6/2/2022

Classification Project

AM: P3622004

Author: Galanakis Michalis

Classification Project

Michalis Galanakis

6/2/2022

Defining the objective

The following Dataset involves predicting the onset of diabetes within 5 years in a women population given medical details. It is a binary (2-class) classification problem. The number of observations for each class is not balanced. There are 768 observations with 8 input variables and 1 output variable. Missing values are believed to be encoded with zero values. The variable names are as follows:

Number of times pregnant.

Plasma glucose concentration a 2 hours in an oral glucose tolerance test.

Diastolic blood pressure (mm Hg).

Triceps skinfold thickness (mm).

2-Hour serum insulin (mu U/ml).

Body mass index (weight in kg/(height in m)^2).

Diabetes pedigree function.

Age (years).

Class variable (0 or 1).

```
# Goal: The data in the folder named data.txt refer to counts from
different variables in a population of women, aiming to gain useful
insights and predict the ones that will develop diabetes in due time (0
refers to the women that won't develop diabetes, 1 refers to the women
that will develop diabetes)
```

Read in the data

```
data <- read.table('C:/Users/mihal/OneDrive/data.txt',sep=",")
```

Descriptive statistics

```
str(data)
```

```
## 'data.frame':    768 obs. of  9 variables:
##  $ V1: int  6 1 8 1 0 5 3 10 2 8 ...
##  $ V2: int  148 85 183 89 137 116 78 115 197 125 ...
##  $ V3: int  72 66 64 66 40 74 50 0 70 96 ...
##  $ V4: int  35 29 0 23 35 0 32 0 45 0 ...
##  $ V5: int  0 0 0 94 168 0 88 0 543 0 ...
```

```
##  $ V6: num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
##  $ V7: num  0.627 0.351 0.672 0.167 2.288 ...
##  $ V8: int  50 31 32 21 33 30 26 29 53 54 ...
##  $ V9: int  1 0 1 0 1 0 1 0 1 1 ...

dim(data)

## [1] 768   9

summary(data)

##        V1               V2              V3               V4
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00
##  Median : 3.000   Median :117.0   Median : 72.00   Median :23.00
##  Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54
##  3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
##        V5              V6              V7               V8
##  Min.   :  0.0   Min.   : 0.00   Min.   :0.0780   Min.   :21.00
##  1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437   1st Qu.:24.00
##  Median : 30.5   Median :32.00   Median :0.3725   Median :29.00
##  Mean   : 79.8   Mean   :31.99   Mean   :0.4719   Mean   :33.24
##  3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262   3rd Qu.:41.00
##  Max.   :846.0   Max.   :67.10   Max.   :2.4200   Max.   :81.00
##        V9
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :0.000
##  Mean   :0.349
##  3rd Qu.:1.000
##  Max.   :1.000

colnames(data) <-
c("t.pregnant","plasma","bl.press","tr.thick","serum.ins","bmi","diab",
"age","class")
head(data,5)

##   t.pregnant plasma bl.press tr.thick serum.ins  bmi  diab age class
## 1          6    148       72       35         0 33.6 0.627  50     1
## 2          1     85       66       29         0 26.6 0.351  31     0
## 3          8    183       64        0         0 23.3 0.672  32     1
## 4          1     89       66       23        94 28.1 0.167  21     0
## 5          0    137       40       35       168 43.1 2.288  33     1

tail(data,5)

##     t.pregnant plasma bl.press tr.thick serum.ins  bmi  diab age
class
## 764         10    101       76       48       180 32.9 0.171  63
0
## 765          2    122       70       27         0 36.8 0.340  27
```

```
0
## 766          5    121       72       23       112 26.2 0.245  30
0
## 767          1    126       60        0         0 30.1 0.349  47
1
## 768          1     93       70       31         0 30.4 0.315  23
0

class <- data$class
t.pregnant <- data$t.pregnant


expl_data <- data[,2:8]
# We assume that the zeros in the variable times.pregnant are not
missing, hence we don't replace zeros with "NA"

head(expl_data,10)

##    plasma bl.press tr.thick serum.ins  bmi  diab age
## 1     148       72       35         0 33.6 0.627  50
## 2      85       66       29         0 26.6 0.351  31
## 3     183       64        0         0 23.3 0.672  32
## 4      89       66       23        94 28.1 0.167  21
## 5     137       40       35       168 43.1 2.288  33
## 6     116       74        0         0 25.6 0.201  30
## 7      78       50       32        88 31.0 0.248  26
## 8     115        0        0         0 35.3 0.134  29
## 9     197       70       45       543 30.5 0.158  53
## 10    125       96        0         0  0.0 0.232  54

expl_data[expl_data==0] <- NA
# Replace the zeros with "NA"

df <- data.frame(t.pregnant,expl_data,class)
# Create the transformed dataframe
head(df,5)

##   t.pregnant plasma bl.press tr.thick serum.ins  bmi  diab age class
## 1          6    148       72       35        NA 33.6 0.627  50     1
## 2          1     85       66       29        NA 26.6 0.351  31     0
## 3          8    183       64       NA        NA 23.3 0.672  32     1
## 4          1     89       66       23        94 28.1 0.167  21     0
## 5          0    137       40       35       168 43.1 2.288  33     1

# View the first 5 obs of the df
tail(df,5)

##     t.pregnant plasma bl.press tr.thick serum.ins  bmi  diab age
class
## 764         10    101       76       48       180 32.9 0.171  63
0
```

```
## 765           2    122       70       27        NA 36.8 0.340  27
0
## 766           5    121       72       23       112 26.2 0.245  30
0
## 767           1    126       60       NA        NA 30.1 0.349  47
1
## 768           1     93       70       31        NA 30.4 0.315  23
0
```

```
# View the last 5 obs of the df
```

```
sum(is.na(expl_data$tr.thick))
```

```
## [1] 227
```

```
# Second way to check the percentage of the missing values
```

```
227/768
```

```
## [1] 0.2955729
```

```
# 29.55% of the values of the variable tr.thick are missing!
```

```
sum(is.na(expl_data$serum.ins))
```

```
## [1] 374
```

```
# Second way to check the percentage of the missing values
```

