Chapter 3
Managing Data in R

In this Chapter, we will discuss strategies to `import` data and `export` results. Also, we are going to learn the basic tricks we need to know about processing different types of data. Specifically, we will illustrate common `R` data structures and strategies for loading (ingesting) and saving (regurgitating) data. In addition, we will (1) present some basic statistics, e.g., for measuring central tendency (mean, median, mode) or dispersion (variance, quartiles, range); (2) explore simple plots; (3) demonstrate the uniform and normal distributions; (4) contrast numerical and categorical types of variables; (5) present strategies for handling incomplete (missing) data; and (6) show the need for cohort-rebalancing when comparing imbalanced groups of subjects, cases or units.

3.1 Saving and Loading R Data Structures

Let's start by extracting the Edgar Anderson's Iris Data from the package `datasets`. The iris dataset quantifies morphologic shape variations of 50 Iris flowers of three related genera – *Iris setosa*, *Iris virginica* and *Iris versicolor*. Four shape features were measured from each sample – length and the width of the sepals and petals (in centimeters). These data were used by Ronald Fisher in his 1936 linear discriminant analysis paper (Fig. 3.1).

```
data()
data(iris)
class(iris)

## [1] "data.frame"
```

I. D. Dinov, *Data Science and Predictive Analytics*,
https://doi.org/10.1007/978-3-319-72347-1_3

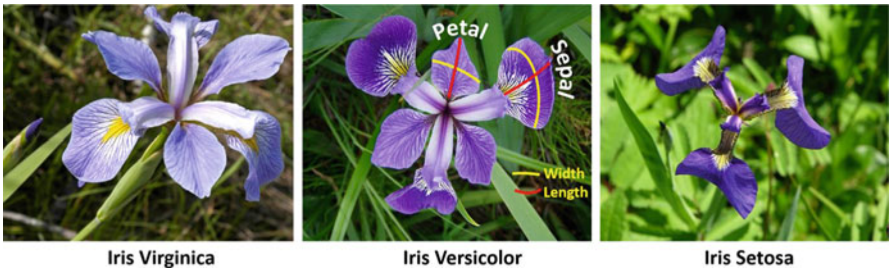

Fig. 3.1 Definitions of petal width and length for the three iris flower genera used in the example below

As an I/O (input/output) demonstration, after we load the `iris` data and examine its class type, we can save it into a file named "myData.RData" and then reload it back into `R`.

```
save(iris, file="myData.RData")
load("myData.RData")
```

3.2 Importing and Saving Data from CSV Files

Importing the data from `"CaseStudy07_WorldDrinkingWater_Data.csv"` from these case-studies (https://umich.instructure.com/courses/38100/files/folder/Case_Studies) and saving it into the R dataset named “water” The variables in the dataset are as follows:

- **Time**: Years (1990, 1995, 2000, 2005, 2010, 2012)
- **Demographic**: Country (across the world)
- **Residence Area Type**: Urban, rural, or total
- **WHO Region**
- **Population using improved drinking water sources**: The percentage of the population using an improved drinking water source.
- **Population using improved sanitation facilities**: The percentage of the population using an improved sanitation facility.

Generally, the separator of a CSV file is comma. By default, we have option `sep = ", "` in the command `read.csv()`. Also, we can use `colnames()` to rename the column variables.

```
water <- read.csv('https://umich.instructure.com/files/399172/download?downl
oad_frd=1', header=T)
water[1:3, ]

##   Year..string. WHO.region..string. Country..string.
## 1          1990              Africa          Algeria
## 2          1990              Africa           Angola
## 3          1990              Africa            Benin
##   Residence.Area.Type..string.
## 1                        Rural
## 2                        Rural
## 3                        Rural
##   Population.using.improved.drinking.water.sources......numeric.
## 1                                                             88
## 2                                                             42
## 3                                                             49
##   Population.using.improved.sanitation.facilities......numeric.
## 1                                                            77
## 2                                                             7
## 3                                                             0

colnames(water)<-c("year", "region", "country", "residence_area", "improved_
water", "sanitation_facilities")
water[1:3, ]

##   year region country residence_area improved_water sanitation_facilities
## 1 1990 Africa Algeria          Rural             88                    77
## 2 1990 Africa  Angola          Rural             42                     7
## 3 1990 Africa   Benin          Rural             49                     0

which.max(water$year);

## 913

# rowMeans(water[,5:6])
mean(water[,6], trim=0.08, na.rm=T)

## [1] 71.63629
```

This code loads CSV files that already include a header line listing the names of the variables. If we don't have a header in the dataset, we can use the `header = FALSE` option (https://umich.instructure.com/courses/38100/files/folder/Case_Studies). R will assign default names to the column variables of the dataset.

```
Simulation <- read.csv("https://umich.instructure.com/files/354289/download?
download_frd=1", header = FALSE)
Simulation[1:3, ]

##   V1 V2  V3    V4       V5  V6  V7   V8     V9    V10      V11     V12
## 1 ID i2 age treat homeless pcs mcs cesd indtot pss_fr drugrisk sexrisk
## 2  1  0  25     0        0  49   7   46     37      0        1       6
## 3  2 18  31     0        0  48  34   17     48      0        0      11
##       V13    V14       V15     V16
## 1 satreat female substance racegrp
## 2       0      0   cocaine   black
## 3       0      0   alcohol   white
```

To save a data frame to CSV files, we could use the `write.csv()` function. The option `file` = `"a/local/file/path"` allows us edit the saved file path.

```
write.csv(iris, file = "C:/Users/iris.csv") # Iris data

write.csv(water, file = "C:/Users/WHO_Water.csv") # World Drinking Water
```

3.3 Exploring the Structure of Data

We can use the command `str()` to explore the structure of a dataset. For instance, using the World Drinking Water dataset:

```
str(water)

## 'data.frame':    3331 obs. of  6 variables:
##  $ year                 : int  1990 1990 1990 1990 1990 1990 1990 1990 19
90 1990 ...
##  $ region               : Factor w/ 6 levels "Africa","Americas",..: 1 1
1 1 1 1 1 1 1 1 ...
##  $ country              : Factor w/ 192 levels "Afghanistan",..: 3 5 19 2
3 26 27 30 32 33 37 ...
##  $ residence_area       : Factor w/ 3 levels "Rural","Total",..: 1 1 1 1
1 1 1 1 1 1 ...
##  $ improved_water       : num  88 42 49 86 39 67 34 46 37 83 ...
##  $ sanitation_facilities: num  77 7 0 22 2 42 27 12 4 11 ...
```

We can see that this (`World Drinking Water`) dataset has 3331 observations and 6 variables. The output also give us the class of each variable and first few elements in the variable.

3.4 Exploring Numeric Variables

Summary statistics for numeric variables in the dataset could be accessed by using the command `summary()` (Fig. 3.2).

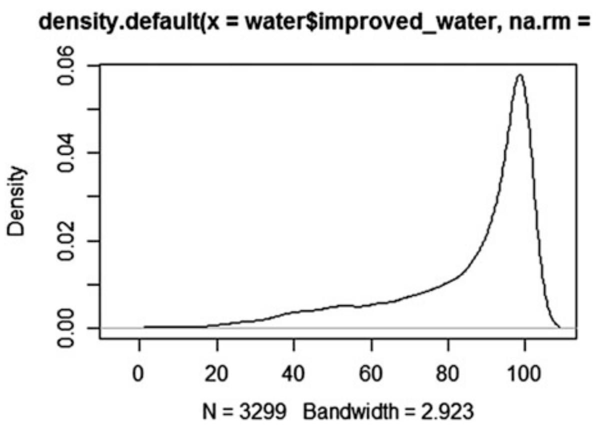

Fig. 3.2 Density plot of the water improvement variable in the World Health Organization (WHO) water quality case-study.

```
summary(water$year)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1990    1995    2005    2002    2010    2012

summary(water[c("improved_water", "sanitation_facilities")])

##  improved_water  sanitation_facilities
##  Min.   :  3.0   Min.   :  0.00
##  1st Qu.: 77.0   1st Qu.: 42.00
##  Median : 93.0   Median : 81.00
##  Mean   : 84.9   Mean   : 68.87
##  3rd Qu.: 99.0   3rd Qu.: 97.00
##  Max.   :100.0   Max.   :100.00
##  NA's   :32      NA's   :135

plot(density(water$improved_water,na.rm = T))

# variables need not be continuous,  we can still get intuition about their
distribution
```

The six summary statistics and NA's (missing data) are reported in the R output above.

3.5 Measuring the Central Tendency: Mean, Median, Mode

Mean and **median** are two frequent measurements of the central tendency. Mean is "the sum of all values divided by the number of values". Median is the number in the middle of an ordered list of values. In R, `mean()` and `median()` functions provide us with these two measurements.

```
vec1<-c(40, 56, 99)
mean(vec1)

## [1] 65

mean(c(40, 56, 99))

## [1] 65

median(vec1)

## [1] 56

median(c(40, 56, 99))

## [1] 56

# install.packages("psych");
library("psych")
geometric.mean(vec1, na.rm=TRUE)

## [1] 60.52866
```

The **mode** is the value that occurs most often in the dataset. It is often used in categorical data, where mean and median are inappropriate measurements.

We can have one or more modes. In the water dataset, we have “Europe” and “Urban” as the modes for region and residence area, respectively. These two variables are unimodal, which has a single mode. For the year variable, we have two modes: 2000 and 2005. Both of the categories have 570 counts. The year variable is an example of a bimodal. We also have multimodal data that has two or more modes.

Mode is just one of the measures for the central tendency. The best way to use it is to compare the counts of the mode to other values. This help us to judge whether one or several categories dominates all others in the data. After that, we are able to analyze the meaning behind these common centrality measures.

In numeric datasets, the mode(s) represents the highest bin(s) in the histogram. In this way, we can also examine if the numeric data is multimodal.

More information about measures of centrality is available here (http://wiki.socr.umich.edu/index.php/AP_Statistics_Curriculum_2007_EDA_Center).

3.6 Measuring Spread: Quartiles and the Five-Number Summary

The five-number summary describes the spread of a dataset. They are:

- Minimum (`Min.`), representing the smallest value in the data
- First quantile/Q1 (`1st Qu.`), representing the 25th percentile, which splits off the lowest 25% of data from the highest 75%
- Median/Q2 (`Median`), representing the 50th percentile, which splits off the lowest 50% of data from the top 50%
- Third quantile/Q3 (`3rd Qu.`), representing the 75th percentile, which splits off the lowest 75% of data from the top 25%
- Maximum (`Max.`), representing the largest value in the data.

`Min` and `Max` can be obtained by using `min()` and `max()` respectively.

The difference between maximum and minimum is known as range. In R, the `range()` function gives us both the minimum and maximum. A combination of `range()` and `diff()` could do the trick of getting the actual range value.

```
range(water$year)

## [1] 1990 2012

diff(range(water$year))

## [1] 22
```

Q1 and Q3 are the 25th and 75th percentiles of the data. Median (Q2) is right in the middle of Q1 and Q3. The difference between Q3 and Q1 is called the interquartile range (IQR). Within the IQR lies half of our data that has no extreme values.

In R, we use the `IQR()` to calculate the interquartile range. If we use `IQR()` for a data with `NA`'s, the `NA`'s are ignored by the function while using the option `na.rm = TRUE`.

```
IQR(water$year)

## [1] 15

summary(water$improved_water)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     3.0    77.0    93.0    84.9    99.0   100.0      32

IQR(water$improved_water, na.rm = T)

## [1] 22
```

Similar to the command `summary()` that we talked about earlier in this Chapter, the function `quantile()` could be used to obtain the five-number summary.

```
quantile(water$improved_water, na.rm = T)

##   0%  25%  50%  75% 100%
##    3   77   93   99  100
```

We can also calculate specific percentiles in the data. For example, if we want the 20th and 60th percentiles, we can do the following.

```
quantile(water$improved_water, probs = c(0.2, 0.6), na.rm = T)

## 20% 60%
##  71  97
```

Using the `seq()` function, we can generate percentiles that are evenly-spaced.

```
quantile(water$improved_water, seq(from=0, to=1, by=0.2), na.rm = T)

##   0%  20%  40%  60%  80% 100%
##    3   71   89   97  100  100
```

Let's reexamine the five-number summary for the `improved_water` variable. When we ignore the `NA`'s, the difference between minimum and Q1 is 74 while the difference between Q3 and maximum is only 1. The interquartile range is 22%. Combining these facts, the first quarter is more widely spread than the middle 50% of values. The last quarter is the most condensed one that has only two percentages: 99% and 100%. Also, we can notice that the mean is smaller than the median. The mean is more sensitive to the extreme values than the median. We have a very small minimum that makes the range of first quantile very large. This extreme value impacts the mean more than the median.

3.7 Visualizing Numeric Variables: Boxplots

We can visualize the five-number summary by a boxplot (box-and-whiskers plot). With the `boxplot()` function we can specify the title (`main = ""`) and labels for x (`xlab = ""`) and y (`ylab = ""`) axes (Fig. 3.3).

```
boxplot(water$improved_water, main="Boxplot for Percent improved_water",
ylab="Percentage")
```

In the boxplot, we have five horizontal lines. Each represents the corresponding value in the five-number summary. The box in the middle represents the middle 50% of values. The bold line in the box is the median. Mean value is not illustrated on the graph.

Boxplots only allow the two ends to extend to a minimum or maximum of 1.5 times the IQR. Therefore, any value that falls outside of the $3 \times IQR$ range will be represented as circles or dots. They are considered as potential outliers. We can see that there are a lot of candidate outliers with small values on the low end of the graph.

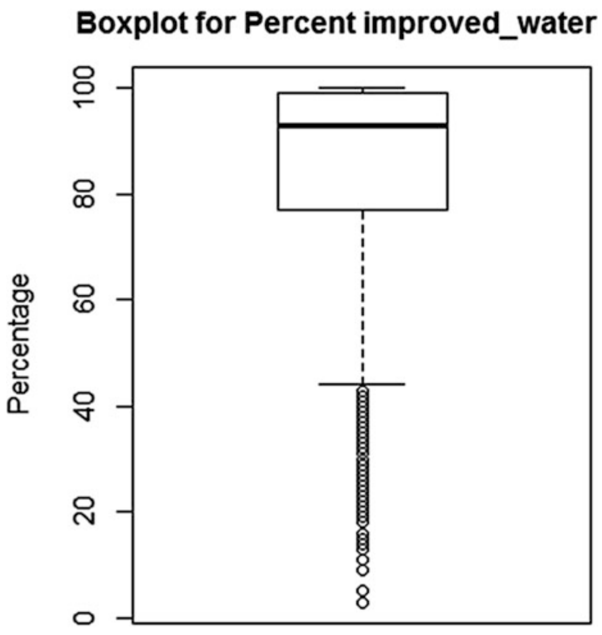

Fig. 3.3 Boxplot of the water improvement variable in the WHO dataset

3.8 Visualizing Numeric Variables: Histograms

A histogram is another way to show the spread of a numeric variable (See Chap. 4 for additional information). It uses predetermined number of bins as containers for values to divide the original data. The height of the bins indicates frequency (Figs. 3.4 and 3.5).

```
hist(water$improved_water, main = "Histogram of  Percent improved_water", xl
ab="Percentage")
```

```
hist(water$sanitation_facilities, main = "Histogram of  Percent sanitation_f
acilities", xlab = "Percentage")
```

We could see that the shape of two graphs are somewhat similar. They are both left skewed patterns ($mean < median$). Other common skew patterns are shown in Fig. 3.6.

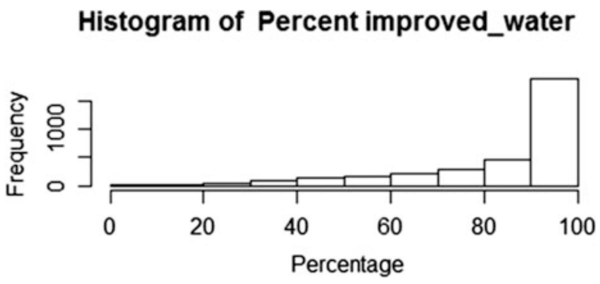

Fig. 3.4 Histogram plot of the water improvement data

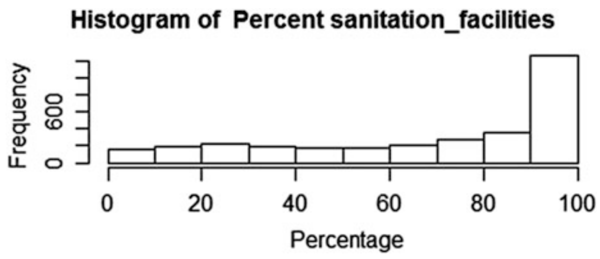

Fig. 3.5 Histogram plot of overall proportion of regions with sanitation facilities (WHO water dataset)

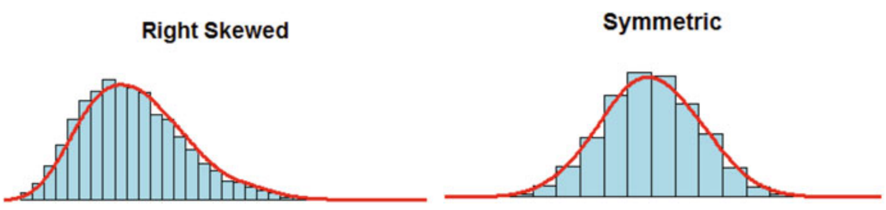

Fig. 3.6 Shape differences between skewed and symmetric distributions

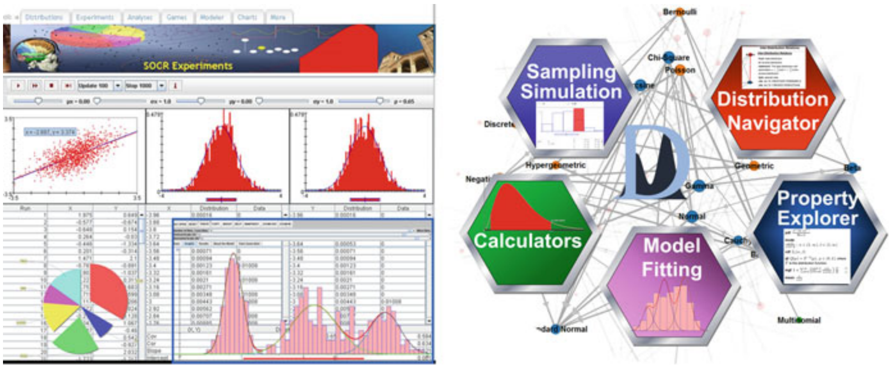

Fig. 3.7 Live visualization demonstrations using SOCR and Distributome resources

You can see the density plots of over 80 different probability distributions using the SOCR Java Distribution Calculators (http://socr.umich.edu/html/dist/) or the Distributome HTML5 Distribution Calculators (http://www.distributome.org/tools.html), Fig. 3.7.

3.9 Understanding Numeric Data: Uniform and Normal Distributions

If the data follows a uniform distribution, then all values are equally likely to occur in any interval of a fixed width. The histogram for a uniformly distributed dataset would have equal heights for each bin, see Fig. 3.9.

```
x <- rnorm(N, 0, 1)
 hist(x, probability=T,
   col='lightblue', xlab=' ', ylab=' ', axes=F,
   main='Normal Distribution')
lines(density(x, bw=0.4), col='red', lwd=3)
```

Often, but not always, real world processes behave as normally distributed. A normal distribution would have a higher frequency for middle values and lower frequency for more extreme values. It has a symmetric and bell-curved shape just like in Fig. 3.8. Many parametric-based statistical approaches assume normality of the data. In cases where this parametric assumption is violated, variable transformations or distribution-free tests may be more appropriate.

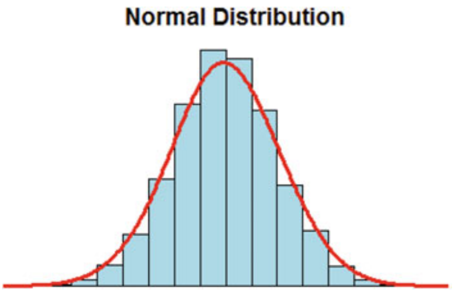

Fig. 3.8 Overlay of a Nornal distribution density (red) and a corresponding Normal sample histogram (blue)

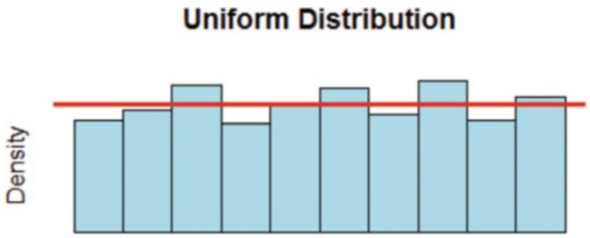

Fig. 3.9 Uniform distribution density and sample histogram plot

3.10 Measuring Spread: Variance and Standard Deviation

Distribution is a great way to characterize data using only a few parameters. For example, normal distribution can be defined by only two parameters: center and spread, or statistically by the mean and standard deviation.

A way to estimate the mean is to divide the sum of the data values by the total number of values. So, we have the following formula:

$$Mean(X) = \mu = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

The variance is the average sum of squares and the standard devision is a square root of the variance:

$$Var(X) = \sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu)^2$$

$$StdDev(X) = \sigma = \sqrt{Var(X)}.$$

Since the water dataset is non-normal, we use a new dataset, including the demographics of baseball players, to illustrate normal distribution properties. The `"01_data.txt"` dataset has following variables:

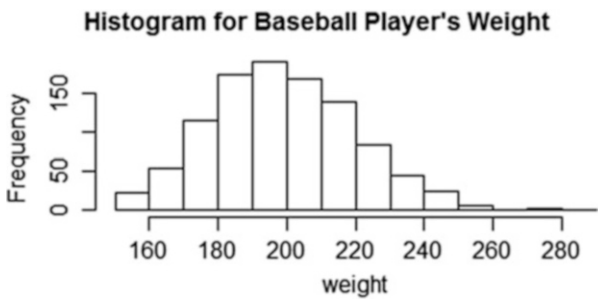

Fig. 3.10 Histogram plot of the players' weights, Major League Baseball (MLB) dataset

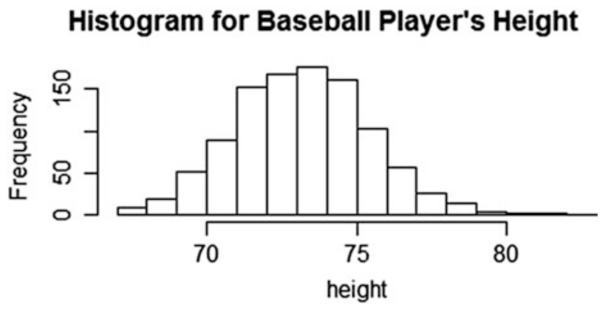

Fig. 3.11 Histogram plot of the players' heights, MLB data

- Name
- Team
- Position
- Height
- Weight
- Age

We can check the histogram for approximate normality of the players' weight and height (Figs. 3.10 and 3.11).

```
baseball<-read.table("https://umich.instructure.com/files/330381/download?do
wnload_frd=1", header=T)
hist(baseball$Weight, main = "Histogram for Baseball Player's Weight", xlab=
"weight")
```

```
hist(baseball$Height, main = "Histogram for Baseball Player's Height", xlab
="height")
```

These plots allow us to visually inspect the normality of the players' heights and weights. We could also obtain mean and standard deviation of the weight and height variables.

```
mean(baseball$Weight)
## [1] 201.7166
mean(baseball$Height)
## [1] 73.69729
var(baseball$Weight)
## [1] 440.9913
sd(baseball$Weight)
## [1] 20.99979
var(baseball$Height)
## [1] 5.316798
sd(baseball$Height)
## [1] 2.305818
```

Larger standard deviation, or variance, suggest the data is more spread out from the mean. Therefore, the weight variable is more spread than the height variable.

Given the first two moments (mean and standard deviation), we can easily estimate how extreme a specific value is. Assuming we have a normal distribution, the values follow a $68 - 95 - 99.7$ rule. This means 68% of the data lies within the interval $[\mu - \sigma, \mu + \sigma]$; 95% of the data lies within the interval $[\mu - 2 * \sigma, \mu + 2 * \sigma]$ and 99.7% of the data lies within the interval $[\mu - 3 * \sigma, \mu + 3 * \sigma]$. The following graph plotted by R illustrates the $68 - 95 - 99.7\%$ rule (Fig. 3.12).

Applying the 68-95-99.7 rule to our baseball weight variable, we know that 68% of our players weighted between 180.7 pounds and 222.7 pounds; 95% of the players weighted between 159.7 pounds and 243.7 pounds; And 99.7% of the players weighted between 138.7 pounds and 264.7 pounds.

