com/corey_sparks/63681

Which Imputation Method?

- KNN is the most widely-used.

- Characteristics of data that may affect choice of imputation method:
  - dimensionality
  - percentage of values missing
  - experimental design (time series, case/control, etc.)
  - patterns of correlation in data

- Suggestion!!
  - add (**same percentage**) artificial missing values to your (**complete cases**) data set
  - impute them with various methods
  - see which is best (since you know the real value)

Junk, Noisy Data, or Outlier

- As in a physics or statistics test, noise is a random error that occurs during the test process to seize the measured data. No matter what means you apply to the data gathering process, noise inevitably exists.

- **Deal with noisy data using smoothing**:
  - **Binning**: This is a local scope smoothing method in which the neighborhood values are used to compute the final value for the certain bin. The sorted data is distributed into a number of bins and each value in that bin will be replaced by a value depending on some certain computation of the neighboring values. The computation can be bin median, bin boundary, which is the boundary data of that bin.
  - **Regression**: The target of regression is to find the best curve or something similar to one in a multidimensional space; as a result, the other values will be used to predict the value of the target attribute or variable. In other aspects, it is a popular means for smoothing.

- Classification or outlier: The classifier is another inherent way to find the noise or outlier. During the process of classifying, most of the source data is grouped into couples of groups, except the outliers.

Outliers Detection

- **Graphical techniques**: index plot, Boxplot side-by-side, scatterplot, heatmap and so on.
- **R packages:**
  - **`outliers`**: Tests for outliers
    - A collection of some tests commonly used for identifying outliers.

    https://cran.r-project.org/web/packages/outliers/index.html
    - Grubbs' test (Grubbs 1969 and Stefansky 1972) is used to detect outliers in a univariate data set. It is based on the assumption of normality. That is, you should first verify that your data can be reasonably approximated by a normal distribution before applying the Grubbs' test.
  - **`extRemes`**: Extreme Value Analysis.
  - **`in2extRemes`**: Into the extRemes Package, GUI to some of the functions in the package extRemes. (http://www.assessment.ucar.edu/toolkit/)
  - **`extremevalues`**: Univariate Outlier Detection
  - **Extreme Value Analysis(EVA) packages in R: `evd, evdbayes, evir, fExtremes, lmom, SpatialExtremes, texmex, extRemes, ismev, texmex, ismev`**

- **Robust approaches to data with outliers**
  - Robustify the classical algorithm by replacing the sample mean vector and covariance matrix with the robust location and scatter estimators.

See also: Chapter 7, Outlier Detection, RDataMining-book-2015

R package: outliers

- Statistlcal Tests:
  - **chisq.out.test**: Chi-squared test for outlier
  - **cochran.test**: Test for outlying or inlying variance
  - **dixon.test**: Dixon tests for outlier
  - **grubbs.test**: Grubbs tests for one or two outliers in data sample.

```
> library(outliers)
> set.seed(1234)
> x <- rnorm(10)
> dixon.test(x)
        Dixon test for outliers

data:  x
Q = 0.39927, p-value = 0.2187
alternative hypothesis: lowest value -2.34569770262935 is an outlier
> dixon.test(x, opposite=TRUE)
        Dixon test for outliers

data:  x
Q = 0.2524, p-value = 0.6484
alternative hypothesis: highest value 1.08444117668306 is an outlier
```


- Dixon, W.J. (1950). Analysis of extreme values. Ann. Math. Stat. 21, 4, 488-506.
- Dixon, W.J. (1951). Ratios involving extreme values. Ann. Math. Stat. 22, 1, 68-78.
- Snedecor, G.W., Cochran, W.G. (1980). Statistical Methods (seventh edition). Iowa State University Press, Ames, Iowa.
- Grubbs, F.E. (1950). Sample Criteria for testing outlying observations. Ann. Math. Stat. 21, 1, 27-58.

Robust Statistical Methods

CRAN Task View: Robust Statistical Methods

https://cran.r-project.org/web/views/Robust.html

- Robust Location and Scatter Estimators
  - Median, MAD (median of the absolute deviations from the median)
  - M-estimator (Huber, 1964; Maronna, 1976)
  - Stahel-Donoho estimator (Stahel, 1981; Donoho, 1982)
  - MVE (minimum volume ellipsoid), MCD (minimum covariance determinant) (Rousseeuw, 1983, 1984, 1985)
  - S-estimator (Davis, 1987)
  - Depth weighted and maximum depth estimators (Zuo, Cui and He, 2004)

MVE (minimum volume ellipsoid)

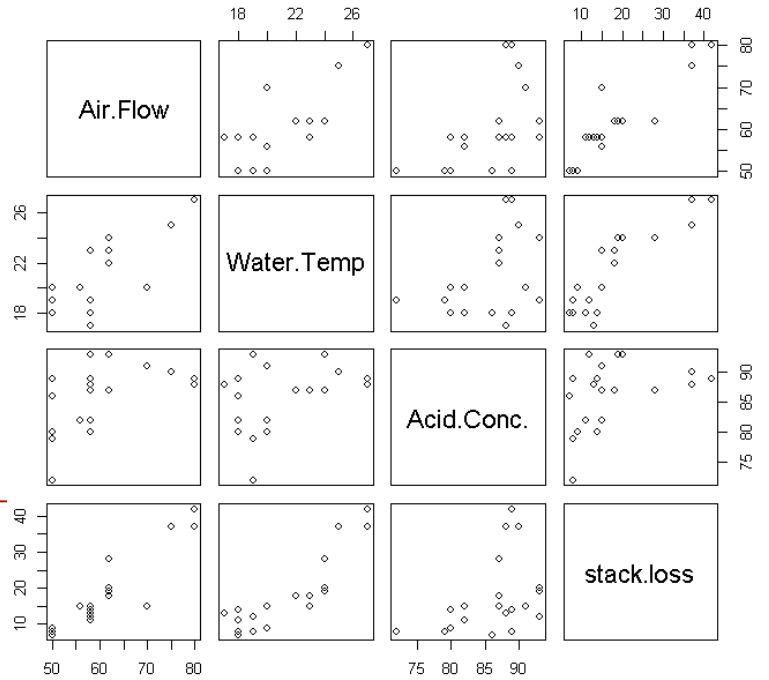
- Affine equivariant with high breakdown points.
- The existing efficient algorithm for computation.
- Readily available implementations.
- Ability to Identify extreme values.