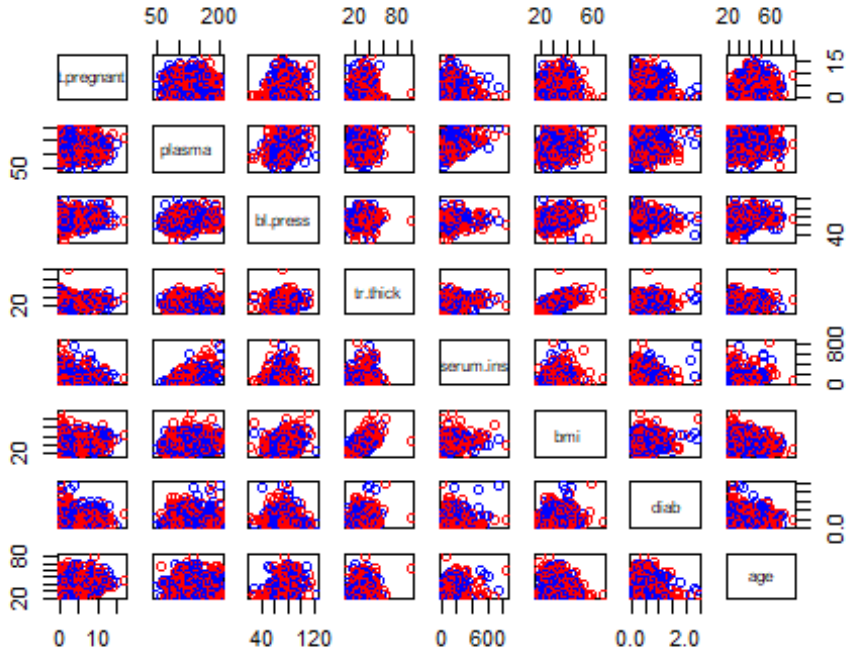
```
374/768
```

```
## [1] 0.4869792
```

```
# 48.69% of the values of the variable serum.ins are missing!

# hist(df)
# Histograms of all the variables, in order to check if there's
normality. We observe that there's no normality in each of the 8
variables, hence we will try some transformations. The reason lies in
the fact that if not each and every one of the variables is normally
distributed then all together, they are not going to be multivariate
normally distributed!
```

Pairwise colorful plots

```
cols <- character(nrow(df))
cols[] <- "black"
cols[df$class %in% c(0,1)] <- c("blue","red")
pairs(df[,-9],col=cols)
```