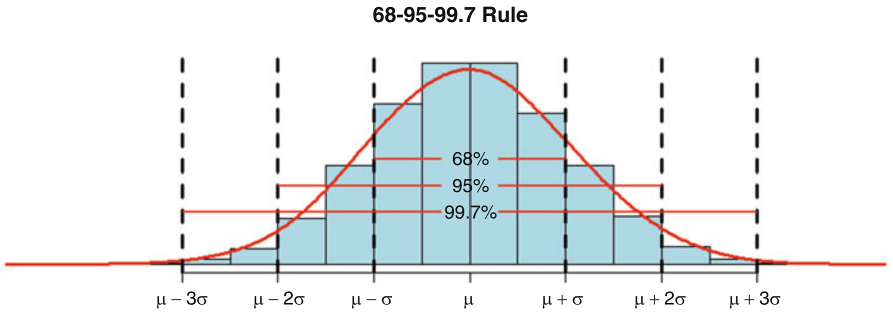

Fig. 3.12 68-95-99.7% rule for Normal distribution

3.11 Exploring Categorical Variables

Back to our water dataset, we can treat the year variable as a categorical rather than a numeric variable. Since the year variable only has six distinctive values, it is reasonable to treat it as a categorical feature, where each value is a category that could apply to multiple WHO regions. Moreover, region and residence area variables are also categorical.

Different from numeric variables, the categorical variables are better examined by tables rather than summary statistics. A one-way tables represent a single categorical variable. It gives us the counts of different categories. The `table()` function can create one-way tables for our water dataset:

```
water <- read.csv('https://umich.instructure.com/files/399172/download?downl
oad_frd=1', header=T)
table(water$Year)

##
## 1990 1995 2000 2005 2010 2012
##  520  561  570  570  556  554

table(water$WHO.region)

##
##                Africa              Americas Eastern Mediterranean
##                   797                   613                   373
##                Europe       South-East Asia       Western Pacific
##                   910                   191                   447

table(water$Residence.Area)

##
## Rural Total Urban
##  1095  1109  1127
```

Given that we have a total of 3331 observations, the WHO region table tells us that about 27% (910/3331) of the areas examined in the study are in Europe.

R can directly give us table proportions when using the `prop.table()` function. The proportion values can be transformed as percentages.

```
year_table<-table(water$Year..string.)
prop.table(year_table)

##
##      1990      1995      2000      2005      2010      2012
## 0.1561093 0.1684179 0.1711198 0.1711198 0.1669168 0.1663164

year_pct<-prop.table(year_table)*100
round(year_pct, digits=1)

##
## 1990 1995 2000 2005 2010 2012
## 15.6 16.8 17.1 17.1 16.7 16.6
```

3.12 Exploring Relationships Between Variables

So far, the methods and statistics that we have seen are univariate. Sometimes, we want to examine the relationship between two or multiple variables. For example, did the percentage of the population that uses improved drinking-water sources increase over time? To address such problems, we need to look at bivariate or multivariate relationships.

Visualizing Relationships: scatterplots
Let's look at a bivariate case first. A scatterplot is a good way to visualize bivariate relationships. We have the x-axis and y-axis each representing one of the variables. Each observation is illustrated on the graph by a dot. If the graph shows a clear pattern, rather than a cluster of random dots, the two variables may be correlated with each other.

In R, we can use the `plot()` function to create scatterplots. We have to define the variables for the x and y-axes. The labels in the graph are editable (Fig. 3.13).

```
plot.window(c(400,1000), c(500,1000))
plot(x=water$year, y=water$improved_water,
     main= "Scatterplot of Year vs. Improved_water",
     xlab= "Year",
     ylab= "Percent of Population Using Improved Water")
```

We can see from the scatterplot that there appears to be a pattern.

Examining Relationships: two-way cross-tabulations
Scatterplot is a useful tool to examine the relationship between two variables where at least one of them is numeric. When both variables are nominal, two-way

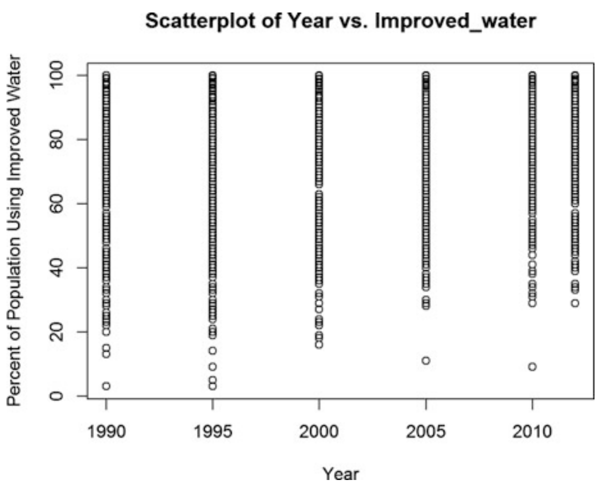

Fig. 3.13 Scatterplot of the percent of world population using improved water quality (WHO dataset)

cross-tabulation would be a better choice (also called crosstab or contingency table).

The function `CrossTable()` is available in R under the package `gmodels`. Let's install it first.

```
#install.packages("gmodels", repos = "http://cran.us.r-project.org")
library(gmodels)
```

We are interested in investigating the relationship between World Health Organization (WHO) region and residence area type in the water study. We might want to know if there is a difference in terms of residence area type between the African WHO region and all other WHO regions.

To address this problem, we need to create an indicator variable for the African WHO region first.

```
water$africa<-water$WHO.region=="Africa"
```

Let's revisit the `table()` function to see how many WHO regions are in Africa.

```
table(water$africa)
## FALSE  TRUE
##  2534   797
```

Now, let's create a two-way cross-tabulation using `CrossTable()`.

```
CrossTable(x=water$Residence.Area, y=water$africa)

##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
## Total Observations in Table:  3331
##
##
##                      | water$africa
## water$Residence.Area |     FALSE |      TRUE | Row Total |
## ---------------------|-----------|-----------|-----------|
##                Rural |       828 |       267 |      1095 |
##                      |     0.030 |     0.096 |           |
##                      |     0.756 |     0.244 |     0.329 |
##                      |     0.327 |     0.335 |           |
##                      |     0.249 |     0.080 |           |
## ---------------------|-----------|-----------|-----------|
##                Total |       845 |       264 |      1109 |
##                      |     0.002 |     0.007 |           |
##                      |     0.762 |     0.238 |     0.333 |
##                      |     0.333 |     0.331 |           |
##                      |     0.254 |     0.079 |           |
## ---------------------|-----------|-----------|-----------|
##                Urban |       861 |       266 |      1127 |
##                      |     0.016 |     0.050 |           |
##                      |     0.764 |     0.236 |     0.338 |
##                      |     0.340 |     0.334 |           |
##                      |     0.258 |     0.080 |           |
## ---------------------|-----------|-----------|-----------|
##         Column Total |      2534 |       797 |      3331 |
##                      |     0.761 |     0.239 |           |
## ---------------------|-----------|-----------|-----------|
```

Each cell in the table contains five numbers. The first one N gives us the count that fall into its corresponding category. The Chi-square contribution yields information about the cell's contribution in the Pearson's Chi-squared test for independence between two variables. This number measures the probability that the differences in cell counts are due to chance alone.

The numbers of interest include `Col Total` and `Row Total`. In this case, these numbers represent the marginal distributions for residence area type among African regions and the regions in the rest of the world. We can see that the numbers are very close between African and non-African regions for each type of residence area. Therefore, we can conclude that African WHO regions do not have a difference in terms of residence area types compared to the rest of the world.

3.13 Missing Data

In the previous sections, we simply ignored the incomplete observations in our water dataset (`na.rm = TRUE`). Is this an appropriate strategy to handle incomplete data? Could the missingness pattern of those incomplete observations be important? It is possible that the arrangement of the missing observations may reflect an important factor that was not accounted for in our statistics or our models.

Missing Completely at Random (MCAR) is an assumption about the probability of missingness being equal for all cases; **Missing at Random (MAR)** assumes the probability of missingness has a known but random mechanism (e.g., different rates for different groups); **Missing not at Random (MNAR)** suggest a missingness mechanism linked to the values of predictors and/or response, e.g., some participants may drop out of a drug trial when they have side-effects.

There are a number of strategies to impute missing data. The expectation maximization (EM) algorithm provides one example for handling missing data. The SOCR EM tutorial, activity, and documentations provide the theory, applications and practice for effective (multidimensional) EM parameter estimation.

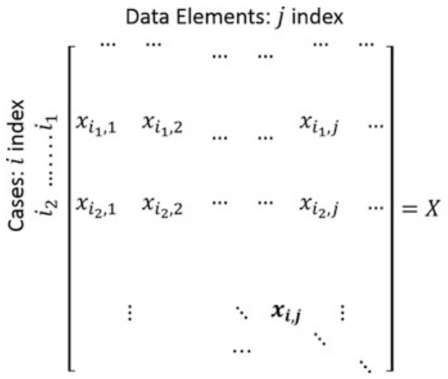

Fig. 3.14 Schematic data representation indexing data values by case (rows) and feature (columns)

The simplest way to handle incomplete data is to substitute each missing value with its (feature or column) average. When the missingness proportion is small, the effect of substituting the means for the missing values will have little effect on the mean, variance, or other important statistics of the data. Also, this will preserve those non-missing values of the same observation or row, see Fig. 3.14.

```
m1<-mean(water$Population.using.improved.drinking, na.rm = T)
m2<-mean(water$Population.using.improved.sanitation, na.rm = T)
water_imp<-water
for(i in 1:3331){
  if(is.na(water_imp$Population.using.improved.drinking[i])){
    water_imp$Population.using.improved.drinking[i]=m1
  }
  if(is.na(water_imp$Population.using.improved.sanitation[i])){
    water_imp$Population.using.improved.sanitation=m2
  }
}
summary(water_imp)
```

```
##   Year..string.                WHO.region..string.           Country..string.
##   Min.   :1990   Africa                :797      Albania            :  18
##   1st Qu.:1995   Americas              :613      Algeria            :  18
##   Median :2005   Eastern Mediterranean:373      Andorra            :  18
##   Mean   :2002   Europe                :910      Angola             :  18
##   3rd Qu.:2010   South-East Asia       :191      Antigua and Barbuda:  18
##   Max.   :2012   Western Pacific       :447      Argentina          :  18
##                                                  (Other)            :3223
##   Residence.Area.Type..string.
##   Rural:1095
##   Total:1109
##   Urban:1127
##   Population.using.improved.drinking.water.sources......numeric.
##   Min.   :  3.0
##   1st Qu.: 77.0
##   Median : 93.0
##   Mean   : 84.9
##   3rd Qu.: 99.0
##   Max.   :100.0
##   NA's   :32
##  Population.using.improved.sanitation.facilities......numeric.
##  Min.    :  0.00
##  1st Qu.: 42.00
##  Median : 81.00
##  Mean   : 68.87
##  3rd Qu.: 97.00
##  Max.   :100.00
##  NA's   :135
##    africa         Population.using.improved.sanitation
##  Mode :logical   Min.   :68.87
##  FALSE:2534      1st Qu.:68.87
##  TRUE :797       Median :68.87
##  NA's :0         Mean   :68.87
##                  3rd Qu.:68.87
##                  Max.   :68.87
##
##  Population.using.improved.drinking
##  Min.   :  3.0
##  1st Qu.: 77.0
##  Median : 93.0
##  Mean   : 84.9
##  3rd Qu.: 99.0
##  Max.   :100.0
```

A more sophisticated way of resolving missing data is to use a model (e.g., linear regression) to predict the missing feature and impute its missing values. This is called the `predictive mean matching approach`. This method is good for data with multivariate normality. However, a disadvantage of it is that it can only predict one value at a time, which is very time consuming. Also, the multivariate normality assumption might not be satisfied and there may be important multivariate relations that are not accounted for. We are using the `mi` package to demonstrate predictive mean matching.

Let's install the `mi` package first.

```
# install.packages("mi", repos = "http://cran.us.r-project.org")
library(mi)
```

Then, we need to get the missing information matrix. We are using the imputation method `pmm` (predictive mean matching approach) for both missing variables.

```
mdf<-missing_data.frame(water)
head(mdf)

##    Year..string. WHO.region..string. Country..string.
## 1           1990              Africa          Algeria
## 2           1990              Africa           Angola
## 3           1990              Africa            Benin
## 4           1990              Africa         Botswana
## 5           1990              Africa     Burkina Faso
## 6           1990              Africa          Burundi
##   Residence.Area.Type..string.
## 1                        Rural
## 2                        Rural
## 3                        Rural
## 4                        Rural
## 5                        Rural
## 6                        Rural
##
##   Population.using.improved.drinking.water.sources......numeric.
## 1                                                             88
## 2                                                             42
## 3                                                             49
## 4                                                             86
## 5                                                             39
## 6                                                             67
##   Population.using.improved.sanitation.facilities......numeric. africa
## 1                                                            77   TRUE
## 2                                                             7   TRUE
## 3                                                             0   TRUE
## 4                                                            22   TRUE
## 5                                                             2   TRUE
## 6                                                            42   TRUE
##   missing_Population.using.improved.drinking.water.sources......numeric.
## 1                                                                  FALSE
## 2                                                                  FALSE
## 3                                                                  FALSE
## 4                                                                  FALSE
## 5                                                                  FALSE
## 6                                                                  FALSE
##   missing_Population.using.improved.sanitation.facilities......numeric.
## 1                                                                 FALSE
## 2                                                                 FALSE
## 3                                                                 FALSE
## 4                                                                 FALSE
## 5                                                                 FALSE
## 6                                                                 FALSE
```

```
show(mdf)
```

```
## Object of class missing_data.frame with 3331 observations on 7 variables
##
## There are 3 missing data patterns
##
## Append '@patterns' to this missing_data.frame to access the corresponding
pattern for every observation or perhaps use table()
##
##
type
## Year..string.
continuous
## WHO.region..string.                                  unordered-categorical
## Country..string.                                     unordered-categorical
## Residence.Area.Type..string.                         unordered-categorical
## Population.using.improved.drinking.water.sources......numeric.
continuous
## Population.using.improved.sanitation.facilities......numeric.
continuous
## africa
binary
```

```
##                                                                  missing
## Year..string.                                                          0
## WHO.region..string.                                                    0
## Country..string.                                                       0
## Residence.Area.Type..string.                                           0
## Population.using.improved.drinking.water.sources......numeric.        32
## Population.using.improved.sanitation.facilities......numeric.        135
## africa                                                                 0
##                                                                 method
## Year..string.                                                     <NA>
…
## africa
<NA>

mdf<-change(mdf, y="Population.using.improved.drinking", what = "imputation_
method", to="pmm")
mdf<-change(mdf, y="Population.using.improved.sanitation", what = "imputatio
n_method", to="pmm")
```

Notes

- Converting the input `data.frame` to a `missing_data.frame` allows us to include in the DF enhanced metadata about each variable, which is essential for the subsequent modeling, interpretation, and imputation of the initial missing data.
- `show()` displays all missing variables and their class-labels (e.g., continuous), along with meta-data. The `missing_data.frame` constructor suggests the most appropriate classes for each missing variable; however, the user often needs to correct, modify, or change these meta-data, using `change()`.
- Use the `change()` function to change/correct meta-data in the constructed `missing_data.frame` object which may be incorrectly reported by `show (mfd)`.
- To get a sense of the raw data, look at the `summary`, `image`, or `hist` of the missing_data.frame.
- The mi vignettes provide many useful examples of handling missing data.

Next, we can perform the initial imputation. Here we imputed three times, which will create three different datasets with slightly different imputed values.

```
imputations<-mi(mdf, n.iter=10, n.chains=3, verbose=T)
```

Next, we need to extract several multiply imputed `data.frames` from `imputations` object. Finally, we can compare the summary statistics between the original dataset and the imputed datasets.

```
data.frames <- complete(imputations, 3)
summary(water)

##   Year..string.             WHO.region..string.           Country..string
.
##   Min.   :1990   Africa                :797      Albania             :  18
##   1st Qu.:1995   Americas              :613      Algeria             :  18
##   Median :2005   Eastern Mediterranean:373      Andorra             :  18
##   Mean   :2002   Europe                :910      Angola              :  18
##  3rd Qu.:2010   South-East Asia      :191      Antigua and Barbuda:   18
##  Max.   :2012   Western Pacific      :447      Argentina          :   18
##                                                (Other)            :3223
##  Residence.Area.Type..string.
##  Rural:1095
##  Total:1109
##  Urban:1127
…
##  missing_Population.using.improved.sanitation.facilities......numeric.
##  Mode :logical
##  FALSE:3196
##  TRUE :135
##  NA's :0
```

This is just a brief introduction for handling incomplete datasets. In later Chapters, we will discuss more about missing data with different imputation methods and how to evaluate the complete imputed results.

3.13.1 Simulate Some Real Multivariate Data

Suppose we would like to generate a synthetic dataset:

$$sim\_data = \{y, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10\}.$$

Then, we can introduce a method that takes a dataset and a desired proportion of missingness and wipes out the same proportion of the data, i.e., introduces random patterns of missingness. Note that there are already R functions that automate the introduction of missingness, e.g., `missForest::prodNA()`; however, writing such method from scratch is also useful. Figure 3.15 shows the results of introducing 30% missingness in the simulated data.

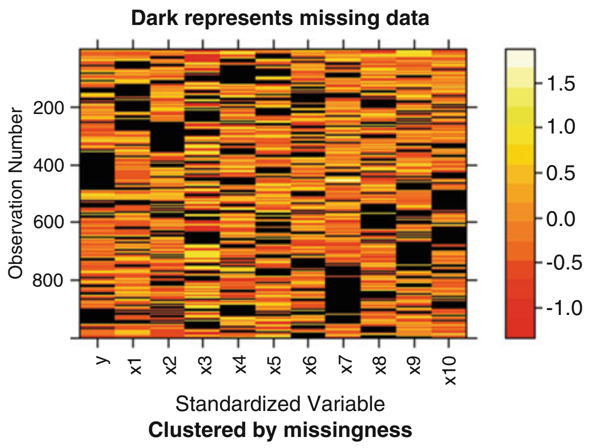

Fig. 3.15 Incomplete data image plot illustrating the pattern of data missingness

```r
set.seed(123)
# create MCAR missing-data generator
create.missing <- function (data, pct.mis = 10)
{
    n <- nrow(data)
    J <- ncol(data)
    if (length(pct.mis) == 1) {
        if(pct.mis>= 0 & pct.mis <=100) {
            n.mis <- rep((n * (pct.mis/100)), J)
        }
        else {
          warning("Percent missing values should be an integer between
             0 and 100! Exiting"); break
        }
      }
    else {
        if (length(pct.mis) < J)
            stop("The length of the missing-vector is not equal to the numbe
r of columns in the data! Exiting!")
            n.mis <- n * (pct.mis/100)
    }
    for (i in 1:ncol(data)) {
        if (n.mis[i] == 0) { # if column has no missing do nothing.
            data[, i] <- data[, i]
        }
        else {
            data[sample(1:n, n.mis[i], replace = FALSE), i] <- NA
              # For each given column (i), sample the row indices (1:n),
              # a number of indices to replace as "missing", n.mis[i], "NA",
              # without replacement
        }
    }
    return(as.data.frame(data))
}
```

Next, let's synthetically generate (simulate) 1,000 cases including all 11 features in the data ($\{y, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10\}$).

```r
n <- 1000; u1 <- rbinom(n, 1, .5); v1<-log(rnorm(n, 5, 1)); x1 <- u1*exp(v1)
u2 <- rbinom(n, 1, .5); v2 <- log(rnorm(n, 5, 1)); x2 <- u2*exp(v2)
x3 <- rbinom(n,1,prob=0.45); x4<-ordered(rep(seq(1, 5), n)[sample(1:n, n)]);
x5 <- rep(letters[1:10], n)[sample(1:n, n)]; x6 <- trunc(runif(n, 1, 10));
x7 <- rnorm(n); x8 <- factor(rep(seq(1, 10), n)[sample(1:n, n)]);
x9 <- runif(n,0.1,0.99); x10 <- rpois(n, 4); y<-x1 + x2 + x7 + x9 + rnorm(n)

# package the simulated data as a data frame object
sim_data <- cbind.data.frame(y, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10)

# randomly create missing values
sim_data_30pct_missing <- create.missing(sim_data, pct.mis=30);
head(sim_data_30pct_missing); summary(sim_data_30pct_missing)
```

```
##           y       x1       x2 x3   x4   x5 x6         x7   x8        x9
## 1        NA       NA 0.000000  0    1    h  8         NA    3        NA
## 2 11.449223       NA 5.236938  0    1    i NA         NA   10 0.2639489
## 3 -1.188296 0.000000 0.000000  0    5    a  3 -1.1469495 <NA> 0.4753195
## 4        NA       NA       NA  0 <NA>    e  6  1.4810186   10 0.6696932
## 5  4.267916 3.490833 0.000000  0 <NA> <NA> NA  0.9161912 <NA> 0.9578455
## 6        NA 0.000000 4.384732  1 <NA>    a NA         NA   10 0.6095176
##   x10
## 1   1
## 2   2
## 3  NA
## 4   3
## 5   8
## 6   6

##        y                 x1               x2               x3
##  Min.   :-3.846   Min.   :0.000   Min.   :0.000   Min.   :0.0000
##  1st Qu.: 2.410   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.0000
##  Median : 5.646   Median :0.000   Median :3.068   Median :0.0000
##  Mean   : 5.560   Mean   :2.473   Mean   :2.545   Mean   :0.4443
##  3rd Qu.: 8.503   3rd Qu.:4.958   3rd Qu.:4.969   3rd Qu.:1.0000
##  Max.   :16.487   Max.   :8.390   Max.   :8.421   Max.   :1.0000
##  NA's   :300      NA's   :300     NA's   :300     NA's   :300
##     x4             x5             x6              x7                 x8
##  1   :138   c       : 80   Min.   :1.00   Min.   :-2.5689   3       : 78
##  2   :129   h       : 76   1st Qu.:3.00   1st Qu.:-0.6099   7       : 77
##  3   :147   b       : 74   Median :5.00   Median : 0.0202   5       : 75
##  4   :144   a       : 73   Mean   :4.93   Mean   : 0.0435   4       : 73
##  5   :142   j       : 72   3rd Qu.:7.00   3rd Qu.: 0.7519   1       : 70
##  NA's:300   (Other):325   Max.   :9.00   Max.   : 3.7157   (Other):327
##             NA's   :300   NA's   :300    NA's   :300       NA's   :300
##        x9               x10
##  Min.   :0.1001   Min.   : 0.000
##  1st Qu.:0.3206   1st Qu.: 2.000
##  Median :0.5312   Median : 4.000
##  Mean   :0.5416   Mean   : 3.929
##  3rd Qu.:0.7772   3rd Qu.: 5.000
##  Max.   :0.9895   Max.   :11.000
##  NA's   :300      NA's   :300
```

```r
# install.packages("mi")
# install.packages("betareg")
library("betareg"); library("mi")

# get show the missing information matrix
mdf <- missing_data.frame(sim_data_30pct_missing)
show(mdf)
```

```
## Object of class missing_data.frame with 1000 observations on 11 variables
##
## There are 542 missing data patterns
##
## Append '@patterns' to this missing_data.frame to access the corresponding
pattern for every observation or perhaps use table()
##
##                         type missing method  model
## y                 continuous     300    ppd  linear
## x1              continuous     300    ppd  linear
## x2              continuous     300    ppd  linear
## x3                  binary     300    ppd   logit
## x4      ordered-categorical     300    ppd  ologit
## x5    unordered-categorical     300    ppd  mlogit
## x6              continuous     300    ppd  linear
## x7              continuous     300    ppd  linear
## x8    unordered-categorical     300    ppd  mlogit
## x9              proportion     300    ppd betareg
## x10             continuous     300    ppd  linear
##
##           family     link transformation
## y       gaussian identity    standardize
## x1      gaussian identity    standardize
## x2      gaussian identity    standardize
## x3      binomial    logit           <NA>
## x4   multinomial    logit           <NA>
## x5   multinomial    logit           <NA>
## x6      gaussian identity    standardize
## x7      gaussian identity    standardize
## x8   multinomial    logit           <NA>
## x9      binomial    logit       identity
## x10     gaussian identity    standardize

# mdf@patterns   # to get the textual missing pattern
image(mdf)   # remember the visual pattern of this MCAR
```

The histogram plots display the distributions of:

- The **observed** data (in **blue color**),
- The **imputed** data (in **red color**), and
- The **completed** values (observed plus imputed, in **gray color**) (Figs. 3.16, 3.17 and 3.18).

```
# Next, try to impute the missing values.

# Get the Graph Parameters (plotting canvas/margins)
# set to plot the histograms for the 3 imputation chains
# mfcol=c(nr, nc). Subsequent histograms are drawn as nr-by-nc arrays on
# the graphics device by columns (mfcol), or rows (mfrow)
# oma: oma=c(bottom, left, top, right) giving the size of the outer
# margins in lines of text
# mar=c(bottom, left, top, right) gives the number of lines of margin
# to be specified on the four sides of the plot.
# tcl=length of tick marks as a fraction of the height of a line of
# text (default=0.5)
par(mfcol=c(5, 5), oma=c(1, 1, 0, 0), mar=c(1, 1, 1, 0), tcl=-0.1,
mgp=c(0, 0, 0))
imputations <- mi(sim_data_30pct_missing, n.iter=5, n.chains=3,verbose=TRUE)
hist(imputations)
```