$$\text{Outlier values} \sim 2 \times \sqrt{\chi^2_{0.975,p}} + N(0,1)$$

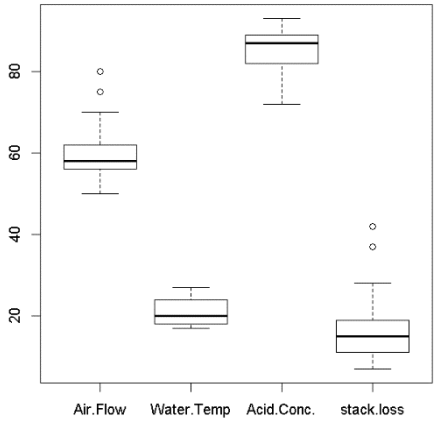
```
> qchisq(0.975,5)
[1] 12.83250
```

Brownlee's Stack Loss Plant Data

- **stackloss {datasets}**, Operational data of a plant for the oxidation of ammonia to nitric acid.
  - Air.Flow: Flow of cooling air
  - Water.Temp: Cooling Water Inlet Temperature
  - Acid.Conc.: Concentration of acid [per 1000, minus 500]
  - stack.loss: Stack loss

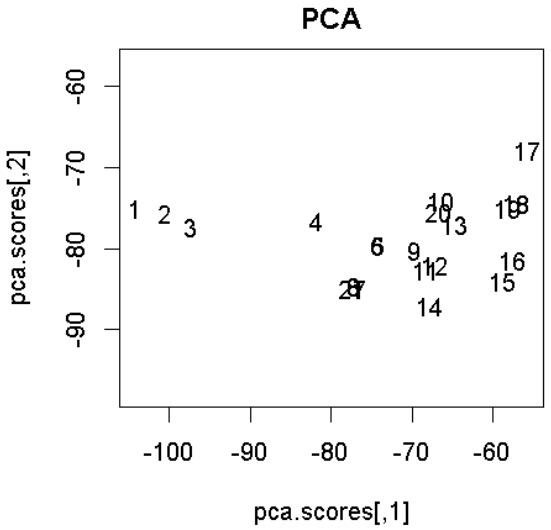



```
> data(stackloss)
> dim(stackloss)
[1] 21  4
> head(stackloss, 4)
  Air.Flow Water.Temp Acid.Conc. stack.loss
1       80         27         89         42
2       80         27         88         37
3       75         25         90         37
4       62         24         87         28
> summary(stackloss)
    Air.Flow       Water.Temp     Acid.Conc.      stack.loss
 Min.   :50.00   Min.   :17.0   Min.   :72.00   Min.   : 7.00
 1st Qu.:56.00   1st Qu.:18.0   1st Qu.:82.00   1st Qu.:11.00
 Median :58.00   Median :20.0   Median :87.00   Median :15.00
 Mean   :60.43   Mean   :21.1   Mean   :86.29   Mean   :17.52
 3rd Qu.:62.00   3rd Qu.:24.0   3rd Qu.:89.00   3rd Qu.:19.00
 Max.   :80.00   Max.   :27.0   Max.   :93.00   Max.   :42.00
```



Robust PCA

```
> library(MASS)
> cov(stackloss)
           Air.Flow Water.Temp Acid.Conc. stack.loss
Air.Flow   84.05714  22.657143  24.571429   85.76429
Water.Temp 22.65714   9.990476   6.621429   28.14762
Acid.Conc. 24.57143   6.621429  28.714286   21.79286
stack.loss 85.76429  28.147619  21.792857  103.46190
> cov.mve(stackloss)$cov
            Air.Flow Water.Temp Acid.Conc. stack.loss
Air.Flow   21.600000   6.657143  11.285714  18.228571
Water.Temp  6.657143   6.066667   4.690476   7.900000
Acid.Conc. 11.285714   4.690476  23.095238   9.642857
stack.loss 18.228571   7.900000   9.642857  17.828571
```




```
> par(mfrow=c(2,1))
> library(MASS)
> ccov <- cov(stackloss)
> pca.scores <- as.matrix(stackloss) %*%
eigen(ccov)$vectors[,1:2]
> plot(pca.scores, main="PCA", asp=1, type="n")
> text(pca.scores, label=1:nrow(stackloss))
> rcov <- cov.mve(stackloss)$cov
> rpca.scores <- as.matrix(stackloss) %*%
                 eigen(rcov)$vectors[,1:2]
> plot(rpca.scores, main="Robust PCA", asp=1, type="n")
> text(rpca.scores, label=1:nrow(stackloss))
```