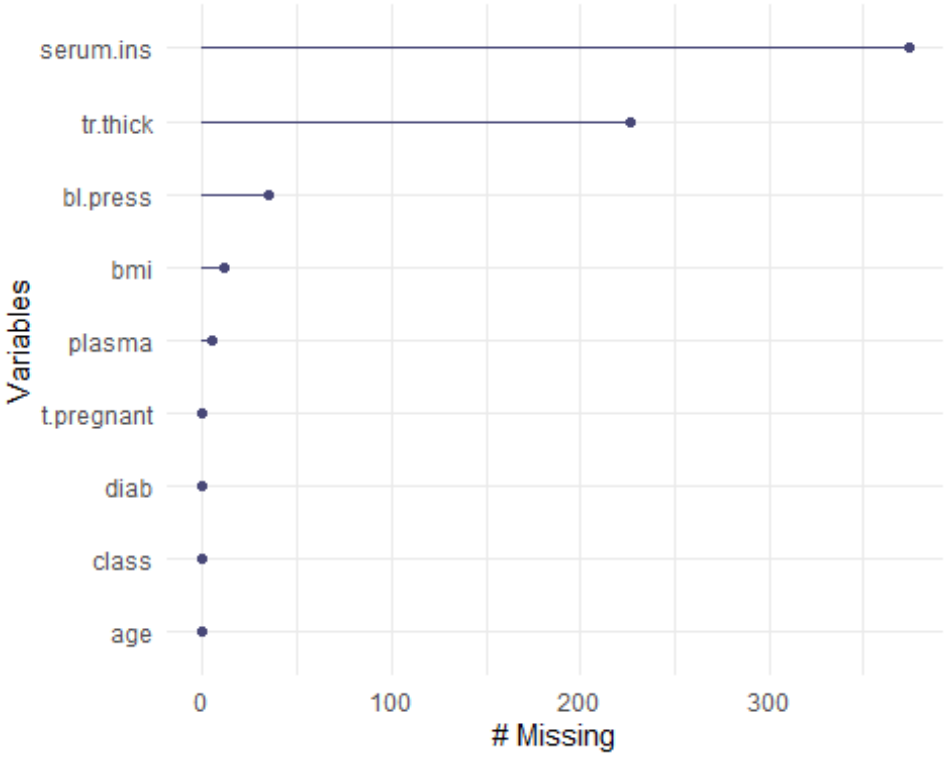



```
# We observe that there's no conspicuous discrimination available
```

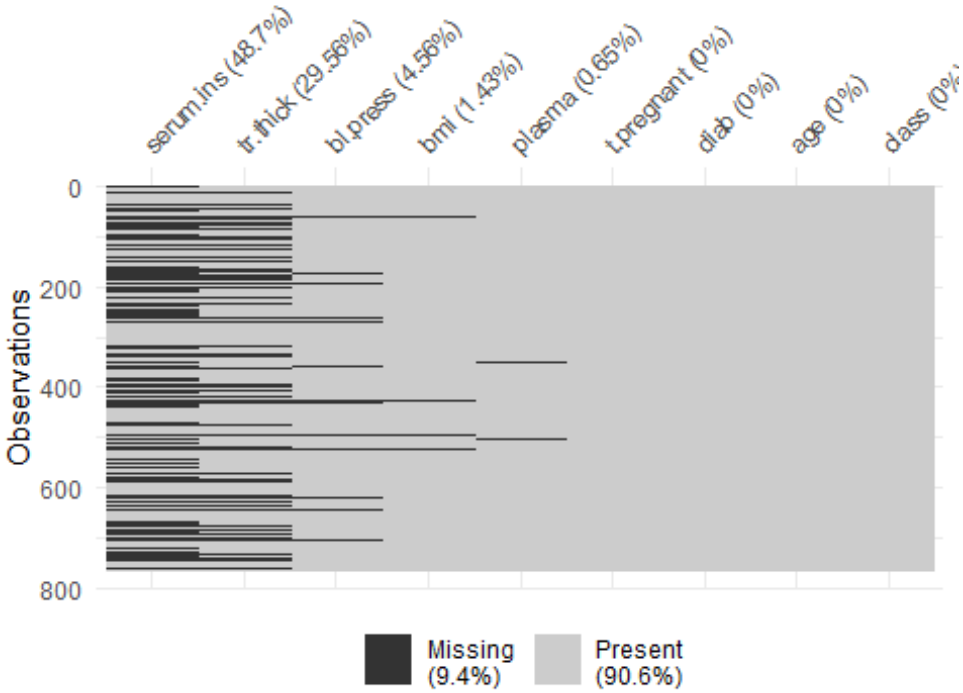
Data visualizations using package naniar

```
library(naniar)
gg_miss_var(df)
```

```
## Warning: It is deprecated to specify `guide = FALSE` to remove a
guide. Please
## use `guide = "none"` instead.
```