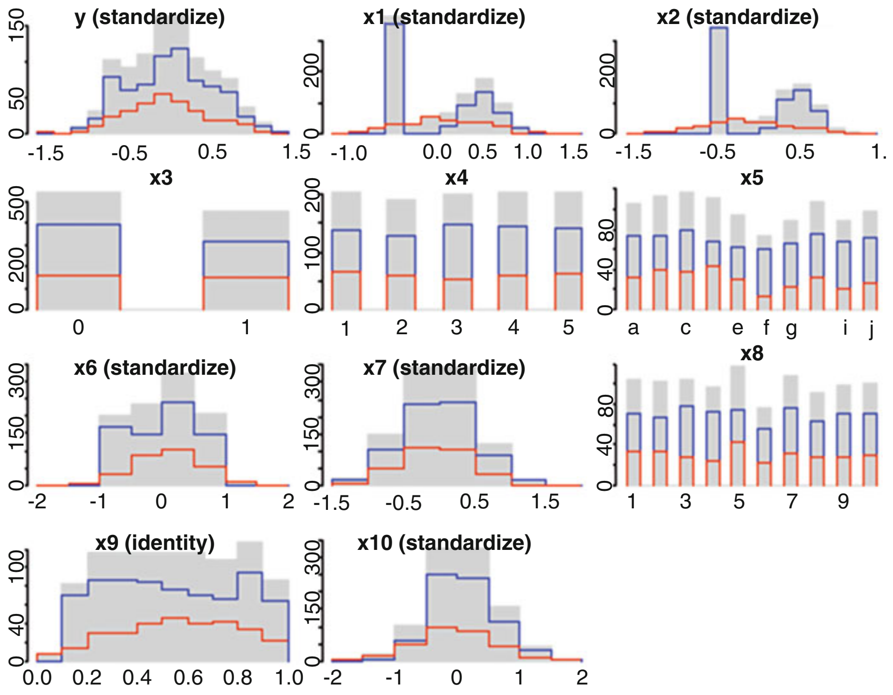

Fig. 3.16 Imputation chain 1: Histogram plots comparing the initially observed (blue), imputed (red), and imputed complete (gray) data

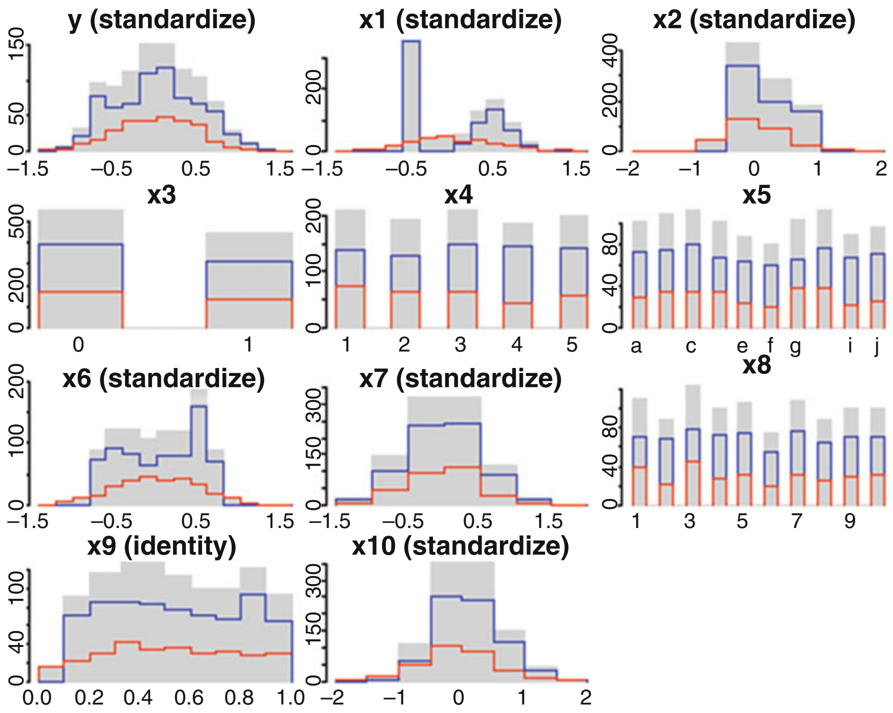

Fig. 3.17 Imputation chain 2: Histogram plots comparing the initially observed (blue), imputed (red), and imputed complete (gray) data

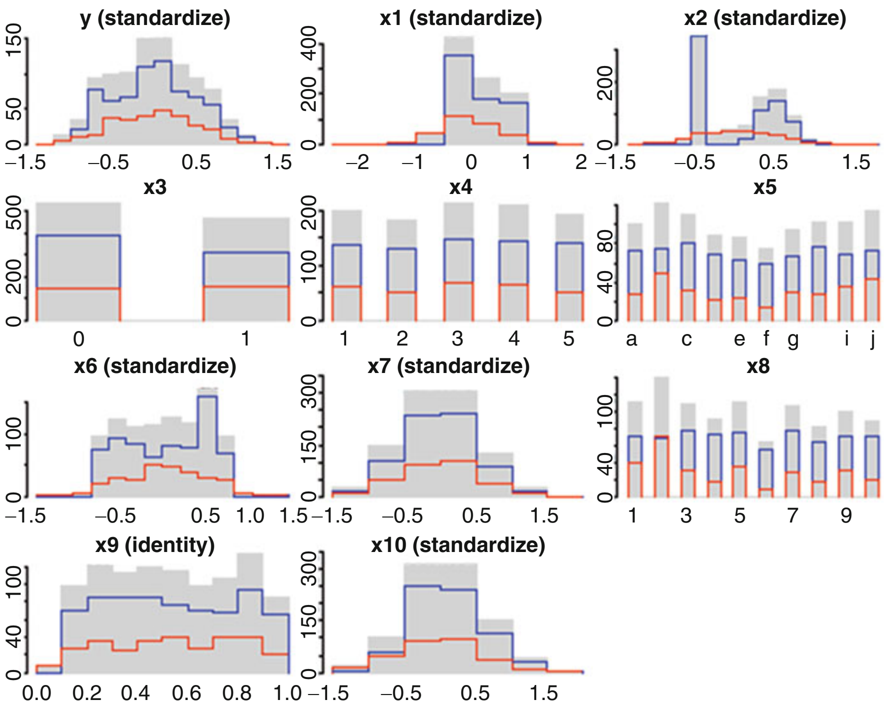

Fig. 3.18 Imputation chain 3: Histogram plots comparing the initially observed (blue), imputed (red), and imputed complete (gray) data

```
# Extracts several multiply imputed data.frames from "imputations" object
data.frames <- complete(imputations, 3)

# Compare the summaries for the original data (prior to introducing missing
# values) with missing data and the re-completed data following imputation
summary(sim_data);summary(sim_data_30pct_missing);summary(data.frames[[1]]);

##         y                x1               x2               x3         x4
##  Min.   :-3.846   Min.   :0.000   Min.   :0.000   Min.   :0.000   1:200
##  1st Qu.: 2.489   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000   2:200
##  Median : 5.549   Median :0.000   Median :2.687   Median :0.000   3:200
##  Mean   : 5.562   Mean   :2.472   Mean   :2.516   Mean   :0.431   4:200
##  3rd Qu.: 8.325   3rd Qu.:4.996   3rd Qu.:5.007   3rd Qu.:1.000   5:200
##  Max.   :16.487   Max.   :8.390   Max.   :8.421   Max.   :1.000
##
…

##         y                x1               x2               x3
##  Min.   :-3.846   Min.   :0.000   Min.   :0.000   Min.   :0.0000
##  1st Qu.: 2.410   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.0000
##  Median : 5.646   Median :0.000   Median :3.068   Median :0.0000
##  Mean   : 5.560   Mean   :2.473   Mean   :2.545   Mean   :0.4443
##  3rd Qu.: 8.503   3rd Qu.:4.958   3rd Qu.:4.969   3rd Qu.:1.0000
##  Max.   :16.487   Max.   :8.390   Max.   :8.421   Max.   :1.0000
##  NA's   :300      NA's   :300     NA's   :300     NA's   :300
…
```

```
##   missing_x10
##   Mode :logical
##   FALSE:700
##   TRUE :300
##   NA's :0
lapply(data.frames, summary)
```

```
## $`chain:1`
##         y                 x1                x2            x3       x4
##   Min.   :-6.852   Min.   :-3.697   Min.   :-4.920   0:545   1:203
##   1st Qu.: 2.475   1st Qu.: 0.000   1st Qu.: 0.000   1:455   2:189
##   Median : 5.470   Median : 2.510   Median : 1.801           3:201
##   Mean   : 5.458   Mean   : 2.556   Mean   : 2.314           4:202
##   3rd Qu.: 8.355   3rd Qu.: 4.892   3rd Qu.: 4.777           5:205
##   Max.   :16.487   Max.   :10.543   Max.   : 8.864
##
…
##   missing_x10
##   Mode :logical
##   FALSE:700
##   TRUE :300
##   NA's :0
##
## $`chain:2`
##         y                 x1                x2            x3       x4
##   Min.   :-4.724   Min.   :-4.744   Min.   :-5.740   0:558   1:211
##   1st Qu.: 2.587   1st Qu.: 0.000   1st Qu.: 0.000   1:442   2:193
##   Median : 5.669   Median : 2.282   Median : 2.135           3:211
##   Mean   : 5.528   Mean   : 2.486   Mean   : 2.452           4:187
##   3rd Qu.: 8.367   3rd Qu.: 4.884   3rd Qu.: 4.782           5:198
##   Max.   :17.054   Max.   :10.445   Max.   :10.932
…
## $`chain:3`
##         y                 x1                x2            x3       x4
##   Min.   :-5.132   Min.   :-8.769   Min.   :-3.643   0:538   1:200
##   1st Qu.: 2.414   1st Qu.: 0.000   1st Qu.: 0.000   1:462   2:182
##   Median : 5.632   Median : 2.034   Median : 2.610           3:215
##   Mean   : 5.537   Mean   : 2.417   Mean   : 2.530           4:211
##   3rd Qu.: 8.434   3rd Qu.: 4.836   3rd Qu.: 4.812           5:192
##   Max.   :16.945   Max.   :10.335   Max.   :11.683
…
##   missing_x10
##   Mode :logical
##   FALSE:700
##   TRUE :300
##   NA's :0
```

Let’s check the imputation convergence (details provided below) (Figs. 3.19 and 3.20).

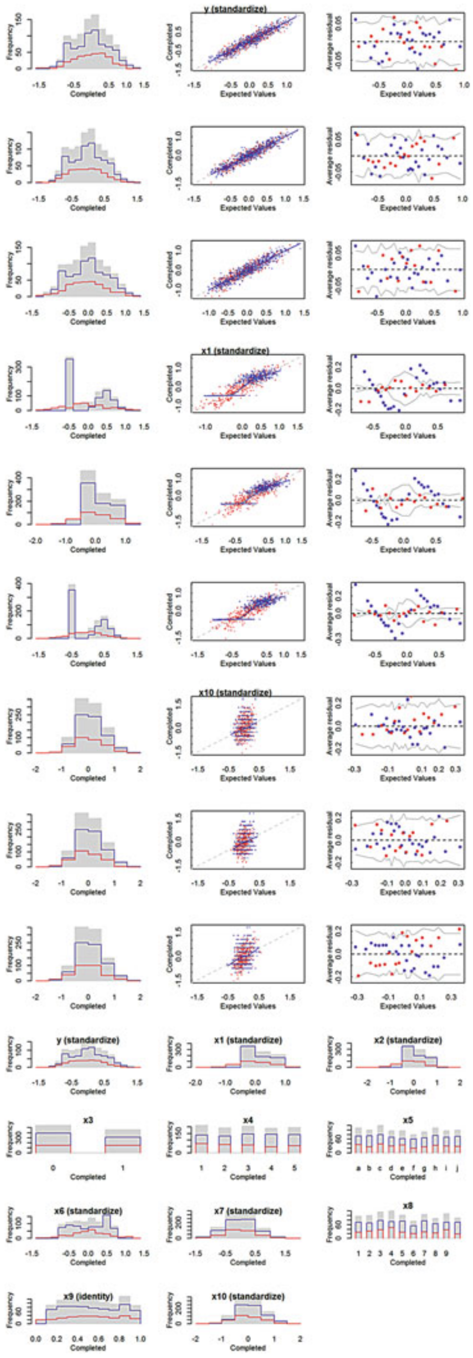

Fig. 3.19 Plots of the imputation iterations for the simulated dataset

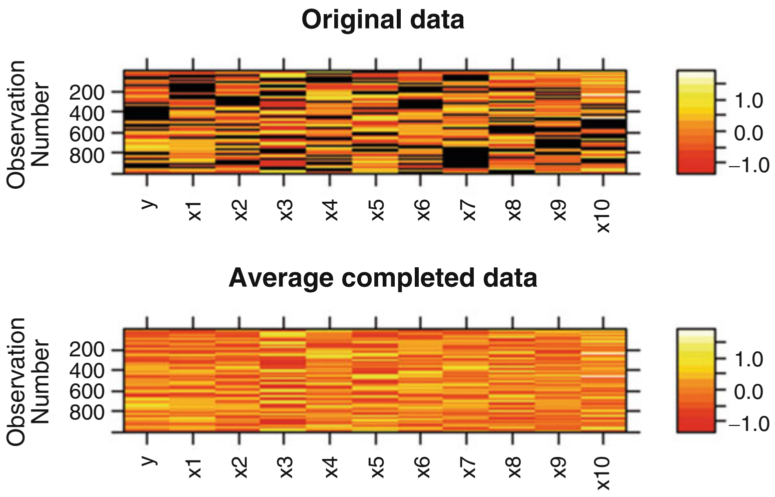

Fig. 3.20 Comparison of the missingness patterns in the raw (top) and imputed (bottom) datasets

```
round(mipply(imputations, mean, to.matrix = TRUE), 3)

##              chain:1 chain:2 chain:3
## y             -0.013  -0.004  -0.003
## x1             0.016   0.003  -0.011
## x2            -0.045  -0.018  -0.003
## x3             1.455   1.442   1.462
## x4             3.017   2.968   3.013
## x5             5.321   5.406   5.480
## x6             0.023   0.004   0.005
## x7            -0.015  -0.005  -0.006
## x8             5.431   5.409   5.202
## x9             0.548   0.536   0.541
## x10           -0.015  -0.020  -0.009
## missing_y      0.300   0.300   0.300
## missing_x1     0.300   0.300   0.300
## missing_x2     0.300   0.300   0.300
## missing_x3     0.300   0.300   0.300
## missing_x4     0.300   0.300   0.300
## missing_x5     0.300   0.300   0.300
## missing_x6     0.300   0.300   0.300
## missing_x7     0.300   0.300   0.300
## missing_x8     0.300   0.300   0.300
## missing_x9     0.300   0.300   0.300
## missing_x10    0.300   0.300   0.300

Rhats(imputations, statistic = "moments")
# assess the convergence of MI algorithm

##    mean_y   mean_x1   mean_x2   mean_x3   mean_x4   mean_x5   mean_x6
## 1.0235026 1.1125720 1.1565542 0.9460979 1.0543446 1.3207898 0.9855947
##   mean_x7   mean_x8   mean_x9  mean_x10      sd_y     sd_x1     sd_x2
## 1.0023935 0.9438358 1.0192697 0.9927675 0.9658852 1.6248062 1.0025950
##     sd_x3     sd_x4     sd_x5     sd_x6     sd_x7     sd_x8     sd_x9
## 0.9463044 1.0706666 1.4470270 1.2510790 0.9008732 1.2865944 1.0195947
##    sd_x10
## 1.1760195

plot(imputations);hist(imputations);image(imputations);summary(imputations)
```

```
## $y
## $y$is_missing
## missing
## FALSE  TRUE
##   700   300
##
## $y$imputed
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -1.55100 -0.36930 -0.01107 -0.02191  0.30080  1.43600
##
## $y$observed
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -1.17500 -0.39350  0.01069  0.00000  0.36770  1.36500
##
## $x1
## $x1$is_missing
## missing
## FALSE  TRUE
##   700   300
##
## $x1$imputed
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -2.168000 -0.353600 -0.023620  0.008851  0.379800  1.556000
##
## $x1$observed
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.4768 -0.4768 -0.4768  0.0000  0.4793  1.1410
…
## $x10$observed
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -1.01800 -0.49980  0.01851  0.00000  0.27760  1.83200
```

Finally, pool over the $m = 3$ completed datasets when we fit the "model". In order to estimate a linear regression model, we pool from across the three chains. Figure 3.21 shows the distribution of a simple bivariate linear model (y = x1 + x2).

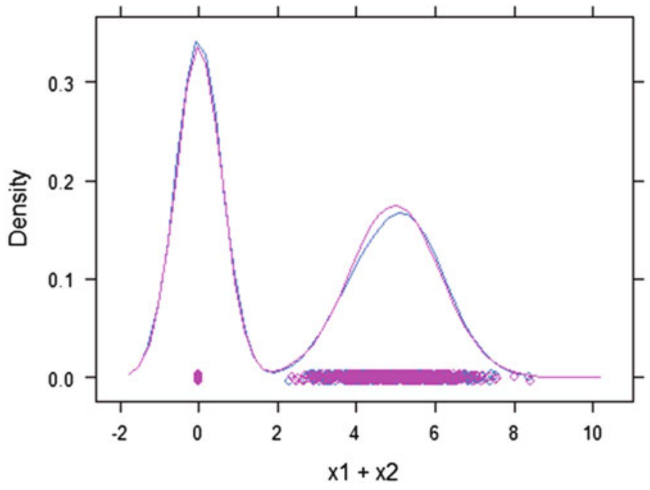

Fig. 3.21 Density plots comparing the observed and imputed outcome variable *y*

```
model_results<-pool(y~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=imputations,  m=3)
display (model_results); summary (model_results)
```

```
## bayesglm(formula = y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 +
##     x9 + x10, data = imputations, m = 3)
##             coef.est coef.se
## (Intercept)  0.77     0.84
## x1           0.94     0.05
## x2           0.97     0.04
## x31         -0.27     0.37
## x4.L         0.21     0.21
## x4.Q        -0.09     0.16
## x4.C         0.03     0.24
## x4^4         0.25     0.20
## x5b          0.03     0.42
## x5c         -0.41     0.26
## x5d         -0.22     0.86
## x5e          0.11     0.56
## x5f         -0.13     0.55
## x5g         -0.27     0.67
## x5h         -0.17     0.66
## x5i         -0.69     0.81
## x5j          0.21     0.28
## x6          -0.04     0.07
## x7           0.98     0.09
## x82          0.44     0.39
## x83          0.40     0.20
## x84         -0.14     0.62
## x85          0.20     0.30
## x86          0.19     0.25
## x87          0.19     0.38
## x88          0.51     0.34
## x89          0.25     0.26
## x810         0.17     0.48
## x9           0.88     0.71
## x10         -0.06     0.05
## n = 970, k = 30
## residual deviance = 2056.5, null deviance = 15851.5 (difference=13795.0)
## overdispersion parameter = 2.1
## residual sd is sqrt(overdispersion) = 1.46

## Call:
## pool(formula = y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 +
##     x10, data = imputations, m = 3)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8821  -0.6925  -0.0005   0.6859   3.7035
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.76906    0.83558   0.920 0.440149
## x1           0.94250    0.04535  20.781 0.000388 ***
## x2           0.97495    0.03517  27.721 2.01e-05 ***
## x31         -0.27349    0.37377  -0.732 0.533696
## x4.L         0.21116    0.21051   1.003 0.378488
## x4.Q        -0.08567    0.15627  -0.548 0.602349
## x4.C         0.02957    0.24490   0.121 0.911557
## x4^4         0.24987    0.19504   1.281 0.271639
## x5b          0.03327    0.41563   0.080 0.940649
## x5c         -0.41124    0.25525  -1.611 0.129304
## x5d         -0.21576    0.86290  -0.250 0.824194
## x5e          0.11334    0.56396   0.201 0.854842
## x5f         -0.13162    0.55187  -0.238 0.827734
## x5g         -0.27014    0.67022  -0.403 0.719913
## x5h         -0.16951    0.66294  -0.256 0.818576
## x5i         -0.68619    0.80975  -0.847 0.477639
## x5j          0.20681    0.27823   0.743 0.473891
## x6          -0.04009    0.07306  -0.549 0.633836
## x7           0.98130    0.08527  11.508 0.000197 ***
## x82          0.43774    0.38574   1.135 0.322775
## x83          0.40307    0.20475   1.969 0.049445 *
## x84         -0.13651    0.62307  -0.219 0.843284
## x85          0.19905    0.29973   0.664 0.528335
## x86          0.18662    0.24702   0.755 0.452036
## x87          0.18792    0.38029   0.494 0.647992
## x88          0.51106    0.34272   1.491 0.192478
## x89          0.25125    0.26340   0.954 0.356132
## x810         0.17383    0.47841   0.363 0.740434
## x9           0.87514    0.71484   1.224 0.334593
## x10         -0.05722    0.05035  -1.136 0.331688
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 2.120095)
##
##     Null deviance: 15851.5  on 999  degrees of freedom
## Residual deviance:  2056.5  on 970  degrees of freedom
## AIC: 3616.9
## Number of Fisher Scoring iterations: 7

# Report the summaries of the imputations
data.frames <- complete(imputations, 3)     # extract the first 3 chains
lapply(data.frames, summary)

## $`chain:1`
##        y                 x1                x2           x3      x4
##  Min.   :-6.852   Min.   :-3.697   Min.   :-4.920   0:545   1:203
##  1st Qu.: 2.475   1st Qu.: 0.000   1st Qu.: 0.000   1:455   2:189
##  Median : 5.470   Median : 2.510   Median : 1.801           3:201
##  Mean   : 5.458   Mean   : 2.556   Mean   : 2.314           4:202
##  3rd Qu.: 8.355   3rd Qu.: 4.892   3rd Qu.: 4.777           5:205
##  Max.   :16.487   Max.   :10.543   Max.   : 8.864
##
##        x5            x6                x7                  x8
##  c      :118   Min.   :-4.291   Min.   :-2.73138   5      :117
##  b      :113   1st Qu.: 3.000   1st Qu.:-0.61765   7      :109
##  d      :111   Median : 5.000   Median : 0.00085   3      :105
##  h      :108   Mean   : 5.051   Mean   : 0.01486   1      :104
##  a      :105   3rd Qu.: 7.000   3rd Qu.: 0.71796   2      :102
##  j      : 99   Max.   :13.284   Max.   : 3.71572   10     :100
##  (Other):346                                       (Other):363
##        x9               x10          missing_y        missing_x1
##  Min.   :0.0073   Min.   :-1.930   Mode :logical   Mode :logical
##  1st Qu.:0.3383   1st Qu.: 2.115   FALSE:700       FALSE:700
##  Median :0.5417   Median : 4.000   TRUE :300       TRUE :300
##  Mean   :0.5476   Mean   : 3.870   NA's :0         NA's :0
##  3rd Qu.:0.7635   3rd Qu.: 5.000
##  Max.   :0.9975   Max.   :11.000
##
##  missing_x2      missing_x3      missing_x4      missing_x5
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical
##  FALSE:700       FALSE:700       FALSE:700       FALSE:700
##  TRUE :300       TRUE :300       TRUE :300       TRUE :300
##  NA's :0         NA's :0         NA's :0         NA's :0
##
##  missing_x6      missing_x7      missing_x8      missing_x9
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical
##  FALSE:700       FALSE:700       FALSE:700       FALSE:700
##  TRUE :300       TRUE :300       TRUE :300       TRUE :300
##  NA's :0         NA's :0         NA's :0         NA's :0
##
##  missing_x10
##  Mode :logical
##  FALSE:700
##  TRUE :300
##  NA's :0
##
## $`chain:2`
##        y                x1                x2          x3      x4
##  Min.   :-4.724   Min.   :-4.744   Min.   :-5.740   0:558   1:211
##  1st Qu.: 2.587   1st Qu.: 0.000   1st Qu.: 0.000   1:442   2:193
##  Median : 5.669   Median : 2.282   Median : 2.135           3:211
##  Mean   : 5.528   Mean   : 2.486   Mean   : 2.452           4:187
##  3rd Qu.: 8.367   3rd Qu.: 4.884   3rd Qu.: 4.782           5:198
##  Max.   :17.054   Max.   :10.445   Max.   :10.932
##
##        x5            x6                x7                 x8
##  c      :114   Min.   :-1.498   Min.   :-2.65008   3      :123
##  h      :114   1st Qu.: 3.000   1st Qu.:-0.58182   1      :110
##  b      :109   Median : 5.000   Median : 0.02262   7      :108
##  g      :104   Mean   : 4.948   Mean   : 0.03298   5      :106
##  a      :103   3rd Qu.: 7.000   3rd Qu.: 0.71906   10     :101
##  d      :102   Max.   :12.954   Max.   : 3.71572   4      :100
##  (Other):354                                       (Other):352
##        x9              x10          missing_y       missing_x1
##  Min.   :0.0132   Min.   :-1.954   Mode :logical   Mode :logical
##  1st Qu.:0.3200   1st Qu.: 2.097   FALSE:700       FALSE:700
##  Median :0.5269   Median : 4.000   TRUE :300       TRUE :300
##  Mean   :0.5357   Mean   : 3.851   NA's :0         NA's :0
##  3rd Qu.:0.7612   3rd Qu.: 5.000
##  Max.   :0.9954   Max.   :11.000
##
##  missing_x2      missing_x3      missing_x4      missing_x5
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical
##  FALSE:700       FALSE:700       FALSE:700       FALSE:700
##  TRUE :300       TRUE :300       TRUE :300       TRUE :300
##  NA's :0         NA's :0         NA's :0         NA's :0
##
##  missing_x6      missing_x7      missing_x8      missing_x9
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical
##  FALSE:700       FALSE:700       FALSE:700       FALSE:700
##  TRUE :300       TRUE :300       TRUE :300       TRUE :300
##  NA's :0         NA's :0         NA's :0         NA's :0
##  missing_x10
##  Mode :logical
##  FALSE:700
##  TRUE :300
##  NA's :0
##
## $`chain:3`
##        y                 x1                x2           x3       x4
##  Min.   :-5.132   Min.   :-8.769   Min.   :-3.643   0:538   1:200
##  1st Qu.: 2.414   1st Qu.: 0.000   1st Qu.: 0.000   1:462   2:182
##  Median : 5.632   Median : 2.034   Median : 2.610           3:215
##  Mean   : 5.537   Mean   : 2.417   Mean   : 2.530           4:211
##  3rd Qu.: 8.434   3rd Qu.: 4.836   3rd Qu.: 4.812           5:192
##  Max.   :16.945   Max.   :10.335   Max.   :11.683
##
##        x5             x6                x7                   x8
##  b      :123   Min.   :-2.223   Min.   :-2.76469   2      :139
##  j      :115   1st Qu.: 3.000   1st Qu.:-0.64886   5      :111
##  c      :111   Median : 5.000   Median : 0.03266   1      :110
##  h      :103   Mean   : 4.957   Mean   : 0.03220   3      :109
##  i      :103   3rd Qu.: 7.000   3rd Qu.: 0.71341   7      :106
##  a      :100   Max.   :11.785   Max.   : 3.71572   9      :100
##  (Other):345                                       (Other):325
##        x9                 x10          missing_y       missing_x1
##  Min.   :0.007236   Min.   :-1.522   Mode :logical   Mode :logical
##  1st Qu.:0.320579   1st Qu.: 2.224   FALSE:700       FALSE:700
##  Median :0.531962   Median : 4.000   TRUE :300       TRUE :300
##  Mean   :0.541147   Mean   : 3.894   NA's :0         NA's :0
##  3rd Qu.:0.772802   3rd Qu.: 5.000
##  Max.   :0.992118   Max.   :11.000
##
##  missing_x2      missing_x3      missing_x4      missing_x5
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical
##  FALSE:700       FALSE:700       FALSE:700       FALSE:700
##  TRUE :300       TRUE :300       TRUE :300       TRUE :300
##  NA's :0         NA's :0         NA's :0         NA's :0
##
##  missing_x6      missing_x7      missing_x8      missing_x9
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical
##  FALSE:700       FALSE:700       FALSE:700       FALSE:700
##  TRUE :300       TRUE :300       TRUE :300       TRUE :300
##  NA's :0         NA's :0         NA's :0         NA's :0
##
##  missing_x10
##  Mode :logical
##  FALSE:700
##  TRUE :300
##  NA's :0
```

```
coef(summary(model_results))[, 1:2]  # get the model coef's and their SE's
##                Estimate Std. Error
## (Intercept)  0.76906403 0.83558319
## x1           0.94250085 0.04535482
## x2           0.97494755 0.03517050
## x31         -0.27348764 0.37377108
## x4.L         0.21116072 0.21050885
## x4.Q        -0.08566591 0.15626753
## x4.C         0.02957084 0.24489775
## x4^4          0.24986739 0.19503550
## x5b           0.03327092 0.41562660
## x5c          -0.41123612 0.25525293
## x5d          -0.21576243 0.86289626
## x5e           0.11334262 0.56396149
## x5f          -0.13161632 0.55187362
## x5g          -0.27013537 0.67021765
## x5h          -0.16951449 0.66293826
## x5i          -0.68618715 0.80974757
## x5j           0.20681081 0.27822872
## x6           -0.04008539 0.07305886
## x7            0.98130349 0.08526896
## x82           0.43773548 0.38574473
## x83           0.40306683 0.20475089
## x84          -0.13651311 0.62306988
## x85           0.19905219 0.29973411
## x86           0.18661601 0.24702280
## x87           0.18792270 0.38028640
## x88           0.51105644 0.34272121
## x89           0.25124560 0.26340027
## x810          0.17382802 0.47841078
## x9            0.87514342 0.71483501
## x10          -0.05721504 0.05035334

library("lattice")
densityplot(y ~ x1 + x2, data=imputations)
```

This plot, Fig. 3.21, allows us to compare the density of observed data and imputed data–these should be similar (though not identical) under MAR assumptions.

3.13.2 TBI Data Example

Next, we will see an example using the traumatic brain injury (TBI) dataset.

```
# Load the (raw) data from the table into a plain text file "08_EpiBioSData_
Incomplete.csv"
TBI_Data <- read.csv("https://umich.instructure.com/files/720782/download?do
wnload_frd=1", na.strings=c("", ".", "NA"))
summary(TBI_Data)

##        id             age             sex             mechanism
##  Min.   : 1.00   Min.   :16.00   Female: 9   Bike_vs_Auto: 4
##  1st Qu.:12.25   1st Qu.:23.00   Male  :37   Blunt       : 4
##  Median :23.50   Median :33.00               Fall        :13
##  Mean   :23.50   Mean   :36.89               GSW         : 2
##  3rd Qu.:34.75   3rd Qu.:47.25               MCA         : 7
##  Max.   :46.00   Max.   :83.00               MVA         :10
##                                              Peds_vs_Auto: 6
##    field.gcs       er.gcs          icu.gcs          worst.gcs
##  Min.   : 3   Min.   : 3.000   Min.   : 0.000   Min.   : 0.0
##  1st Qu.: 3   1st Qu.: 4.000   1st Qu.: 3.000   1st Qu.: 3.0
##  Median : 7   Median : 7.500   Median : 6.000   Median : 3.0
##  Mean   : 8   Mean   : 8.182   Mean   : 6.378   Mean   : 5.4
##  3rd Qu.:12   3rd Qu.:12.250   3rd Qu.: 8.000   3rd Qu.: 7.0
##  Max.   :15   Max.   :15.000   Max.   :14.000   Max.   :14.0
##  NA's   :2    NA's   :2        NA's   :1        NA's   :1
##     X6m.gose       X2013.gose       skull.fx       temp.injury
##  Min.   :2.000   Min.   :2.000   Min.   :0.0000   Min.   :0.000
##  1st Qu.:3.000   1st Qu.:5.000   1st Qu.:0.0000   1st Qu.:0.000
##  Median :5.000   Median :7.000   Median :1.0000   Median :1.000
##  Mean   :4.805   Mean   :5.804   Mean   :0.6087   Mean   :0.587
##  3rd Qu.:6.000   3rd Qu.:7.000   3rd Qu.:1.0000   3rd Qu.:1.000
##  Max.   :8.000   Max.   :8.000   Max.   :1.0000   Max.   :1.000
##  NA's   :5
##     surgery          spikes.hr          min.hr          max.hr
##  Min.   :0.0000   Min.   :  1.280   Min.   : 0.000   Min.   :  12.00
##  1st Qu.:0.0000   1st Qu.:  5.357   1st Qu.: 0.000   1st Qu.:  35.25
##  Median :1.0000   Median : 18.170   Median : 0.000   Median :  97.50
##  Mean   :0.6304   Mean   : 52.872   Mean   : 3.571   Mean   : 241.89
##  3rd Qu.:1.0000   3rd Qu.: 57.227   3rd Qu.: 0.000   3rd Qu.: 312.75
##  Max.   :1.0000   Max.   :294.000   Max.   :42.000   Max.   :1199.00
##                   NA's   :18        NA's   :18       NA's   :18
##     acute.sz         late.sz          ever.sz
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000
##  Median :0.0000   Median :1.0000   Median :1.000
##  Mean   :0.1739   Mean   :0.5652   Mean   :0.587
##  3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:1.000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.000
##
```

1. `Convert to a missing_data.frame` (Fig. 3.22)

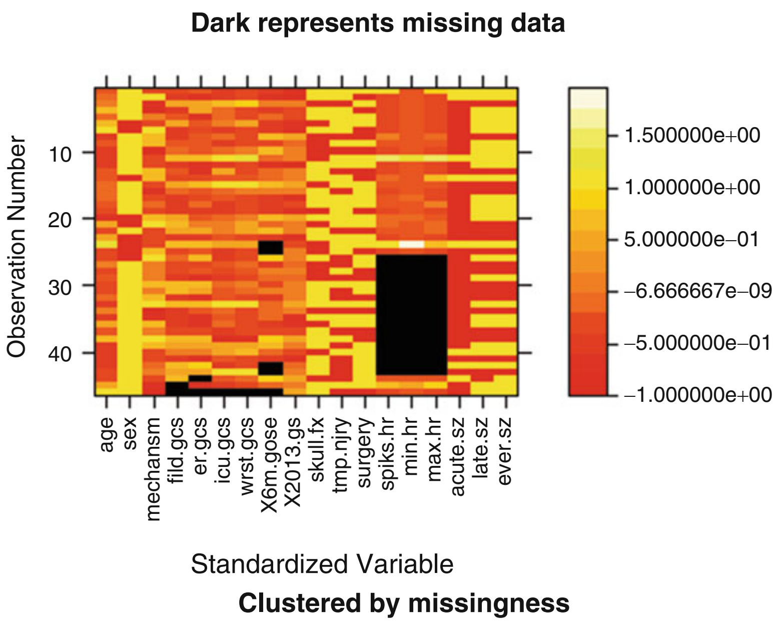

Fig. 3.22 Missing data pattern for the TBI case-study

```
# Get information matrix of the data
# library("betareg"); library("mi")
mdf <- missing_data.frame(TBI_Data) # warnings about missingness patterns

## NOTE: The following pairs of variables appear to have the same missingnes
s pattern.
##  Please verify whether they are in fact logically distinct variables.
##      [,1]      [,2]
## [1,] "icu.gcs" "worst.gcs"

show(mdf); mdf@patterns; image(mdf)

## Object of class missing_data.frame with 46 observations on 19 variables
##
## There are 7 missing data patterns
##
## Append '@patterns' to this missing_data.frame to access the corresponding
pattern for every observation or perhaps use table()
##
##                               type missing method  model
## id                      irrelevant       0   <NA>   <NA>
## age                     continuous       0   <NA>   <NA>
## sex                         binary       0   <NA>   <NA>
## mechanism   unordered-categorical       0   <NA>   <NA>
## field.gcs               continuous       2    ppd linear
## er.gcs                  continuous       2    ppd linear
## icu.gcs                 continuous       1    ppd linear
## worst.gcs               continuous       1    ppd linear
## X6m.gose                continuous       5    ppd linear
## X2013.gose              continuous       0   <NA>   <NA>
## skull.fx                    binary       0   <NA>   <NA>
## temp.injury                 binary       0   <NA>   <NA>
## surgery                     binary       0   <NA>   <NA>
## spikes.hr               continuous      18    ppd linear
## min.hr                  continuous      18    ppd linear
## max.hr                  continuous      18    ppd linear
## acute.sz                    binary       0   <NA>   <NA>
## late.sz                     binary       0   <NA>   <NA>
## ever.sz                     binary       0   <NA>   <NA>
##
##                family     link transformation
## id               <NA>     <NA>           <NA>
## age              <NA>     <NA>    standardize
## sex              <NA>     <NA>           <NA>
## mechanism        <NA>     <NA>           <NA>
## field.gcs    gaussian identity    standardize
## er.gcs       gaussian identity    standardize
## icu.gcs      gaussian identity    standardize
## worst.gcs    gaussian identity    standardize
## X6m.gose     gaussian identity    standardize
## X2013.gose       <NA>     <NA>    standardize
## skull.fx         <NA>     <NA>           <NA>
## temp.injury      <NA>     <NA>           <NA>
## surgery          <NA>     <NA>           <NA>
## spikes.hr    gaussian identity    standardize
## min.hr       gaussian identity    standardize
## max.hr       gaussian identity    standardize
## acute.sz         <NA>     <NA>           <NA>
## late.sz          <NA>     <NA>           <NA>
## ever.sz          <NA>     <NA>            <NA>
```

```
##  [1] spikes.hr, min.hr, max.hr
##  [2] field.gcs
##  [3] nothing
##  [4] nothing
##  [5] nothing
##  [6] nothing
##  [7] spikes.hr, min.hr, max.hr
##  [8] nothing
##  [9] nothing
## [10] nothing
## [11] nothing
## [12] nothing
## [13] spikes.hr, min.hr, max.hr
## [14] nothing
## [15] spikes.hr, min.hr, max.hr
## [16] spikes.hr, min.hr, max.hr
## [17] nothing
## [18] spikes.hr, min.hr, max.hr
## [19] spikes.hr, min.hr, max.hr
## [20] spikes.hr, min.hr, max.hr
## [21] X6m.gose, spikes.hr, min.hr, max.hr
## [22] nothing
## [23] spikes.hr, min.hr, max.hr
## [24] spikes.hr, min.hr, max.hr
## [25] spikes.hr, min.hr, max.hr
## [26] spikes.hr, min.hr, max.hr
## [27] spikes.hr, min.hr, max.hr
## [28] X6m.gose
## [29] spikes.hr, min.hr, max.hr
## [30] nothing
## [31] X6m.gose, spikes.hr, min.hr, max.hr
## [32] spikes.hr, min.hr, max.hr
## [33] nothing
## [34] nothing
## [35] nothing
## [36] nothing
## [37] field.gcs, er.gcs, icu.gcs, worst.gcs, X6m.gose
## [38] er.gcs
## [39] nothing
## [40] nothing
## [41] nothing
## [42] spikes.hr, min.hr, max.hr
## [43] nothing
## [44] nothing
## [45] nothing
## [46] X6m.gose
## 7 Levels: nothing field.gcs X6m.gose er.gcs ... field.gcs, er.gcs, icu.gc
s, worst.gcs, X6m.gose
```

2. *Configuring the imputation process.*

```
# mi::change() method changes the family imputation method,
# size, type, and so forth of a missing variable. It's called
# before calling mi to affect how the conditional expectation of each
# missing variable is modeled.
mdf <- change(mdf, y = "spikes.hr", what = "transformation", to="identity")
```

3. `Examine the missingness patterns.`

```
summary(mdf); hist(mdf);

##        id             age            sex           mechanism
##  Min.   : 1.00   Min.   :16.00   Female: 9   Bike_vs_Auto: 4
##  1st Qu.:12.25   1st Qu.:23.00   Male  :37   Blunt       : 4
##  Median :23.50   Median :33.00               Fall        :13
##  Mean   :23.50   Mean   :36.89               GSW         : 2
##  3rd Qu.:34.75   3rd Qu.:47.25               MCA         : 7
##  Max.   :46.00   Max.   :83.00               MVA         :10
##                                              Peds_vs_Auto: 6
##    field.gcs       er.gcs          icu.gcs          worst.gcs
##  Min.   : 3   Min.   : 3.000   Min.   : 0.000   Min.   : 0.0
##  1st Qu.: 3   1st Qu.: 4.000   1st Qu.: 3.000   1st Qu.: 3.0
##  Median : 7   Median : 7.500   Median : 6.000   Median : 3.0
##  Mean   : 8   Mean   : 8.182   Mean   : 6.378   Mean   : 5.4
##  3rd Qu.:12   3rd Qu.:12.250   3rd Qu.: 8.000   3rd Qu.: 7.0
##  Max.   :15   Max.   :15.000   Max.   :14.000   Max.   :14.0
##  NA's   :2    NA's   :2        NA's   :1        NA's   :1
##     X6m.gose       X2013.gose       skull.fx       temp.injury
##  Min.   :2.000   Min.   :2.000   Min.   :0.0000   Min.   :0.000
##  1st Qu.:3.000   1st Qu.:5.000   1st Qu.:0.0000   1st Qu.:0.000
##  Median :5.000   Median :7.000   Median :1.0000   Median :1.000
##  Mean   :4.805   Mean   :5.804   Mean   :0.6087   Mean   :0.587
##  3rd Qu.:6.000   3rd Qu.:7.000   3rd Qu.:1.0000   3rd Qu.:1.000
##  Max.   :8.000   Max.   :8.000   Max.   :1.0000   Max.   :1.000
##  NA's   :5
##     surgery          spikes.hr          min.hr           max.hr
##  Min.   :0.0000   Min.   :  1.280   Min.   : 0.000   Min.   :  12.00
##  1st Qu.:0.0000   1st Qu.:  5.357   1st Qu.: 0.000   1st Qu.:  35.25
##  Median :1.0000   Median : 18.170   Median : 0.000   Median :  97.50
##  Mean   :0.6304   Mean   : 52.872   Mean   : 3.571   Mean   : 241.89
##  3rd Qu.:1.0000   3rd Qu.: 57.227   3rd Qu.: 0.000   3rd Qu.: 312.75
##  Max.   :1.0000   Max.   :294.000   Max.   :42.000   Max.   :1199.00
##                   NA's   :18        NA's   :18       NA's   :18
##     acute.sz          late.sz          ever.sz
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000
##  Median :0.0000   Median :1.0000   Median :1.000
##  Mean   :0.1739   Mean   :0.5652   Mean   :0.587
##  3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:1.000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.000
##

image(mdf)
```

4. `Perform the initial imputation` (Fig. 3.23).

```
imputations <- mi(mdf, n.iter=10, n.chains=5, verbose=TRUE)
hist(imputations)
```

5. *Extracts several multiply imputed data.frames from the "imputations" object.*

```
data.frames <- complete(imputations, 5)
```

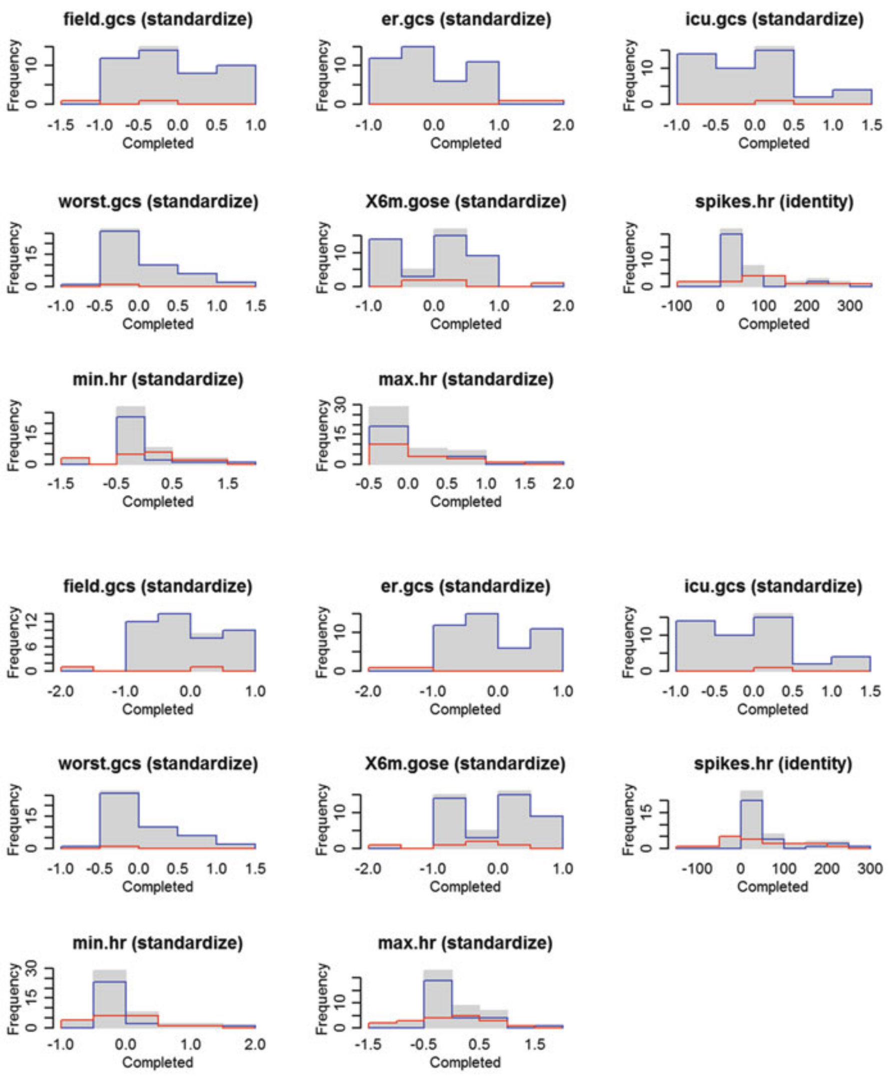

Fig. 3.23 Validation plots for the original, imputed and complete TBI datasets

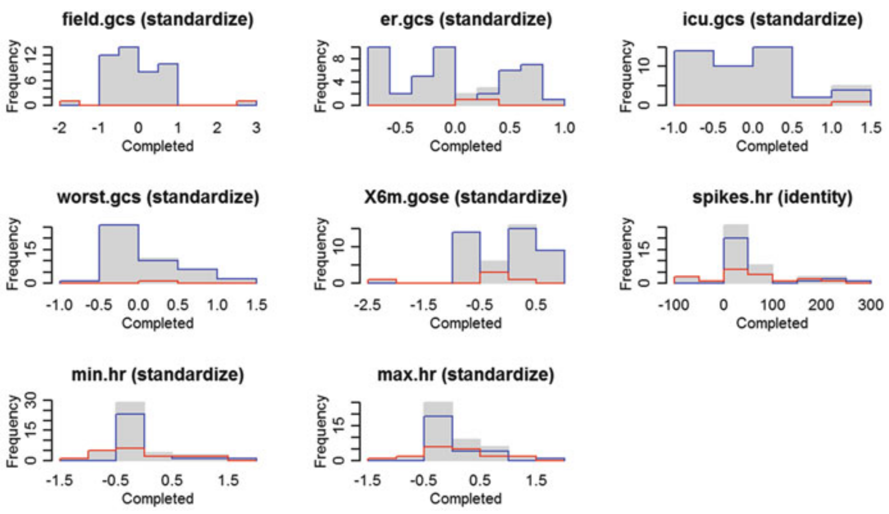

Fig. 3.23 (continued)