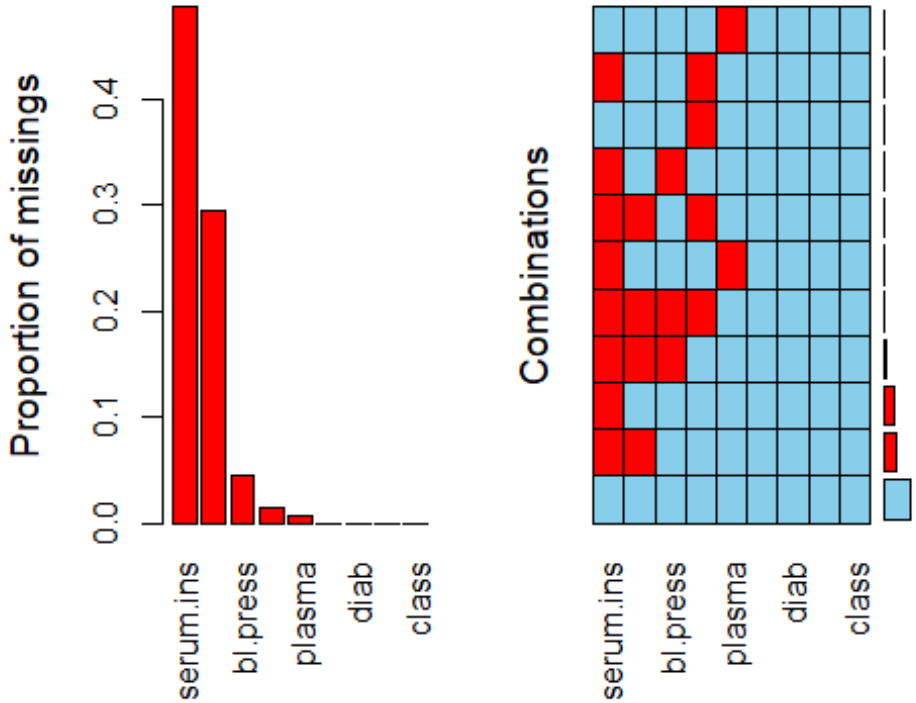

```
# The other package that can be used is the package naniar developped
by Nick Tierney's and which is based on ggplot. Naniar provides
# principled, tidy ways to summarise, visualise, and manipulate missing
data with minimal deviations from the workflows in ggplot2 and
# tidy data.

vis_miss(df, sort_miss = TRUE)
```



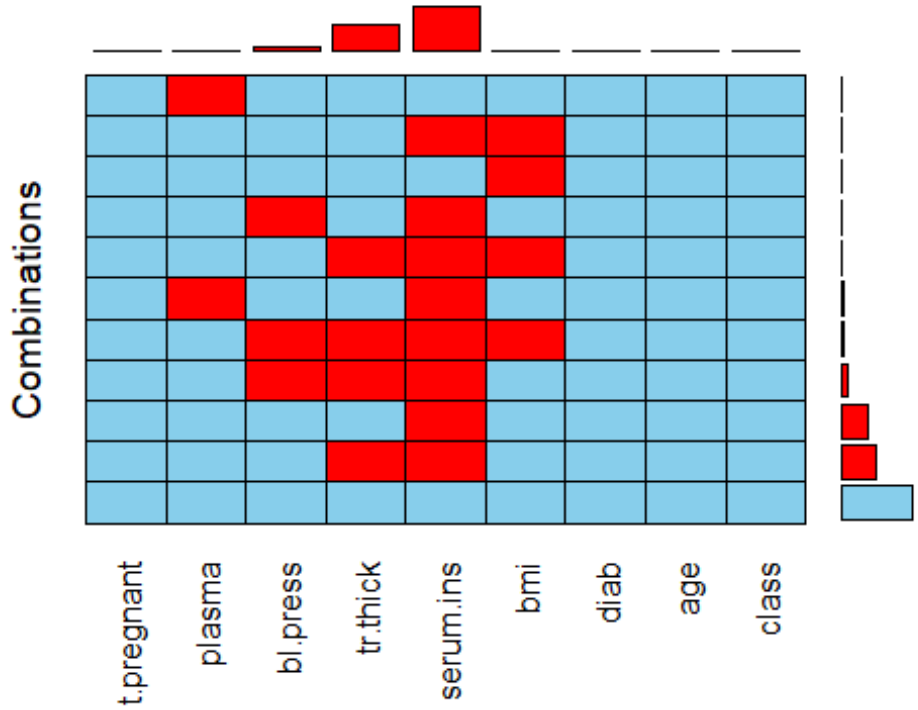

```
# As VIM package has matrix plot, similarly naniar has the var_miss()
function. It provides a summary of whether the data is missing
# (in black) or not. It also provides the percentage of missing values
in each column.
```