6. *Report a list of "summaries" for each element (imputation instance).*

```
lapply(data.frames, summary)

## $`chain:1`
##        id             age             sex          mechanism
##  Min.   : 1.00   Min.   :16.00   Female: 9   Bike_vs_Auto: 4
##  1st Qu.:12.25   1st Qu.:23.00   Male  :37   Blunt       : 4
##  Median :23.50   Median :33.00               Fall        :13
##  Mean   :23.50   Mean   :36.89               GSW         : 2
##  3rd Qu.:34.75   3rd Qu.:47.25               MCA         : 7
##  Max.   :46.00   Max.   :83.00               MVA         :10
##                                              Peds_vs_Auto: 6
##    field.gcs         er.gcs          icu.gcs         worst.gcs
##  Min.   :-3.424   Min.   : 3.000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 3.000   1st Qu.: 4.250   1st Qu.: 3.000   1st Qu.: 3.000
##  Median : 6.500   Median : 8.000   Median : 6.000   Median : 3.000
##  Mean   : 7.593   Mean   : 8.442   Mean   : 6.285   Mean   : 5.494
##  3rd Qu.:12.000   3rd Qu.:13.000   3rd Qu.: 7.750   3rd Qu.: 7.750
##  Max.   :15.000   Max.   :15.000   Max.   :14.000   Max.   :14.000
##
##     X6m.gose       X2013.gose     skull.fx temp.injury surgery
##  Min.   :2.000   Min.   :2.000   0:18     0:19        0:17
##  1st Qu.:3.000   1st Qu.:5.000   1:28     1:27        1:29
##  Median :5.000   Median :7.000
##  Mean   :5.031   Mean   :5.804
##  3rd Qu.:6.815   3rd Qu.:7.000
##  Max.   :8.169   Max.   :8.000
##
```

```
##    spikes.hr          min.hr            max.hr        acute.sz late.sz
##  Min.   :-86.914   Min.   :-11.697   Min.   :-153.94   0:38     0:20
##  1st Qu.:  3.953   1st Qu.:  0.000   1st Qu.:  42.25   1: 8     1:26
##  Median : 28.125   Median :  0.000   Median : 211.49
##  Mean   : 59.108   Mean   :  7.133   Mean   : 282.79
##  3rd Qu.:113.615   3rd Qu.: 11.329   3rd Qu.: 390.63
##  Max.   :294.000   Max.   : 43.706   Max.   :1199.00
##
##  ever.sz missing_field.gcs missing_er.gcs  missing_icu.gcs
##  0:19    Mode :logical     Mode :logical   Mode :logical
##  1:27    FALSE:44          FALSE:44        FALSE:45
##          TRUE :2           TRUE :2         TRUE :1
##          NA's :0           NA's :0         NA's :0
##
##  missing_worst.gcs missing_X6m.gose missing_spikes.hr missing_min.hr
##  Mode :logical     Mode :logical    Mode :logical     Mode :logical
##  FALSE:45          FALSE:41         FALSE:28          FALSE:28
##  TRUE :1           TRUE :5          TRUE :18          TRUE :18
##  NA's :0           NA's :0          NA's :0           NA's :0
##
##  missing_max.hr
##  Mode :logical
##  FALSE:28
##  TRUE :18
##  NA's :0
##
## $`chain:2`
##        id             age            sex            mechanism
##  Min.   : 1.00   Min.   :16.00   Female: 9   Bike_vs_Auto: 4
##  1st Qu.:12.25   1st Qu.:23.00   Male  :37   Blunt       : 4
##  Median :23.50   Median :33.00               Fall        :13
##  Mean   :23.50   Mean   :36.89               GSW         : 2
##  3rd Qu.:34.75   3rd Qu.:47.25               MCA         : 7
##  Max.   :46.00   Max.   :83.00               MVA         :10
##                                              Peds_vs_Auto: 6
##    field.gcs         er.gcs          icu.gcs        worst.gcs
##  Min.   :-3.324   Min.   : 3.000   Min.   : 0.00   Min.   : 0.000
##  1st Qu.: 3.000   1st Qu.: 4.000   1st Qu.: 3.00   1st Qu.: 3.000
##  Median : 6.500   Median : 7.000   Median : 6.00   Median : 3.000
##  Mean   : 7.658   Mean   : 8.046   Mean   : 6.24   Mean   : 5.466
##  3rd Qu.:12.000   3rd Qu.:12.000   3rd Qu.: 7.75   3rd Qu.: 7.750
##  Max.   :15.000   Max.   :15.000   Max.   :14.00   Max.   :14.000
##
##     X6m.gose        X2013.gose    skull.fx temp.injury surgery
##  Min.   :-3.196   Min.   :2.000   0:18     0:19        0:17
##  1st Qu.: 3.000   1st Qu.:5.000   1:28     1:27        1:29
##  Median : 5.000   Median :7.000
##  Mean   : 4.755   Mean   :5.804
##  3rd Qu.: 6.117   3rd Qu.:7.000
##  Max.   : 8.000   Max.   :8.000
##
##    spikes.hr           min.hr            max.hr        acute.sz late.sz
##  Min.   :-138.854   Min.   :-30.603   Min.   :-432.95   0:38     0:20
##  1st Qu.:   5.518   1st Qu.:  0.000   1st Qu.:  28.75   1: 8     1:26
##  Median :  34.522   Median :  0.000   Median :  97.50
##  Mean   :  61.310   Mean   :  2.329   Mean   : 209.53
##  3rd Qu.:  97.394   3rd Qu.:  4.306   3rd Qu.: 306.98
##  Max.   : 294.000   Max.   : 42.000   Max.   :1336.12  
##                                                         
##  ever.sz missing_field.gcs missing_er.gcs  missing_icu.gcs
##  0:19    Mode :logical     Mode :logical   Mode :logical  
##  1:27    FALSE:44          FALSE:44        FALSE:45       
##          TRUE :2           TRUE :2         TRUE :1        
##          NA's :0           NA's :0         NA's :0        
##                                                           
##  missing_worst.gcs missing_X6m.gose missing_spikes.hr missing_min.hr
##  Mode :logical     Mode :logical    Mode :logical     Mode :logical 
##  FALSE:45          FALSE:41         FALSE:28          FALSE:28      
##  TRUE :1           TRUE :5          TRUE :18          TRUE :18      
##  NA's :0           NA's :0          NA's :0           NA's :0       
##                                                                     
##  missing_max.hr
##  Mode :logical 
##  FALSE:28      
##  TRUE :18      
##  NA's :0       
##                
## $`chain:3`
##        id             age            sex           mechanism 
##  Min.   : 1.00   Min.   :16.00   Female: 9   Bike_vs_Auto: 4  
##  1st Qu.:12.25   1st Qu.:23.00   Male  :37   Blunt       : 4  
##  Median :23.50   Median :33.00               Fall        :13  
##  Mean   :23.50   Mean   :36.89               GSW         : 2  
##  3rd Qu.:34.75   3rd Qu.:47.25               MCA         : 7  
##  Max.   :46.00   Max.   :83.00               MVA         :10  
##                                              Peds_vs_Auto: 6  
##    field.gcs          er.gcs          icu.gcs         worst.gcs    
##  Min.   : 3.000   Min.   : 3.000   Min.   : 0.000   Min.   : 0.000  
##  1st Qu.: 3.250   1st Qu.: 4.191   1st Qu.: 3.000   1st Qu.: 3.000  
##  Median : 7.000   Median : 7.500   Median : 6.000   Median : 3.000  
##  Mean   : 8.218   Mean   : 8.513   Mean   : 6.325   Mean   : 5.304  
##  3rd Qu.:12.000   3rd Qu.:12.750   3rd Qu.: 7.750   3rd Qu.: 7.000  
##  Max.   :17.978   Max.   :26.831   Max.   :14.000   Max.   :14.000  
##                                                                     
##     X6m.gose       X2013.gose    skull.fx temp.injury surgery
##  Min.   :2.000   Min.   :2.000   0:18     0:19        0:17   
##  1st Qu.:3.000   1st Qu.:5.000   1:28     1:27        1:29   
##  Median :5.000   Median :7.000                               
##  Mean   :4.892   Mean   :5.804                               
##  3rd Qu.:6.000   3rd Qu.:7.000                               
##  Max.   :8.000   Max.   :8.000                               
##                                                              
##    spikes.hr          min.hr            max.hr       acute.sz late.sz
##  Min.   :-40.459   Min.   :-27.222   Min.   :-236.6   0:38     0:20   
##  1st Qu.:  5.518   1st Qu.:  0.000   1st Qu.:  37.5   1: 8     1:26   
##  Median : 34.864   Median :  0.000   Median : 195.6                   
##  Mean   : 65.781   Mean   :  2.619   Mean   : 281.8                   
##  3rd Qu.:100.137   3rd Qu.:  5.681   3rd Qu.: 476.1                   
##  Max.   :294.000   Max.   : 42.000   Max.   :1199.0                   
##                                                                       
##  ever.sz missing_field.gcs missing_er.gcs  missing_icu.gcs
##  0:19    Mode :logical     Mode :logical   Mode :logical  
##  1:27    FALSE:44          FALSE:44        FALSE:45       
##          TRUE :2           TRUE :2         TRUE :1        
##          NA's :0           NA's :0          NA's :0
##
##  missing_worst.gcs missing_X6m.gose missing_spikes.hr missing_min.hr
##  Mode :logical     Mode :logical    Mode :logical     Mode :logical
##  FALSE:45          FALSE:41         FALSE:28          FALSE:28
##  TRUE :1           TRUE :5          TRUE :18          TRUE :18
##  NA's :0           NA's :0          NA's :0           NA's :0
##
##  missing_max.hr
##  Mode :logical
##  FALSE:28
##  TRUE :18
##  NA's :0
##
## $`chain:4`
##        id              age            sex           mechanism
##  Min.   : 1.00   Min.   :16.00   Female: 9   Bike_vs_Auto: 4
##  1st Qu.:12.25   1st Qu.:23.00   Male  :37   Blunt       : 4
##  Median :23.50   Median :33.00               Fall        :13
##  Mean   :23.50   Mean   :36.89               GSW         : 2
##  3rd Qu.:34.75   3rd Qu.:47.25               MCA         : 7
##  Max.   :46.00   Max.   :83.00               MVA         :10
##                                              Peds_vs_Auto: 6
##    field.gcs          er.gcs          icu.gcs          worst.gcs
##  Min.   : 3.000   Min.   :-7.960   Min.   :-1.610   Min.   :-4.339
##  1st Qu.: 3.250   1st Qu.: 4.000   1st Qu.: 3.000   1st Qu.: 3.000
##  Median : 7.000   Median : 7.000   Median : 6.000   Median : 3.000
##  Mean   : 8.001   Mean   : 7.746   Mean   : 6.204   Mean   : 5.188
##  3rd Qu.:12.000   3rd Qu.:12.000   3rd Qu.: 7.750   3rd Qu.: 7.000
##  Max.   :15.000   Max.   :15.000   Max.   :14.000   Max.   :14.000
##
##     X6m.gose       X2013.gose    skull.fx temp.injury surgery
##  Min.   :2.000   Min.   :2.000   0:18     0:19        0:17
##  1st Qu.:3.000   1st Qu.:5.000   1:28     1:27        1:29
##  Median :5.000   Median :7.000
##  Mean   :5.095   Mean   :5.804
##  3rd Qu.:6.997   3rd Qu.:7.000
##  Max.   :8.930   Max.   :8.000
##
##    spikes.hr           min.hr             max.hr       acute.sz late.sz
##  Min.   :-106.439   Min.   :-28.5276   Min.   :-536.00   0:38     0:20
##  1st Qu.:   4.577   1st Qu.:  0.0000   1st Qu.:  32.21   1: 8     1:26
##  Median :  29.593   Median :  0.0000   Median :  98.15
##  Mean   :  52.290   Mean   : -0.1032   Mean   : 197.81
##  3rd Qu.:  84.794   3rd Qu.:  0.1667   3rd Qu.: 333.46
##  Max.   : 294.000   Max.   : 42.0000   Max.   :1199.00
##
##  ever.sz missing_field.gcs missing_er.gcs  missing_icu.gcs
##  0:19    Mode :logical     Mode :logical   Mode :logical
##  1:27    FALSE:44          FALSE:44        FALSE:45
##          TRUE :2           TRUE :2         TRUE :1
##          NA's :0           NA's :0         NA's :0
##
##  missing_worst.gcs missing_X6m.gose missing_spikes.hr missing_min.hr
##  Mode :logical     Mode :logical    Mode :logical     Mode :logical
##  FALSE:45          FALSE:41         FALSE:28          FALSE:28
##  TRUE :1           TRUE :5          TRUE :18          TRUE :18
##  NA's :0           NA's :0          NA's :0           NA's :0
##
##  missing_max.hr
##  Mode :logical
##  FALSE:28
##  TRUE :18
##  NA's :0
##
## $`chain:5`
##        id             age            sex            mechanism
##  Min.   : 1.00   Min.   :16.00   Female: 9   Bike_vs_Auto: 4
##  1st Qu.:12.25   1st Qu.:23.00   Male  :37   Blunt       : 4
##  Median :23.50   Median :33.00               Fall        :13
##  Mean   :23.50   Mean   :36.89               GSW         : 2
##  3rd Qu.:34.75   3rd Qu.:47.25               MCA         : 7
##  Max.   :46.00   Max.   :83.00               MVA         :10
##                                              Peds_vs_Auto: 6
##    field.gcs          er.gcs          icu.gcs          worst.gcs
##  Min.   : 3.000   Min.   :-15.73   Min.   : 0.000   Min.   :-2.742
##  1st Qu.: 3.250   1st Qu.:  4.00   1st Qu.: 3.000   1st Qu.: 3.000
##  Median : 7.000   Median :  7.32   Median : 6.000   Median : 3.000
##  Mean   : 8.473   Mean   :  7.65   Mean   : 6.439   Mean   : 5.223
##  3rd Qu.:12.750   3rd Qu.: 12.00   3rd Qu.: 8.000   3rd Qu.: 7.000
##  Max.   :20.172   Max.   : 15.00   Max.   :14.000   Max.   :14.000
##
##     X6m.gose        X2013.gose    skull.fx temp.injury surgery
##  Min.   : 2.000   Min.   :2.000   0:18     0:19        0:17
##  1st Qu.: 3.000   1st Qu.:5.000   1:28     1:27        1:29
##  Median : 5.000   Median :7.000
##  Mean   : 4.972   Mean   :5.804
##  3rd Qu.: 6.000   3rd Qu.:7.000
##  Max.   :11.481   Max.   :8.000
##
##    spikes.hr           min.hr            max.hr       acute.sz late.sz
##  Min.   :-74.552   Min.   :-11.877   Min.   :-570.4   0:38     0:20
##  1st Qu.:  5.518   1st Qu.: -1.924   1st Qu.:  37.5   1: 8     1:26
##  Median : 32.297   Median :  0.000   Median : 175.3
##  Mean   : 54.268   Mean   :  1.022   Mean   : 253.7
##  3rd Qu.: 71.288   3rd Qu.:  0.000   3rd Qu.: 432.1
##  Max.   :294.000   Max.   : 42.000   Max.   :1199.0
##
##  ever.sz missing_field.gcs missing_er.gcs  missing_icu.gcs
##  0:19    Mode :logical     Mode :logical   Mode :logical
##  1:27    FALSE:44          FALSE:44        FALSE:45
##          TRUE :2           TRUE :2         TRUE :1
##          NA's :0           NA's :0         NA's :0
##
##  missing_worst.gcs missing_X6m.gose missing_spikes.hr missing_min.hr
##  Mode :logical     Mode :logical    Mode :logical     Mode :logical
##  FALSE:45          FALSE:41         FALSE:28          FALSE:28
##  TRUE :1           TRUE :5          TRUE :18          TRUE :18
##  NA's :0           NA's :0          NA's :0           NA's :0
##
##  missing_max.hr
##  Mode :logical
##  FALSE:28
##  TRUE :18
##  NA's :0
```

7. Cast the imputed numbers as integers.

```
# (not necessary, but may be useful)
indx <- sapply(data.frames[[5]], is.numeric)  # get the indices of
numeric columns
data.frames[[5]][indx] <- lapply(data.frames[[5]][indx], function(x)
as.numeric(as.integer(x)))

# cast each value as integer data.frames[[5]]$spikes.hr
```

8. Save results out.

```
write.csv(data.frames[[5]], "C:\\Users\\User\\Desktop\\TBI_MIData.csv")
```

9. Complete Data analytics functions.

```
# library("mi")
# lm.mi(); glm.mi(); polr.mi(); bayesglm.mi(); bayespolr.mi(); lmer.mi(); gl
mer.mi()
```

10. Fit a linear model for one multiply imputed chain.

```
# Also see Step (9)
##linear regression for each imputed data set - 5 regression models are fit
fit_lm1 <- glm(ever.sz ~ surgery + worst.gcs + factor(sex) + age, data.frame
s$`chain:1`, family = "binomial"); summary(fit_lm1); display(fit_lm1)

##
## Call:
## glm(formula = ever.sz ~ surgery + worst.gcs + factor(sex) + age,
##     family = "binomial", data = data.frames$`chain:1`)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7000  -1.2166   0.8222   1.0007   1.3871
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)       0.249780   1.356397   0.184    0.854
## surgery1          0.947392   0.685196   1.383    0.167
## worst.gcs        -0.068734   0.097962  -0.702    0.483
## factor(sex)Male -0.329313   0.842761  -0.391    0.696
## age               0.004453   0.019431   0.229    0.819
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 62.371  on 45  degrees of freedom
## Residual deviance: 60.046  on 41  degrees of freedom
## AIC: 70.046
##
## Number of Fisher Scoring iterations: 4
## glm(formula = ever.sz ~ surgery + worst.gcs + factor(sex) + age,
##     family = "binomial", data = data.frames$`chain:1`)
##                   coef.est coef.se
## (Intercept)        0.25     1.36
## surgery1           0.95     0.69
## worst.gcs         -0.07     0.10
## factor(sex)Male -0.33     0.84
## age                0.00     0.02
## ---
##   n = 46, k = 5
##   residual deviance = 60.0, null deviance = 62.4 (difference = 2.3)
```

11. Fit the appropriate model and pool the results.

```
# (estimates over MI chains)
model_results <- pool(ever.sz ~ surgery + worst.gcs + factor(sex) + age,
family = "binomial", data=imputations,  m=5)
display (model_results); summary (model_results)
```

```
## bayesglm(formula = ever.sz ~ surgery + worst.gcs + factor(sex) +
##     age, data = imputations, m = 5, family = "binomial")
##                   coef.est coef.se
## (Intercept)        0.46     1.34
## surgery1           0.94     0.66
## worst.gcs         -0.09     0.10
## factor(sex)Male -0.33     0.77
## age                0.00     0.02
## n = 41, k = 5
## residual deviance = 59.3, null deviance = 62.4 (difference = 3.0)

##
## Call:
## pool(formula = ever.sz ~ surgery + worst.gcs + factor(sex) +
##     age, data = imputations, m = 5, family = "binomial")
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6796  -1.1802   0.8405   1.0225   1.3824
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)        0.459917   1.344700   0.342    0.732
## surgery1           0.938109   0.661646   1.418    0.156
## worst.gcs         -0.089340   0.098293  -0.909    0.363
## factor(sex)Male -0.332875   0.770327  -0.432    0.666
## age                0.001582   0.019685   0.080    0.936
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 62.371  on 45  degrees of freedom
## Residual deviance: 59.343  on 41  degrees of freedom
## AIC: 69.343
##
## Number of Fisher Scoring iterations: 6.6
```

12. *Report the summaries of the imputations.*

```
data.frames <- complete(imputations, 3)     # extract the first 3 chains
lapply(data.frames, summary)
## $`chain:1`
##        id              age              sex            mechanism
##  Min.   : 1.00   Min.   :16.00   Female: 9   Bike_vs_Auto: 4
##  1st Qu.:12.25   1st Qu.:23.00   Male  :37   Blunt       : 4
##  Median :23.50   Median :33.00               Fall        :13
##  Mean   :23.50   Mean   :36.89               GSW         : 2
##  3rd Qu.:34.75   3rd Qu.:47.25               MCA         : 7
##  Max.   :46.00   Max.   :83.00               MVA         :10
##                                              Peds_vs_Auto: 6
…
##  missing_max.hr
##  Mode :logical
##  FALSE:28
##  TRUE :18
##  NA's :0

## $`chain:2`
##        id              age              sex            mechanism
##  Min.   : 1.00   Min.   :16.00   Female: 9   Bike_vs_Auto: 4
##  1st Qu.:12.25   1st Qu.:23.00   Male  :37   Blunt       : 4
##  Median :23.50   Median :33.00               Fall        :13
##  Mean   :23.50   Mean   :36.89               GSW         : 2
##  3rd Qu.:34.75   3rd Qu.:47.25               MCA         : 7
##  Max.   :46.00   Max.   :83.00               MVA         :10
##                                              Peds_vs_Auto: 6
…
##  missing_max.hr
##  Mode :logical
##  FALSE:28
##  TRUE :18
##  NA's :0
##
## $`chain:3`
##        id              age              sex            mechanism
##  Min.   : 1.00   Min.   :16.00   Female: 9   Bike_vs_Auto: 4
##  1st Qu.:12.25   1st Qu.:23.00   Male  :37   Blunt       : 4
##  Median :23.50   Median :33.00               Fall        :13
##  Mean   :23.50   Mean   :36.89               GSW         : 2
##  3rd Qu.:34.75   3rd Qu.:47.25               MCA         : 7
##  Max.   :46.00   Max.   :83.00               MVA         :10
##                                              Peds_vs_Auto: 6
…
##  missing_max.hr
##  Mode :logical
##  FALSE:28
##  TRUE :18
##  NA's :0
```

13. *Validation:*
Next, we can verify whether enough iterations were conducted. One validation criteria demands that the mean of each completed variable is similar to the corresponding meen of the complete data (Fig. 3.24).

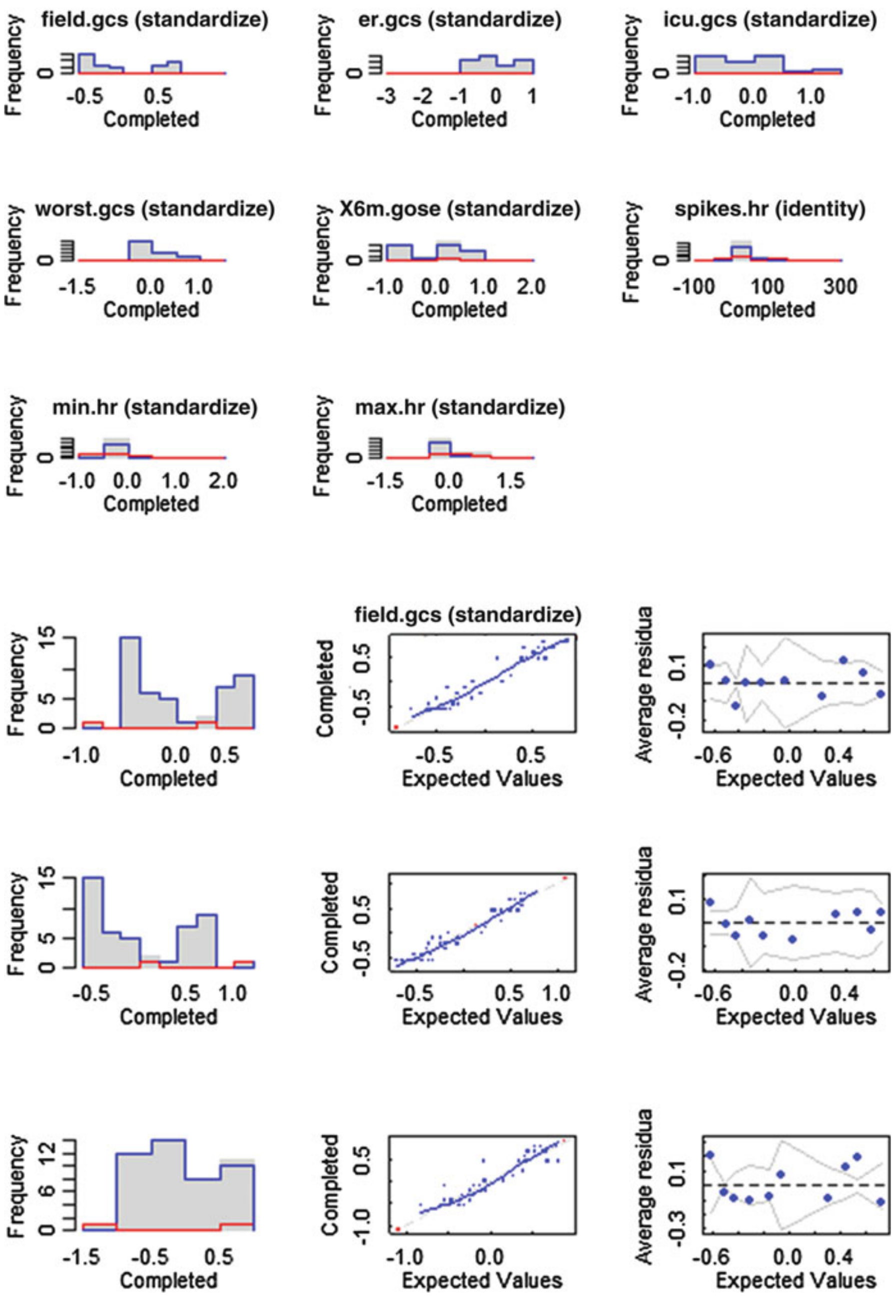

Fig. 3.24 TBI data imputation quality plots

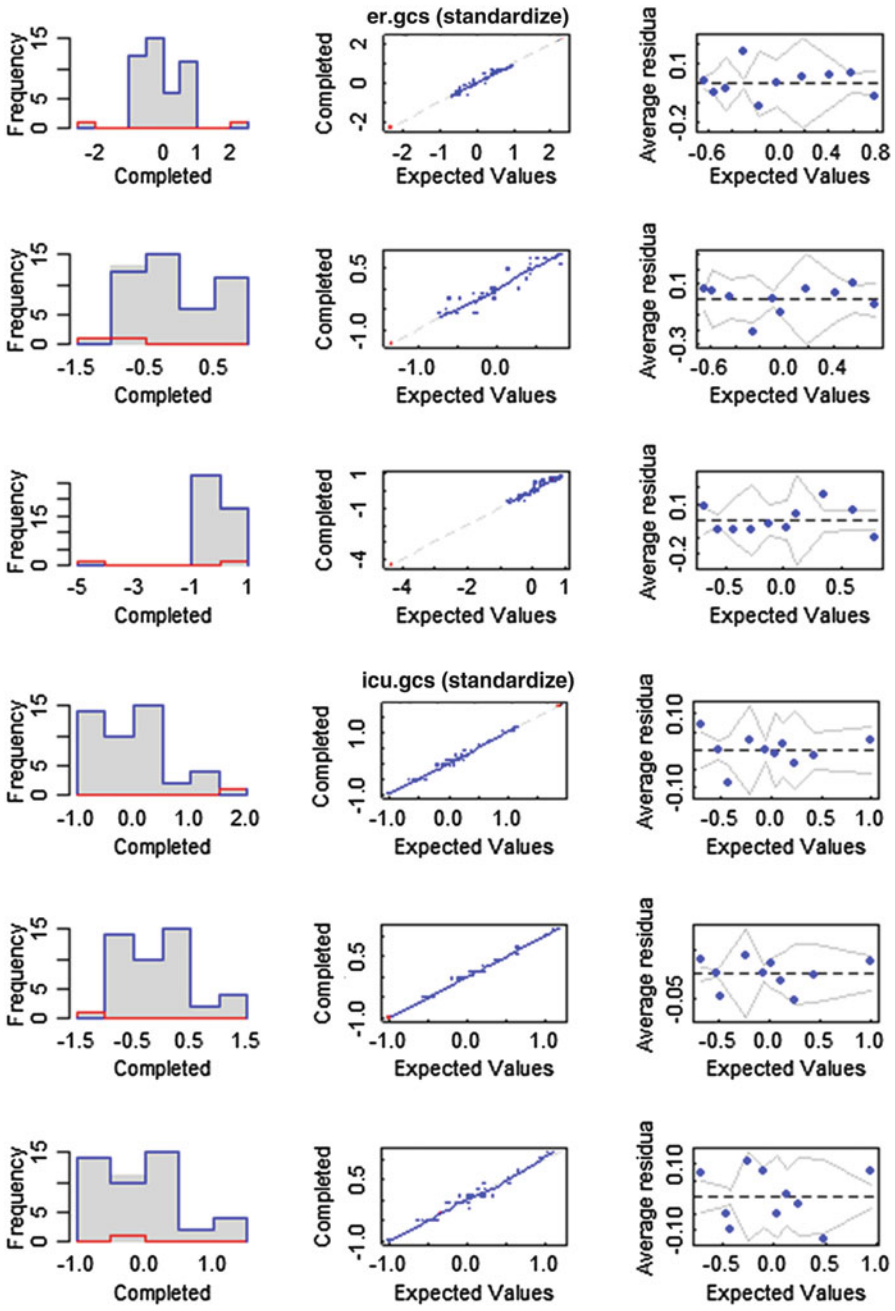

Fig. 3.24 (continued)