Data visualizations using package VIM

```
library(VIM)

## Loading required package: colorspace

## Loading required package: grid

## VIM is ready to use.

## Suggestions and bug-reports can be submitted at:
https://github.com/statistikat/VIM/issues

##
## Attaching package: 'VIM'

## The following object is masked from 'package:datasets':
##
##     sleep

res1 <- summary(aggr(df,sortVar=TRUE))$combinations
```




```
##
##  Variables sorted by number of missings:
##    Variable       Count
##   serum.ins 0.486979167
##    tr.thick 0.295572917
##    bl.press 0.045572917
##         bmi 0.014322917
##      plasma 0.006510417
##  t.pregnant 0.000000000
##        diab 0.000000000
##         age 0.000000000
##       class 0.000000000

# The function VIM aggr calculates and represents the number of missing
entries in each variable and for certain combinations of
# variables (which tend to be missing simultaneously).

res2 <- summary(aggr(df,prop=TRUE,combined=TRUE))$combinations
```




```
# The graph represents the pattern, with blue for observed and red for
missing.
```

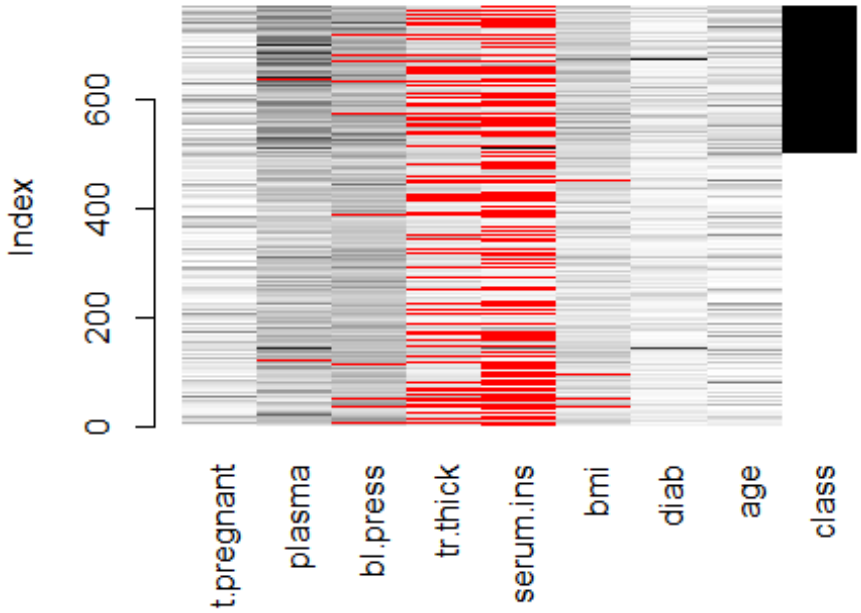
Most frequent combination of the variables

```
head(res1[rev(order(res1[,2])),])
```