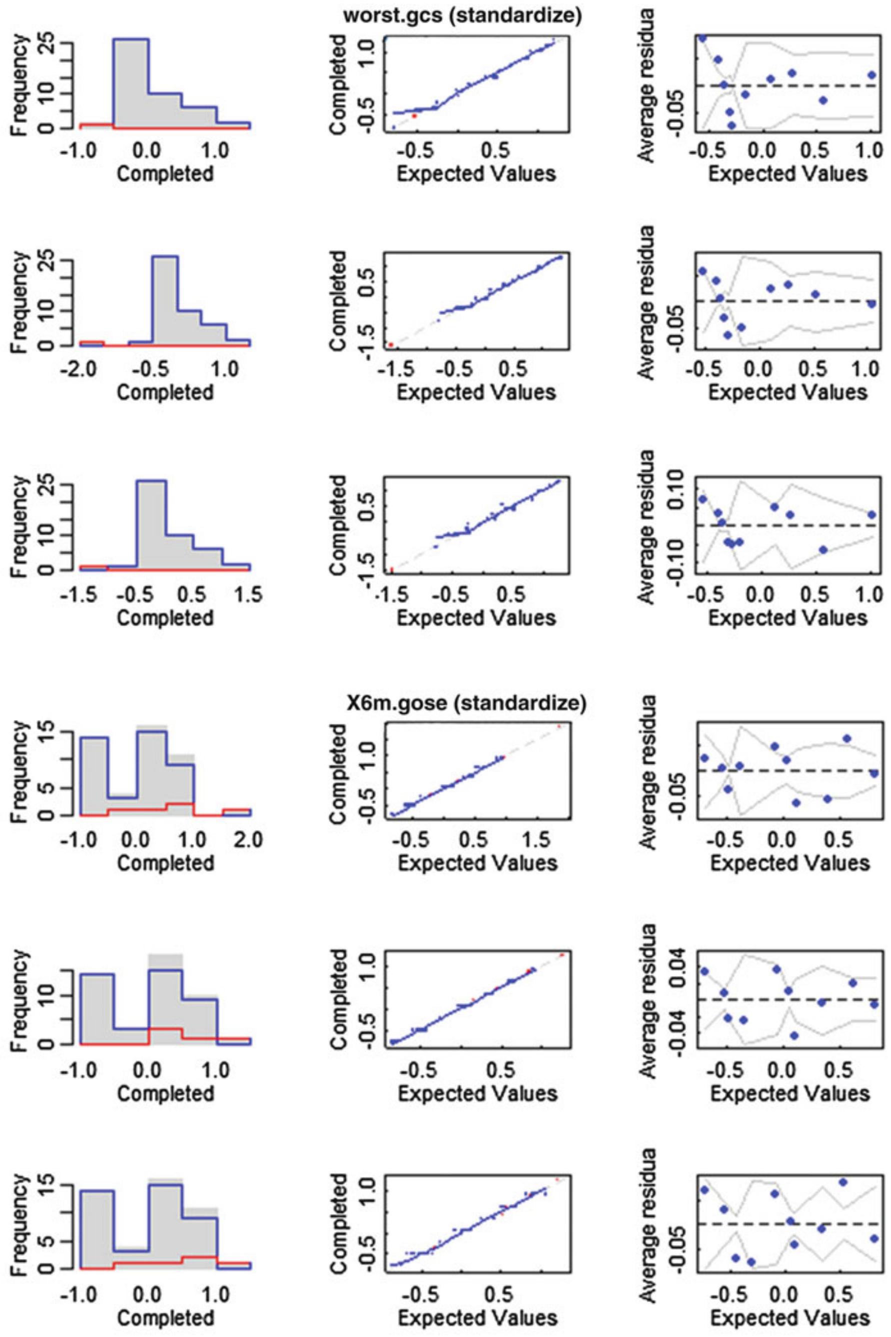

Fig. 3.24 (continued)

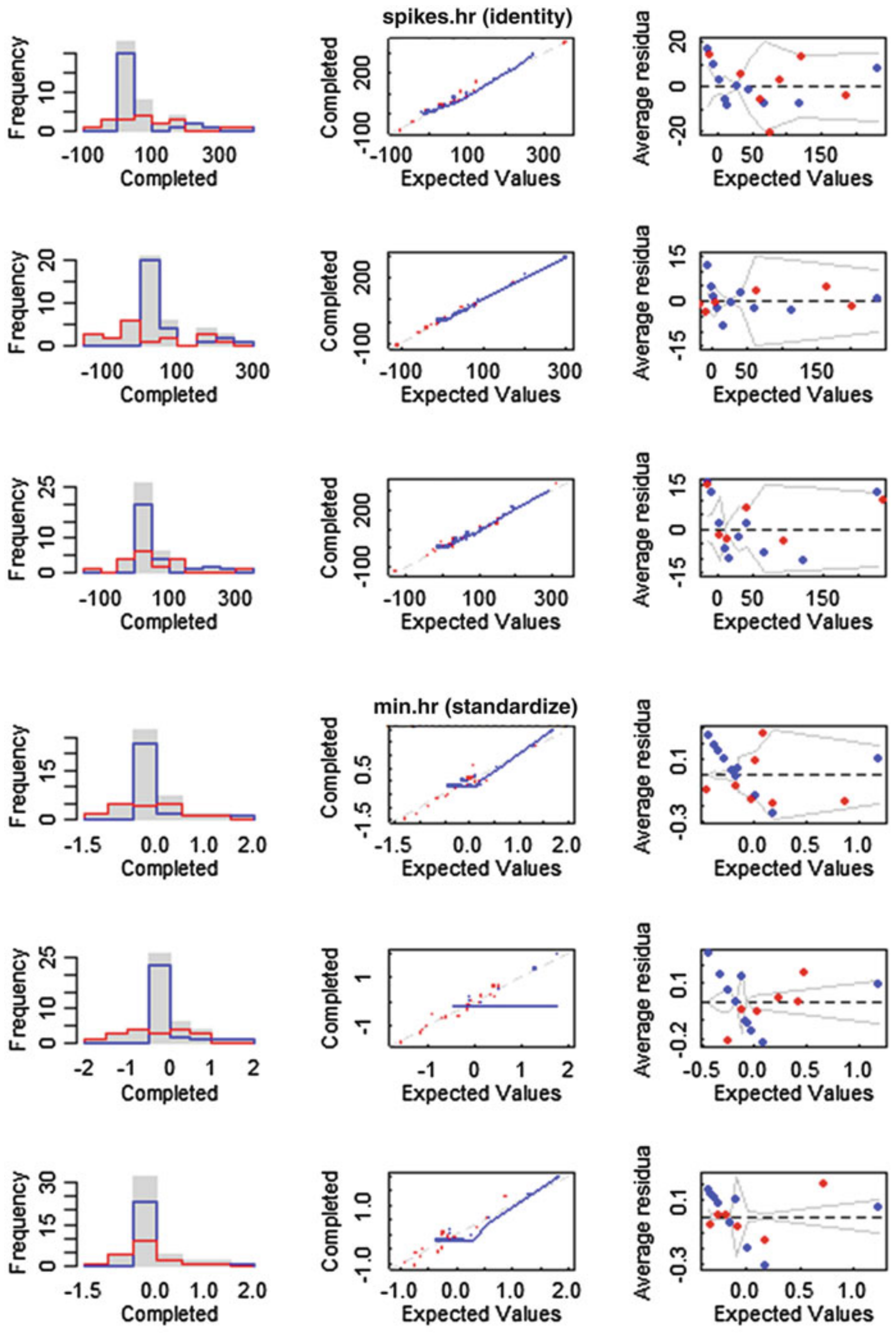

Fig. 3.24 (continued)

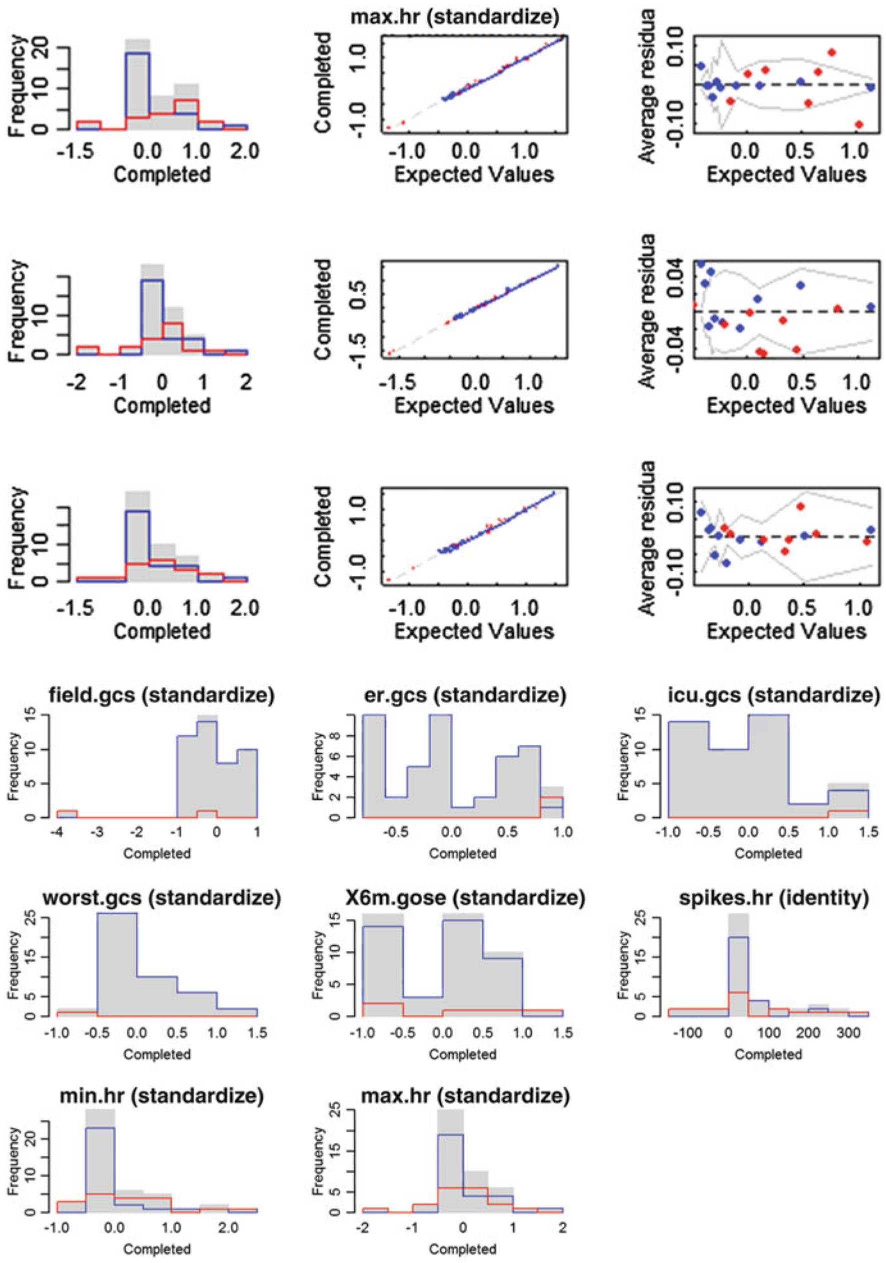

Fig. 3.24 (continued)

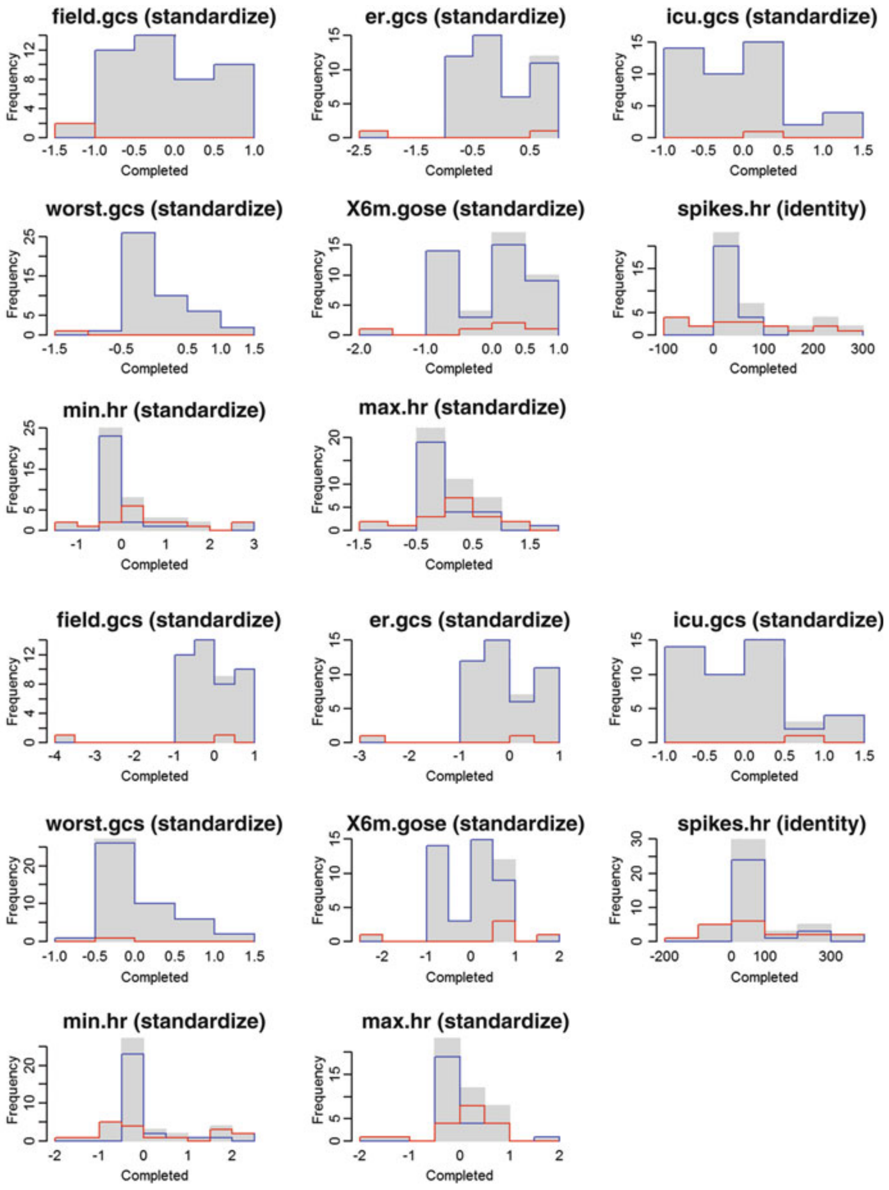

Fig. 3.24 (continued)

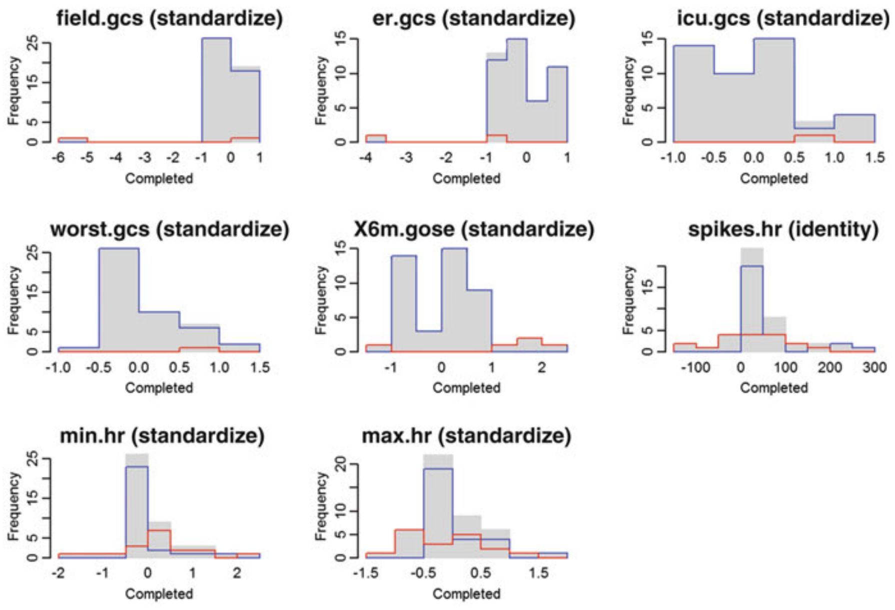

Fig. 3.24 (continued)

```
# should be similar for each of the k chains (in this case k=5).
# mipply is wrapper for sapply invoked on mi-class objects to

# compute the col means
round(mipply(imputations, mean, to.matrix = TRUE), 3)

##                     chain:1 chain:2 chain:3 chain:4 chain:5
## id                   23.500  23.500  23.500  23.500  23.500
## age                   0.000   0.000   0.000   0.000   0.000
## sex                   1.804   1.804   1.804   1.804   1.804
..
## missing_max.hr        0.391   0.391   0.391   0.391   0.391

# Rhat convergence statistics compares the variance between chains to the va
riance
# across chains.
# Rhat Values ~ 1.0 indicate likely convergence,
# Rhat Values > 1.1 indicate that the chains should be run longer
# (use large number of iterations)
Rhats(imputations, statistic = "moments") # assess the convergence of MI alg
orithm

## mean_field.gcs    mean_er.gcs   mean_icu.gcs mean_worst.gcs   mean_X6m.gose
##       1.858663       2.200902       1.484120       2.360286        1.752000
## mean_spikes.hr    mean_min.hr    mean_max.hr   sd_field.gcs       sd_er.gcs
##       1.972090       1.393025       1.743846       1.291126        1.479967
##     sd_icu.gcs    sd_worst.gcs    sd_X6m.gose   sd_spikes.hr       sd_min.hr
##       1.417884       1.861408       1.355365       1.514089        1.723325
##      sd_max.hr
##       1.497625
```

```
# When convergence is unstable, we can continue the iterations for

# all chains, e.g.
imputations <- mi(imputations, n.iter=20) # add additional 20 iterations

# To plot the produced mi results, for all missing_variables we can generate
# a histogram of the observed, imputed, and completed data.
# We can compare of the completed data to the fitted values

# implied by the model for the completed data, by plotting binned residuals.
# hist function works similarly as plot.
# image function gives a sense of the missingness patterns in the data

plot(imputations);hist(imputations)

image(imputations); summary(imputations)
```

```
## $id
## $id$is_missing
## [1] "all values observed"
##
## $id$observed
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00   12.25   23.50   23.50   34.75   46.00
##
##
## $age
## $age$is_missing
## [1] "all values observed"
##
## $age$observed
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.6045 -0.4019 -0.1126  0.0000  0.2997  1.3340
##
##
## $sex
## $sex$is_missing
## [1] "all values observed"
##
## $sex$observed
##
##  1  2
##  9 37
…
## $late.sz$observed
##
##  1  2
## 20 26
## $ever.sz
## $ever.sz$is_missing
## [1] "all values observed"
##
## $ever.sz$observed
##
##  1  2
## 19 27
```

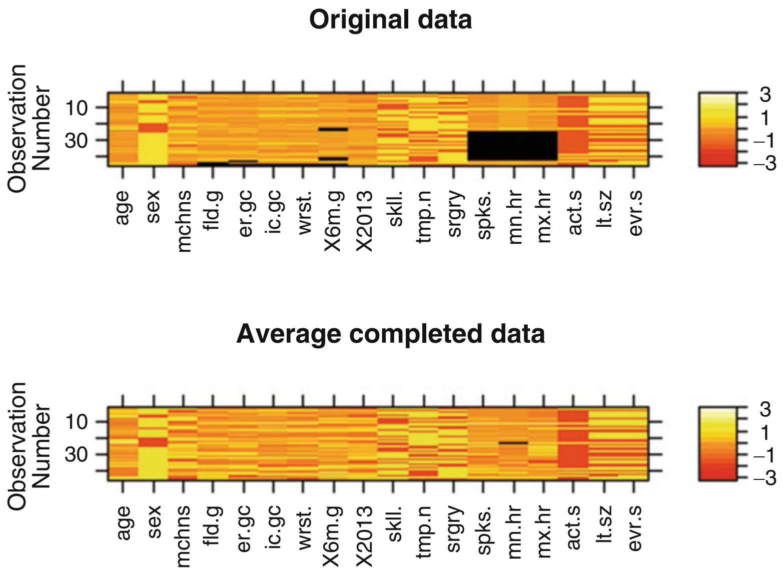

Fig. 3.25 Comparison of the missing data patterns in the original (top) and the completed (bottom) TBI sets

14. `Pool over them m = 5 imputed chains when fitting the "linear model". We can pool from across the five chains in order to estimate the final linear model` (Fig. 3.25) .

```
# regression model and impact of various predictors
model_results <- pool(ever.sz ~ surgery + worst.gcs + factor(sex) + age,
data =  imputations,  m=5); display (model_results); summary (model_results)

## bayesglm(formula = ever.sz ~ surgery + worst.gcs + factor(sex) +
##     age, data = imputations, m = 5)
##                  coef.est coef.se
## (Intercept)       0.58     1.35
## surgery1          0.99     0.66
## worst.gcs        -0.11     0.10
## factor(sex)Male -0.36     0.77
## age               0.00     0.02
## n = 41, k = 5
## residual deviance = 59.0, null deviance = 62.4 (difference = 3.4)

## Call:
## pool(formula = ever.sz ~ surgery + worst.gcs + factor(sex) +
##     age, data = imputations, m = 5)
##
## Deviance Residuals:
##     Min       1Q    Median       3Q      Max
## -1.6927  -1.1539   0.8245   1.0052   1.4009
##
```

```
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)      0.578917   1.348831   0.429    0.668
## surgery1         0.990656   0.662991   1.494    0.135
## worst.gcs       -0.105240   0.095335  -1.104    0.270
## factor(sex)Male -0.357285   0.772307  -0.463    0.644
## age              0.000198   0.019702   0.010    0.992
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 62.371  on 45  degrees of freedom
## Residual deviance: 58.995  on 41  degrees of freedom
## AIC: 68.995
##
## Number of Fisher Scoring iterations: 7
```

```
coef(summary(model_results))[, 1:2]  # get the model coef's and their SE's
```

```
##                      Estimate Std. Error
## (Intercept)      0.5789166170 1.34883106
## surgery1         0.9906554934 0.66299111
## worst.gcs       -0.1052399155 0.09533513
## factor(sex)Male -0.3572845034 0.77230674
## age              0.0001980042 0.01970153
```

```
# Report the summaries of the imputations
data.frames <- complete(imputations, 3)     # extract the first 3 chains
lapply(data.frames, summary)              # report summaries
```

```
## [[1]]
##        id             age            sex           mechanism
##  Min.   : 1.00   Min.   :16.00   Female: 9   Bike_vs_Auto: 4
##  1st Qu.:12.25   1st Qu.:23.00   Male  :37   Blunt       : 4
##  Median :23.50   Median :33.00               Fall        :13
##  Mean   :23.50   Mean   :36.89               GSW         : 2
##  3rd Qu.:34.75   3rd Qu.:47.25               MCA         : 7
##  Max.   :46.00   Max.   :83.00               MVA         :10
##                                              Peds_vs_Auto: 6
…
##  missing_max.hr
##  Mode :logical
##  FALSE:28
##  TRUE :18
##  NA's :0

## [[2]]
##        id             age            sex           mechanism
##  Min.   : 1.00   Min.   :16.00   Female: 9   Bike_vs_Auto: 4
##  1st Qu.:12.25   1st Qu.:23.00   Male  :37   Blunt       : 4
##  Median :23.50   Median :33.00               Fall        :13
##  Mean   :23.50   Mean   :36.89               GSW         : 2
##  3rd Qu.:34.75   3rd Qu.:47.25               MCA         : 7
##  Max.   :46.00   Max.   :83.00               MVA         :10
##                                              Peds_vs_Auto: 6
…
##  missing_max.hr
##  Mode :logical
##  FALSE:28
##  TRUE :18
##  NA's :0
##
## [[3]]
##        id            age             sex            mechanism
##  Min.   : 1.00   Min.   :16.00   Female: 9   Bike_vs_Auto: 4
##  1st Qu.:12.25   1st Qu.:23.00   Male  :37   Blunt       : 4
##  Median :23.50   Median :33.00               Fall        :13
##  Mean   :23.50   Mean   :36.89               GSW         : 2
##  3rd Qu.:34.75   3rd Qu.:47.25               MCA         : 7
##  Max.   :46.00   Max.   :83.00               MVA         :10
##                                              Peds_vs_Auto: 6
…
##
##  missing_max.hr
##  Mode :logical
##  FALSE:28
##  TRUE :18
##  NA's :0
```

3.13.3 Imputation via Expectation-Maximization

Below we present the theory and practice of one specific statistical computing strategy for imputing incomplete datasets.

Types of Missing Data

Recall that we have the following three distinct types of incomplete data.

- **MCAR**: Data which is Missing Completely At Random has nothing systematic about which observations are missing. There is no relationship between missingness and either observed or unobserved covariates.
- **MAR**: Missing At Random is weaker than MCAR. The missingness is still random, but solely due to the observed variables. For example, those from a lower socioeconomic status (SES) may be less willing to provide salary information (but we know their SES). The key is that the missingness is not due to the values which are not observed. MCAR implies MAR, but not vice-versa.
- **MNAR**: If the data are Missing Not At Random, then the missingness depends on the values of the missing data. Examples include censored data, self-reported data for individuals who are heavier, who are less likely to report their weight, and response-measuring device that can only measure values above 0.5, anything below that is missing.

General Idea of EM Algorithm

Expectation-Maximization (EM) is an iterative parameter estimation process involving two steps, *expectation* and *maximization*, which are applied in tandem. EM can be employed to find parameter estimates using maximum likelihood and is specifically useful when the equations determining the relations of the data-parameters cannot be directly solved. For example, a Gaussian mixture modeling assumes that each data point (X) has a corresponding latent (unobserved) variable or a missing value (Y), which may be specified as a mixture of coefficients determining the affinity of the data as a linear combination of Gaussian kernels, determined by a set of parameters (θ), e.g., means and variance-covariances. Thus, EM estimation relies on:

- An observed data set X,
- A set of missing (or latent) values Y,
- A parameter θ, which may be a vector of parameters,
- A likelihood function $L(\theta| X, Y) = p(X, Y| \theta)$, and
- The maximum likelihood estimate (MLE) of the unknown parameter(s) θ that is computed using the marginal likelihood of the observed data:

$$L(\theta|X) = p(X|\theta) = \int p(X, Y|\theta)dY.$$

Most of the time, this equation may not be directly solved, e.g., when Y is missing.

- *Expectation step (E step)*: computes the expected value of the *log likelihood function*, with respect to the conditional distribution of Z given X using the parameter estimates at the previous iteration (or at the position of initialization, for the first iteration), θ_t:

$$Q\left(\theta|\theta^{(t)}\right) = E_{Y|X,\theta^{(t)}}\left[log(L(\theta|X, Y)\right];$$

- *Maximization step (M step)*: Determine the parameters, θ, that maximize the expectation above,

$$\theta^{(t+1)} = \arg\max_{\theta} Q\left(\theta|\theta^{(t)}\right).$$

EM-Based Imputation

The EM algorithm is an alternative to Newton-Raphson, or the method of scoring, for computing MLE in cases where the complications in calculating the MLE are due to incomplete observation and data are MAR, missing at random, with separate parameters for observation and the missing data mechanism, so the missing data mechanism can be ignored.

$$\textbf{Complete Data: } Z = \begin{pmatrix} X \\ Y \end{pmatrix}, ZZ^T = \begin{pmatrix} XX^T & XY^T \\ YX^T & YY^T \end{pmatrix},$$

where X is the observed data and Y is the missing data.

- E-step: (Expectation) Get the expectations of Y and YY^T based on observed data.
- M-step: (Maximization) Maximize the conditional expectation in E-step to estimate the parameters.

Details: If $o = obs$ and $m = mis$ stand for observed and missing, the mean vector, $(\mu_{obs}, \mu_{mis})^T$, and the variance-covariance matrix, $\Sigma^{(t)} = \begin{pmatrix} \Sigma_{oo} & \Sigma_{om} \\ \Sigma_{mo} & \Sigma_{mm} \end{pmatrix}$, are represented by:

$$\mu^{(t)} = \begin{pmatrix} \mu_{obs} \\ \mu_{mis} \end{pmatrix}, \quad \Sigma^{(t)} = \begin{pmatrix} \Sigma_{oo} & \Sigma_{om} \\ \Sigma_{mo} & \Sigma_{mm} \end{pmatrix}$$

E-step:

$$E(Z|X) = \begin{pmatrix} X \\ E(Y|X) \end{pmatrix}, \quad E\left(ZZ^T|X\right) = \begin{pmatrix} XX^T & XE(Y|X)^T \\ E(Y|X)X^T & E(YY^T|X) \end{pmatrix}.$$

$$E(Y|X) = \mu_{mis} + \Sigma_{mo}\Sigma_{oo}^{-1}(X - \mu_{obs}).$$

$$E\left(YY^T|X\right) = \left(\Sigma_{mm} - \Sigma_{mo}\Sigma_{oo}^{-1}\Sigma_{om}\right) + E(Y|X)E(Y|X)^T.$$

M-step:

$$\mu^{(t+1)} = \frac{1}{n}\sum_{i=1}^{n} E(Z|X) \text{ and } \Sigma^{(t+1)} = \frac{1}{n}\sum_{i=1}^{n} E\left(ZZ^T|X\right) - \mu^{(t+1)}\mu^{(t+1)^T}.$$

A Simple Manual Implementation of EM-Based Imputation

To demonstrate the implementation of an EM-based imputation method from first principles, let's simulate 20 (feature) vectors of 200 (cases) Normal Distributed random values.

```r
set.seed(202227)
mu <- as.matrix(rep(2,20) )
sig <- diag(c(1:20) )
# Add a noise item. The noise is
#   $ \epsilon ~ MVN(as.matrix(rep(0,20)), diag(rep(1,20)))$
sim_data <- mvrnorm(n = 200, mu, sig) +
  mvrnorm(n=200, as.matrix(rep(0,20)), diag( rep(1,20) ))

# save these in the "original" object
sim_data.orig <- sim_data

# introduce 500 random missing indices (in the total of 4000=200*20)
# discrete distribution where the probability of the elements of values is p
roportional to probs, which are normalized to add up to 1.
rand.miss <- e1071::rdiscrete(500, probs = rep(1,length(sim_data)), values =
seq(1, length(sim_data)))
sim_data[rand.miss] <- NA
sum(is.na(sim_data))  # check now many missing (NA) are there < 500
## [1] 459

# cast the data into a data.frame object and report 15*10 elements
sim_data.df <- data.frame(sim_data)
kable( sim_data.df[1:15, 1:10], caption = "The first 15 rows and first 10 co
lumns of the simulation data")
```

The first 15 rows and first 10 columns of the simulation data are included below, mind the missing values, Table 3.1.

Now, let's define the EM imputation method function:

Table 3.1 Excerpt of 15 rows and 10 columns of the obfuscated simulation data

X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
0.1710559	4.8500284	1.7282025	NA	4.5511409	2.1699719	4.7118571	2.6972905	NA	−0.9250101
1.7721898	−1.6955535	0.4234295	6.2619764	−3.1339307	−1.8043889	1.2932913	0.9906410	1.0790366	1.1638161
2.2859237	4.2713494	3.8900670	3.5104578	3.0025607	3.9058619	3.0548309	NA	2.7208548	1.3601231
1.4906345	6.9253703	1.3387561	−2.6177675	−1.4074433	−2.0861790	−0.8847125	5.6485105	1.3348689	0.6512754
−0.0305062	2.3069337	NA	1.2474398	−0.2091862	1.1726298	1.4622491	0.6492390	0.1458781	7.2195026
1.9411674	3.6606019	6.3004943	0.3543817	1.0875460	1.3077307	0.0269935	−0.0112352	2.9724355	2.4616466
0.5209321	4.2225983	−0.6328796	NA	5.7103015	NA	4.5803136	0.2535158	−0.9758147	NA
3.3111727	0.1245427	0.9768425	4.7764851	4.9810860	−0.0438375	−0.3213523	4.3612903	2.7483497	3.7744105
2.1097813	0.2427543	1.1718823	3.6791032	3.7784795	5.5533794	1.8635898	0.1565858	4.6743012	−1.0232277
1.6203382	NA	1.3625543	1.4006816	−1.3467049	6.1636769	3.1703070	−3.4308300	0.2195447	−2.2921107
NA	0.5601779	NA	2.1652769	0.1060700	2.3763668	−3.7629196	−0.9718406	1.2196606	0.7038253
1.8570634	1.6714614	0.2000255	2.3308968	0.4593543	0.1716613	−0.5795159	−1.3327330	NA	1.2607465
2.5416914	1.0388186	4.7421120	2.3110187	NA	−1.5291668	2.4574501	NA	2.5364297	NA
0.9393925	1.8668513	3.4702117	1.6797620	0.7797323	2.5548284	−3.5498813	3.6595296	3.2124694	5.3120133
−0.8334973	NA	1.1949414	−0.6573305	1.7484169	−1.7743495	1.7168340	2.1065878	2.9478528	−0.5562349

```r
EM_algorithm <- function(x, tol = 0.001) {
  # identify the missing data entries (Boolean indices)
  missvals <- is.na(x)
  # instantialize the EM-iteration
  new.impute <- x
  old.impute <- x
  count.iter <- 1
  reach.tol <- 0

  # compute \Sigma on complete data
  sigma <- as.matrix(var(na.exclude(x)))
  # compute the vector of feature (column) means
  mean.vec <- as.matrix(apply(na.exclude(x), 2, mean))

  while (reach.tol != 1) {
    for (i in 1:nrow(x)) {
      pick.miss <- (c(missvals[i, ]))
      if (sum(pick.miss) != 0) {

        # compute invese-Sigma_completeData, variance-covariance matrix
        inv.S <- solve(sigma[!pick.miss, !pick.miss], tol = 1e-40)

        # Expectation Step
        # $$E(Y|X)=\mu_{mis}+\Sigma_{mo}\Sigma_{oo}^{-1}(X-\mu_{obs})$$
        new.impute[i, pick.miss] <- mean.vec[pick.miss] +
          sigma[pick.miss,!pick.miss] %*% inv.S %*%
          (t(new.impute[i, !pick.miss]) - t(t(mean.vec[!pick.miss])))
      }
    }

    # Maximization Step
    # Compute the complete \Sigma complete vector of feature (column) means
    # $$\Sigma^{(t+1)} = \frac{1}{n}\sum_{i=1}^nE(ZZ^T|X) -
      # \mu^{(t+1)}{\mu^{(t+1)}}^T$$
    sigma <- var((new.impute))
    #$$\mu^{(t+1)} = \frac{1}{n}\sum_{i=1}^nE(Z|X)$$
    mean.vec <- as.matrix(apply(new.impute, 2, mean))

    # Inspect for convergence tolerance, start with the 2nd iteration
    if (count.iter > 1) {
      for (l in 1:nrow(new.impute)) {
        for (m in 1:ncol(new.impute)) {
          if (abs((old.impute[l, m] - new.impute[l, m])) > tol) {
            reach.tol < -0
          } else {
            reach.tol <- 1
          }
        }
      }
    }
    count.iter <- count.iter + 1
    old.impute <- new.impute
  }
  # return the imputation output of the current iteration that passed the
tolerance level
  return(new.impute)
}

sim_data.imputed <- EM_algorithm(sim_data.df, tol=0.0001)
```

Plotting Complete and Imputed Data

Smaller black colored points represent observed data, and magenta-color and circle-shapes denote the imputated data (Fig. 3.26).

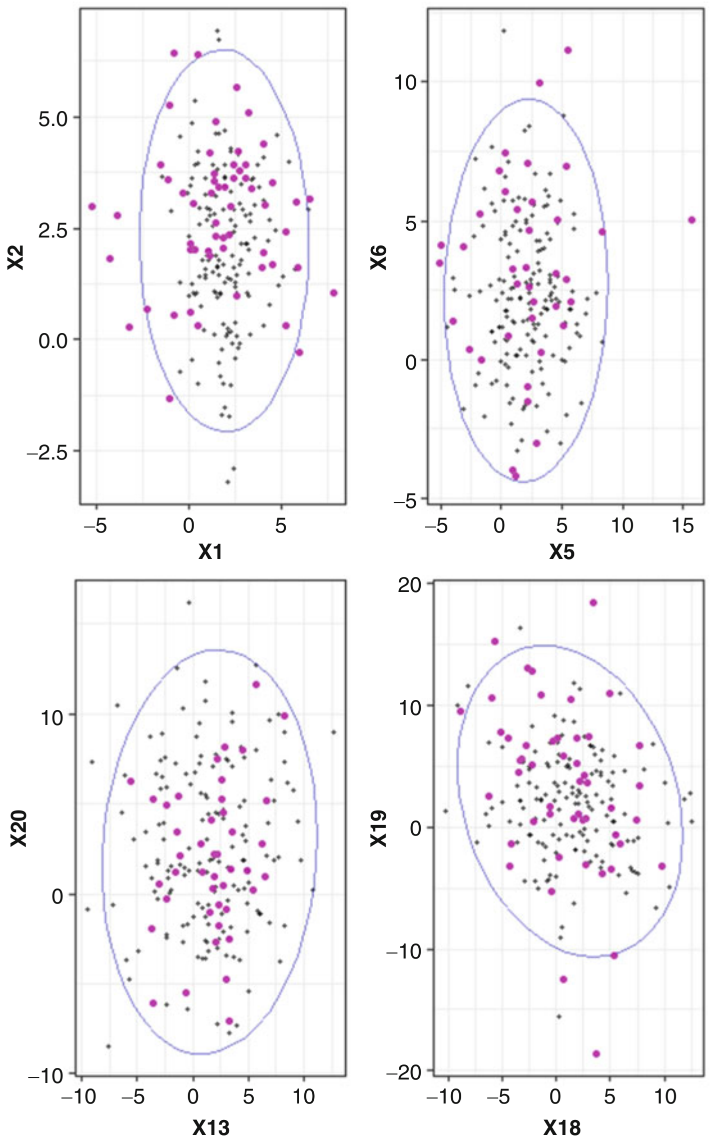

Fig. 3.26 Four scatterplots for pairs of features illustrating the complete data (small-black points), the imputed data points (larger-pink points), and 2D Gaussian kernels

```
plot.me <- function(index1, index2){
  plot.imputed <- sim_data.imputed[row.names(
    subset(sim_data.df, is.na(sim_data.df[, index1]) |
is.na(sim_data.df[, index2]))), ]
  p = ggplot(sim_data.imputed, aes_string( paste0("X",index1)  ,
paste0("X",index2 ))) +
  geom_point(alpha = 0.5, size = 0.7)+theme_bw() +
  stat_ellipse(type = "norm", color = "#000099", alpha=0.5) +
  geom_point(data = plot.imputed, aes_string( paste0("X",index1) ,
paste0("X",(index2))),size = 1.5, color = "Magenta", alpha = 0.8)
}

gridExtra::grid.arrange( plot.me(1,2), plot.me(5,6), plot.me(13,20),
plot.me(18,19), nrow = 2)
```

Validation of EM-Imputation Using the `Amelia` R Package

See this Amelia paper (https://gking.harvard.edu/files/gking/files/amelia_jss.pdf) and the corresponding R manual.

Comparison

Let's use the `amelia` function to impute the original data *sim_data_df* and compare the results to the simpler manual `EM_algorithm` imputation defined above.

```
# install.packages("Amelia")
library(Amelia)

dim(sim_data.df)

## [1] 200  20

amelia.out <- amelia(sim_data.df, m = 5)

## -- Imputation 1 --
##   1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
## -- Imputation 2 --
##   1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
## -- Imputation 3 --
##   1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
## -- Imputation 4 --
##   1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  21 22 23 24
## -- Imputation 5 --
##   1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17

amelia.out

##
## Amelia output with 5 imputed datasets.
## Return code:  1
## Message:  Normal EM convergence.
##
## Chain Lengths:
## --------------
## Imputation 1:  20
## Imputation 2:  15
## Imputation 3:  16
## Imputation 4:  24
## Imputation 5:  17

amelia.imputed.5 <- amelia.out$imputations[[5]]
```

- Magenta-color and circle-shape denote manual imputation via `EM_algorithm`
- Orange-color and square-shapes denote Amelia imputation (Figs. 3.27 and 3.28).

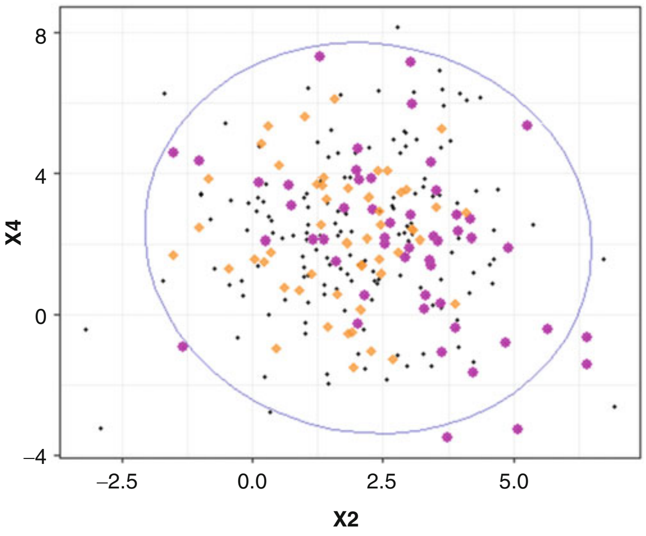

Fig. 3.27 Scatter plot of the second and fourth features. Magenta-circles and Orange-squares represent the manual imputation via EM_algorithm and the automated Amelia-based imputation

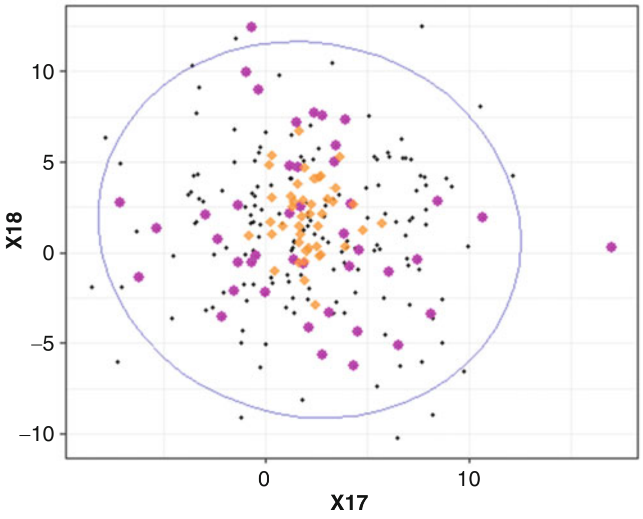

Fig. 3.28 Same as Fig. 3.27, for features 17 and 18

```
plot.me2 <- function(index1, index2){
  plot.imputed <- sim_data.imputed[row.names(
    subset(sim_data.df, is.na(sim_data.df[, index1]) |
is.na(sim_data.df[, index2]))), ]
  plot.imputed2 <- amelia.imputed.5[row.names(
  subset(sim_data.df, is.na(sim_data.df[, index1]) |
is.na(sim_data.df[, index2]))), ]
  p = ggplot(sim_data.imputed, aes_string( paste0("X",index1)  ,
paste0("X",index2 ))) +
  geom_point(alpha = 0.8, size = 0.7)+theme_bw() +
  stat_ellipse(type = "norm", color = "#000099", alpha=0.5) +
  geom_point(data = plot.imputed, aes_string( paste0("X",index1) ,
paste0("X",(index2))),size=2.5, color="Magenta", alpha=0.9, shape=16) +
    geom_point(data = plot.imputed2, aes( X1 , X2),size = 2.5,
color = "#FF9933", alpha = 0.8, shape = 18)
  return(p)
}

plot.me2(2, 4)
```

```
plot.me2(17, 18)
```

Density Plots

Finally, we can compare the densities of the original, manually-imputed and Amelia-imputed datasets. Remember that in this simulation, we had about 500 observations missing out of the 4000 that we synthetically generated (Fig. 3.29).

3.14 Parsing Webpages and Visualizing Tabular HTML Data

In this section, we will utilize the Earthquakes dataset on the SOCR website. It stores information about earthquakes of magnitudes larger than 5 on the Richter scale that were recorded between 1969 and 2007. Here is how we download and parse the data on the source webpage and ingest the information into R:

```
# install.packages("xml2")
library("XML"); library("xml2")
library("rvest")
wiki_url <- read_html("http://wiki.socr.umich.edu/index.php/SOCR_Data_Dinov_
021708_Earthquakes")
html_nodes(wiki_url, "#content")

## {xml_nodeset (1)}
## [1] <div id="content" class="mw-body-primary" role="main">\n\t<a id="top
...

earthquake<- html_table(html_nodes(wiki_url, "table")[[2]])
```

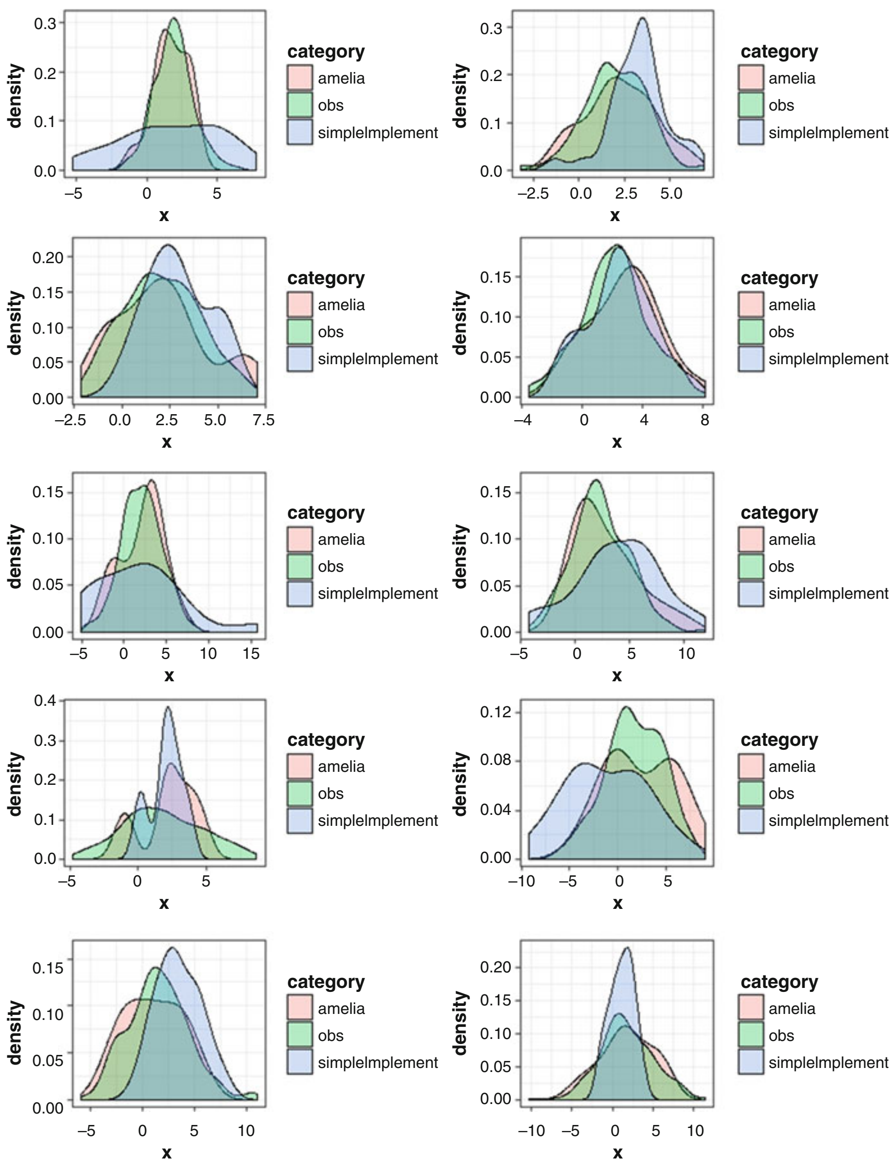

Fig. 3.29 Density plots of the original, manually-imputed and Amelia-imputed datasets, 10 features only

In this dataset, `Magt` (magnitude type) may be used as grouping variable. We will draw a “Longitude vs. Latitude” line plot from this dataset. The function we are using is called `ggplot()`, available from R package `ggplot2`. The input type for this function is a data frame, and `aes()` specifies the axes (Fig. 3.30).

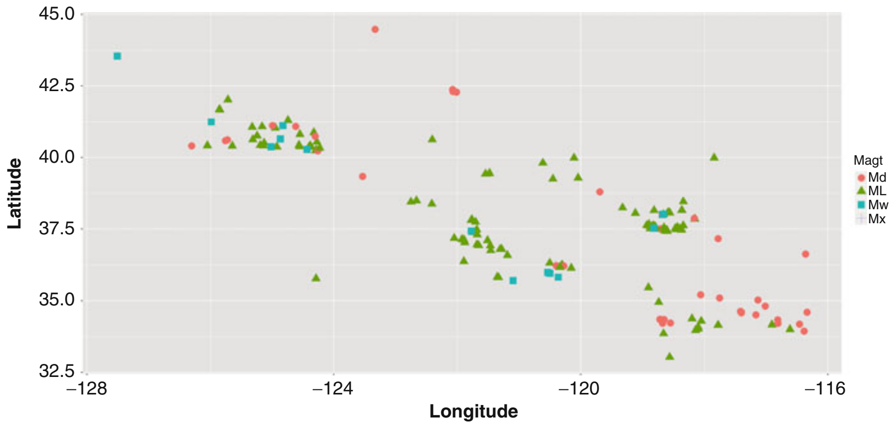

Fig. 3.30 Earthquake data plot of magnitude type (color/shape) against longitude (x) and latitude (y)

```
library(ggplot2)
plot4<-ggplot(earthquake, aes(Longitude, Latitude, group=Magt, color=Magt))+
  geom_point(data=earthquake, size=4, mapping=aes(x=Longitude, y=Latitude,
shape=Magt))
plot4  # or plint(plot4)
```

We can see the plotting script consists of two parts. The first part `ggplot (earthquake, aes(Longitude, Latitude, group = Magt, color=Magt))` specifies the setting of the plot: dataset, group and color. The second part specifies that we are going to draw points for all data points. In later Chapters, we will frequently use `ggplot2`; which always takes multiple function calls, e.g., `function1 + function2`.

We can visualize the distribution for different variables using density plots. The following chunk of codes plots the distribution for Latitude among different Magnitude types. Also, it uses the `ggplot()` function combined with `geom_density()` (Fig. 3.31).

```
plot5<-ggplot(earthquake, aes(Latitude, size=1))+
     geom_density(aes(color=Magt))
plot5
```

We can also compute and display 2D Kernel Density and 3D Surface Plots. Plotting 2D Kernel Density and 3D Surface plots is very important and useful in multivariate exploratory data analytic.

We will use the `plot_ly()` function under `plotly` package, which takes data frame inputs.

To create a surface plot, we use two vectors: *x* and *y*, with length *m* and *n*, respectively. We also need a matrix: *z* of size $m \times n$. This *z* matrix is created from matrix multiplication between *x* and *y*.

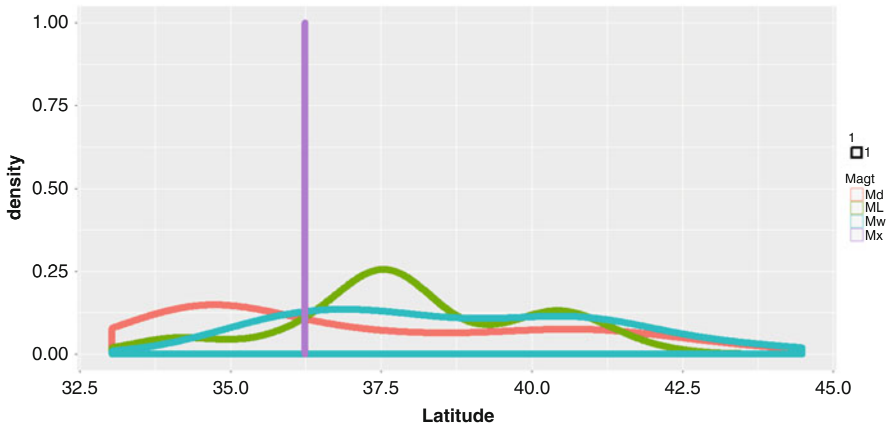

Fig. 3.31 Modified Earthquake density plot (y) of magnitude type against latitude coordinates (x)

The kde2d() function is needed for 2D kernel density estimation.

```
kernal_density <- with(earthquake, MASS::kde2d(Longitude, Latitude, n = 50))
```

Here, z is an estimate of the kernel density function. Then, we apply plot_ly to the list kernal_density via the with() function.

```
library(plotly)

with(kernal_density, plot_ly(x=x, y=y, z=z, type="surface"))
```

Note that we used the option "surface", however you can experiment with the type option.

Alternatively, one can plot 1D, 2D, or 3D plots (Fig. 3.32):

```
plot_ly(x = ~ earthquake$Longitude)

## No trace type specified:
##   Based on info supplied, a 'histogram' trace seems appropriate.
##   Read more about this trace type->https://plot.ly/r/reference/#histogram

plot_ly(x = ~ earthquake$Longitude, y = ~earthquake$Latitude)

plot_ly(x=~earthquake$Longitude, y=~earthquake$Latitude, z=~earthquake$Mag)

df3D <- data.frame(x=earthquake$Longitude, y=earthquake$Latitude,
z=earthquake$Mag)

# Convert he Long (X, Y, Z) Earthquake format data into a Matrix Format
# install.packages("Matrix")
library("Matrix")
matrix_EarthQuakes <- with(df3D, sparseMatrix(i = as.numeric(180-x),
```

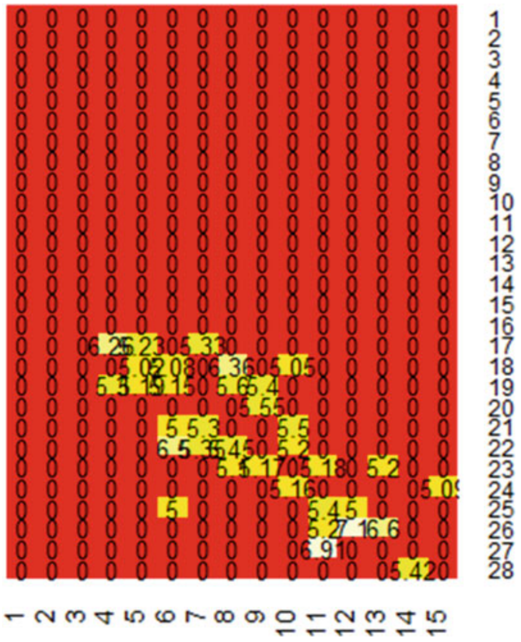

Fig. 3.32 Image representation of the kernel-density estimation of the Earthquake magnitude rendered as a heatmap

```
j=as.numeric(y), x=z, use.last.ij=T, dimnames=list(levels(x), levels(y))))
dim(matrix_EarthQuakes)

## [1] 307  44

View(as.matrix(matrix_EarthQuakes))

# view matrix is 2D heatmap :
library("ggplot2"); library("gplots")

heatmap.2( as.matrix(matrix_EarthQuakes[280:307, 30:44]), Rowv=FALSE,
Colv=FALSE, dendrogram='none', cellnote=as.matrix(matrix_EarthQuakes[
280:307, 30:44]), notecol="black", trace='none', key=FALSE, lwid =
c(.01, .99), lhei = c(.01, .99), margins = c(5, 15 ))
```

```
# Long -180<x<-170, Lat: 30<y<45, Z: 5<Mag<8
matrix_EarthQuakes <- with(df3D, sparseMatrix(i = as.numeric(180+x),
j=as.numeric(y), x=z,use.last.ij=TRUE,dimnames=list(levels(x), levels(y))))
mat1 <- as.matrix(matrix_EarthQuakes)
plot_ly(z = ~mat1, type = "surface")

# To plot the Aggregate (Summed) Magnitudes at all Long/Lat:
matrix_EarthQuakes <- with(df3D, sparseMatrix(i = as.numeric(180+x),
j=as.numeric(y), x=z, dimnames=list(levels(x), levels(y))))
mat1 <- as.matrix(matrix_EarthQuakes)
plot_ly(z = ~mat1, type = "surface")
# plot_ly(z = ~mat1[30:60, 20:40], type = "surface")
```

http://www.socr.umich.edu/people/dinov/courses/DSPA_notes/02_ManagingData.html

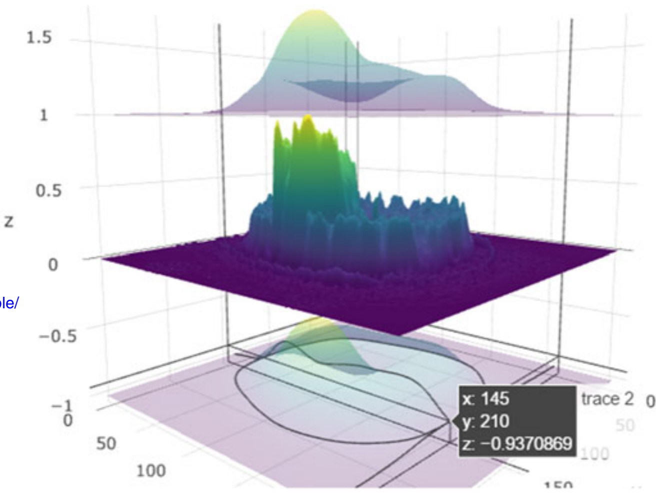

Fig. 3.33 Live demo of 3D kernel density surface plots using the Earthquake and 2D brain imaging data (http://www.socr.umich.edu/people/dinov/courses/DSPA_notes/02_ManagingData.html)

You can see interactive surface plot generated by plotly in the live demo listed on Fig. 3.33.

3.15 Cohort-Rebalancing (for Imbalanced Groups)

Comparing cohorts with imbalanced sample sizes (unbalanced designs) may present hidden biases in the results. Frequently, a cohort-rebalancing protocol is necessary to avoid such unexpected effects. Extremely unequal sample sizes can invalidate various parametric assumptions (e.g., homogeneity of variances). Also, there may be insufficient data representing the patterns belonging to the minority class(es) leading to inadequate capturing of the feature distributions. Although, the groups do not have to have equal sizes, a general rule of thumb is $0.5 < \frac{size1}{size2} < 2$. Tat is group sizes where one group is more than an order of magnitude larger than the size of another group has the `potential` for bias.

Example 1 Parkinson's Diseases Study involving neuroimaging, genetics, clinical, and phenotypic data for over 600 volunteers produced multivariate data for three cohorts: *HC=Healthy Controls(166)*, *PD=Parkinson's (434)*, *SWEDD = subjects without evidence for dopaminergic deficit (61)* (Figs. 3.34 and 3.35).

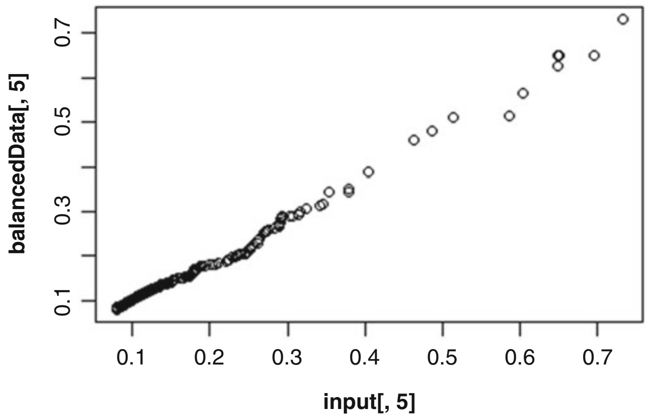

Fig. 3.34 Validation that cohort rebalancing does not substantially alter the distributions of features. This QQ plot of one variable shows the linearity of the quantiles of the initial (x) and rebalanced (y) data

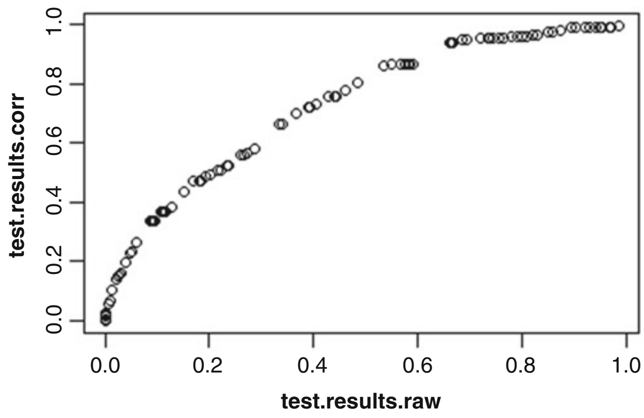

Fig. 3.35 Scatter plot of the raw (x) and corrected/adjusted (y) p-values corresponding to the paired two-sample Wilcoxon non-parametric test comparing the raw and rebalanced features

```
# update.packages()
# load the data: 06_PPMI_ClassificationValidationData_Short.csv
ppmi_data <-read.csv("https://umich.instructure.com/files/330400/download?do
wnload_frd=1", header=TRUE)

table(ppmi_data$ResearchGroup)

# binarize the Dx classes
ppmi_data$ResearchGroup <- ifelse(ppmi_data$ResearchGroup == "Control",
"Control", "Patient")
attach(ppmi_data)

head(ppmi_data)
```

```
# Model-free analysis, classification
# install.packages("crossval")
# install.packages("ada")
# library("crossval")
require(crossval)
require(ada)
#set up adaboosting prediction function

# Define a new classification result-reporting function
my.ada <- function (train.x, train.y, test.x, test.y, negative, formula)
{
  ada.fit <- ada(train.x, train.y)
  predict.y <- predict(ada.fit, test.x)
  #count TP, FP, TN, FN, Accuracy, etc.
  out <- confusionMatrix(test.y, predict.y, negative = negative)
 # negative  is the label of a negative "null" sample (default: "control").
  return (out)
}

# balance cases
# SMOTE: Synthetic Minority Oversampling Technique to handle class
misbalance in binary classification.
set.seed(1000)
# install.packages("unbalanced") to deal with unbalanced group data
require(unbalanced)
ppmi_data$PD <- ifelse(ppmi_data$ResearchGroup=="Control", 1, 0)
uniqueID <- unique(ppmi_data$FID_IID)
ppmi_data <- ppmi_data[ppmi_data$VisitID==1, ]
ppmi_data$PD <- factor(ppmi_data$PD)

colnames(ppmi_data)
# ppmi_data.1<-ppmi_data[, c(3:281, 284, 287, 336:340, 341)]
n <- ncol(ppmi_data)
output.1 <- ppmi_data$PD

# remove Default Real Clinical subject classifications!
ppmi_data$PD <- ifelse(ppmi_data$ResearchGroup=="Control", 1, 0)
input <- ppmi_data[ , -which(names(ppmi_data) %in% c("ResearchGroup",
"PD", "X", "FID_IID"))]
# output <- as.matrix(ppmi_data[ , which(names(ppmi_data) %in% {"PD"})])
output <- as.factor(ppmi_data$PD)
c(dim(input), dim(output))

#balance the dataset
data.1<-ubBalance(X= input, Y=output, type="ubSMOTE", percOver=300,
percUnder=150, verbose=TRUE)

# percOver = A number that drives the decision of how many extra cases from
the minority class are generated (known as over-sampling).
# k = A number indicating the number of nearest neighbors that are used to g
enerate the new examples of the minority class.
# percUnder = A number that drives the decision of how many extra cases from
the majority classes are selected for each case generated from the minority
class (known as under-sampling)
```

```
balancedData<-cbind(data.1$X, data.1$Y)
table(data.1$Y)

nrow(data.1$X); ncol(data.1$X)
nrow(balancedData); ncol(balancedData)
nrow(input); ncol(input)

colnames(balancedData) <- c(colnames(input), "PD")

# check visually for differences between the distributions of the raw
(input) and rebalanced data (for only one variable, in this case)
qqplot(input[, 5], balancedData [, 5])
```

```
###Check balance
## Wilcoxon test
alpha.0.05 <- 0.05
test.results.bin <- NULL        # binarized/dichotomized p-values
test.results.raw <- NULL        # raw p-values

for (i in 1:(ncol(balancedData)-1))
{
    test.results.raw[i]<-wilcox.test(input[, i], balancedData [, i])$p.value
    test.results.bin [i] <- ifelse(test.results.raw [i] > alpha.0.05, 1, 0)
    print(c("i=", i, "Wilcoxon-test=", test.results.raw [i]))
}
print(c("Wilcoxon test results: ", test.results.bin))

test.results.corr <- stats::p.adjust(test.results.raw, method = "fdr", n = l
ength(test.results.raw))
# where methods are "holm", "hochberg", "hommel", "bonferroni", "BH", "BY",
"fdr", "none")
plot(test.results.raw, test.results.corr)
```

```
# zeros (0) are significant independent between-group T-test differences,
ones (1) are insignificant

# Check the Differences between the rate of significance between the raw and
FDR-corrected p-values
test.results.bin <- ifelse(test.results.raw > alpha.0.05, 1, 0)
table(test.results.bin)
test.results.corr.bin <- ifelse(test.results.corr > alpha.0.05, 1, 0)
table(test.results.corr.bin)
```

3.16 Appendix

3.16.1 Importing Data from SQL Databases

We can also import SQL databases into R. First, we need to install and load the RODBC(R Open Database Connectivity) package.

```
install.packages("RODBC", repos = "http://cran.us.r-project.org")
library(RODBC)
```

Then, we could open a connection to the SQL server database with Data Source Name (DSN), via Microsoft Access. More details are provided online.

3.16.2 R Code Fragments

Below are some code snippets used to generate some of the graphs shown in this Chapter.

```r
#Right Skewed
N <- 10000
 x <- rnbinom(N, 10, .5)
 hist(x,
 xlim=c(min(x), max(x)), probability=T, nclass=max(x)-min(x)+1,
   col='lightblue', xlab=' ', ylab=' ', axes=F,
   main='Right Skewed')
lines(density(x, bw=1), col='red', lwd=3)

#No Skew
N <- 10000
 x <- rnorm(N, 0, 1)
 hist(x, probability=T,
   col='lightblue', xlab=' ', ylab=' ', axes=F,
   main='No Skew')
lines(density(x, bw=0.4), col='red', lwd=3)

#Uniform density
x<-runif(1000, 1, 50)
hist(x, col='lightblue', main="Uniform Distribution", probability = T, xlab=
"", ylab="Density", axes=F)
abline(h=0.02, col='red', lwd=3)

#68-95-99.7 rule
x <- rnorm(N, 0, 1)
 hist(x, probability=T,
   col='lightblue', xlab=' ', ylab=' ', axes = F,
   main='68-95-99.7 Rule')
lines(density(x, bw=0.4), col='red', lwd=3)
axis(1, at=c(-3, -2, -1, 0, 1, 2, 3), labels = expression(mu-3*sigma,
mu-2*sigma, mu-sigma, mu, mu+sigma, mu+2*sigma, mu+3*sigma))
abline(v=-1, lwd=3, lty=2)
abline(v=1, lwd=3, lty=2)
abline(v=-2, lwd=3, lty=2)
abline(v=2, lwd=3, lty=2)
abline(v=-3, lwd=3, lty=2)
abline(v=3, lwd=3, lty=2)
text(0, 0.2, "68%")
segments(-1, 0.2, -0.3, 0.2, col = 'red', lwd=2)
segments(1, 0.2, 0.3, 0.2, col = 'red', lwd=2)
text(0, 0.15, "95%")
segments(-2, 0.15, -0.3, 0.15, col = 'red', lwd=2)
segments(2, 0.15, 0.3, 0.15, col = 'red', lwd=2)
text(0, 0.1, "99.7%")
segments(-3, 0.1, -0.3, 0.1, col = 'red', lwd=2)
segments(3, 0.1, 0.3, 0.1, col = 'red', lwd=2)
```

3.17 Assignments: 3. Managing Data in R

3.17.1 Import, Plot, Summarize and Save Data

Load the following two datasets, generate summary statistics for all variables, plot some of the features (e.g., histograms, box plots, density plots, etc.), and save the data locally as CSV files:

- ALS case-study data, https://umich.instructure.com/courses/38100/files/folder/Case_Studies/15_ALS_CaseStudy.
- SOCR Knee Pain Data, http://wiki.socr.umich.edu/index.php/SOCR_Data_KneePainData_041409.

3.17.2 Explore some Bivariate Relations in the Data

Use ALS case-study data or SOCR Knee Pain Data to explore some bivariate relations (e.g. bivariate plot, correlation, table, crosstable, etc.)

Use 07_UMich_AnnArbor_MI_TempPrecipitation_HistData_1900_2015 data to show the relations between temperature and time. [Hint: use `geom_line` or `geom_bar`].

Some sample code for dealing with the table of temperatures data is included below.

```
<code>
  Temp_Data <- as.data.frame(read.csv("https://umich.instructure.com/files/706163/download?
download_frd=1", header=T, na.strings=c("", ".", "NA", "NR")))
  summary(Temp_Data)
  # View(Temp_Data); colnames(Temp_Data)

  # Wide-to-Long transformation: reshape arguments include
  # (1) list of variable names that define the different times or metrics (varying),
  # (2) the name we wish to give the variable containing these values in our long dataset (
v.names),
  # (3) the name we wish to give the variable describing the different times or metrics (ti
mevar),
  # (4) the values this variable will have (times), and
  # (5) the end format for the data (direction)
  # Before reshaping make sure all data types are the same as putting them in 1 column will
  # otherwise generate inconsistencies/errors
  colN <- colnames(Temp_Data[,-1])
  longTempData <- reshape(Temp_Data, varying = colN, v.names = "Temps", timevar="Months", t
imes = colN, direction = "long")

  # View(longTempData)
  bar2 <- ggplot(longTempData, aes(x = Months, y = Temps, fill = Months)) +
  geom_bar(stat = "identity")
  print(bar2)
  bar3 <- ggplot(longTempData, aes(x = Year, y = Temps, fill = Months)) +
  geom_bar(stat = "identity")
  print(bar3)

  p <- ggplot(longTempData, aes(x=Year, y=as.integer(Temps), colour=Months)) +
  geom_line()
  p
</code>
```

3.17.3 Missing Data

Introduce (artificially) some missing data, impute the missing values and examine the differences between the original, incomplete, and imputed data.

3.17.4 Surface Plots

Generate a surface plot for the (`RF`) Knee Pain data illustrating the 2D distribution of locations of the patient reported knee pain (use *plot_ly* and kernel density estimation).

3.17.5 Unbalanced Designs

Rebalance the groups of ALS (training data) patients according to $Age > 50$ and $Age \leq 50$ using synthetic minoroty oversampling (SMOTE) to ensure approximately equal cohort sizes. (Hint: you may need to set `1` as the minority class.)

3.17.6 Aggregate Analysis

Use the California Ozone Data to generate a summary report. Make sure to include: summary for every variable, the structure of the data, convert to appropriate data type, discuss the tendency of the ozone average concentration, explore the differences of the ozone concentration a specific area (you may select year 2006), explore the seasonal change of ozone concentration.

References

https://plot.ly/r/
http://www.statmethods.net/management