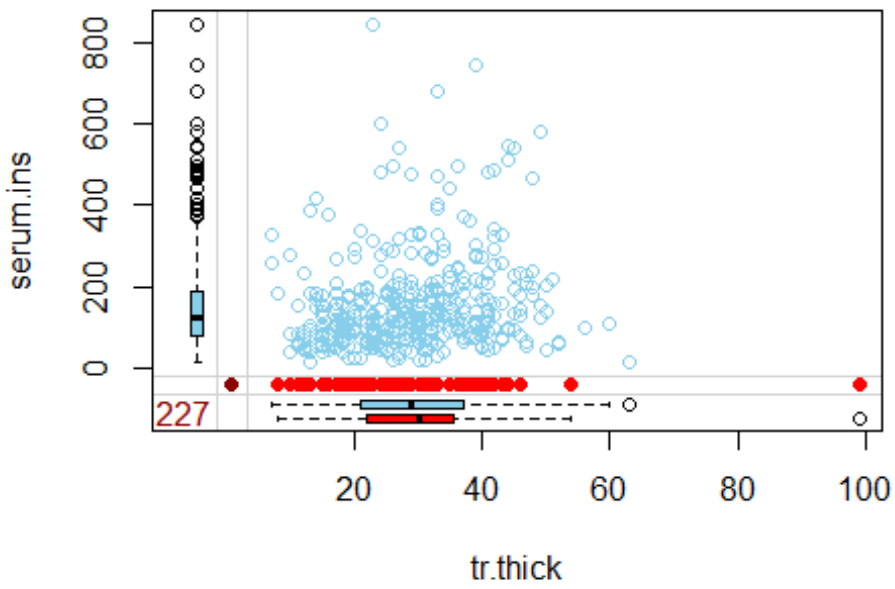
```
##         Combinations Count    Percent
## 1  0:0:0:0:0:0:0:0:0   392 51.0416667
## 5  0:0:0:1:1:0:0:0:0   192 25.0000000
## 3  0:0:0:0:1:0:0:0:0   140 18.2291667
## 8  0:0:1:1:1:0:0:0:0    26  3.3854167
## 9  0:0:1:1:1:1:0:0:0     7  0.9114583
## 11 0:1:0:0:1:0:0:0:0     4  0.5208333

# We can see that the combination which is the most frequent is the one
where all the variables are observed (392 values).
# Then, the second one is the one where variable 4 (which corresponds
to tr.thick) and variable 5(which corresponds to serum.ins) are
# simultaneously missing (192 rows). 1 stands for the variables missing
while 0 stands for the observed ones!
```

More data visualizations using package VIM

```
matrixplot(df, sortby = 9)
```



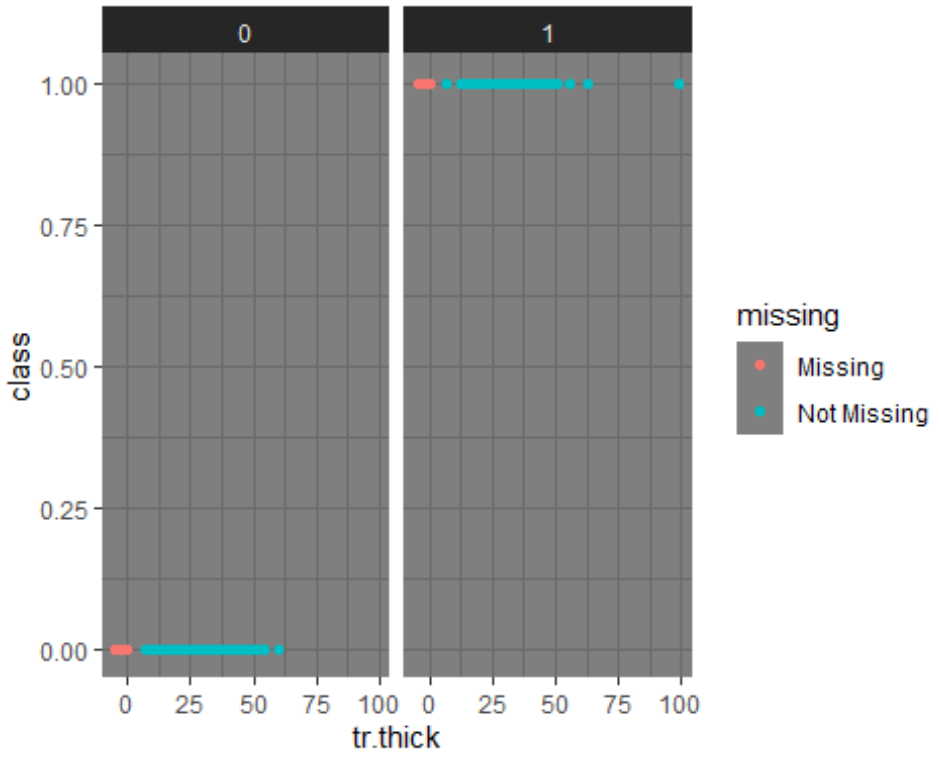

```r
# The VIM function matrixplot creates a matrix plot in which all cells of a data matrix are visualized by rectangles. Available data
# is coded according to a continuous color scheme (gray scale), while missing/imputed data is visualized by a clearly distinguishable
# color (red). If you use Rstudio the plot is not interactive (there are the warnings), but if you use R directly, you can click on a
# column of your choice: the rows are sorted (decreasing order) of the values of this column. This is useful to check if there is an
# association between the value of a variable and the missingness of another one.
# Here, the variable selected to sort by, is variable 9 (which refers to class!)


marginplot(df[,c("tr.thick","serum.ins")])
```



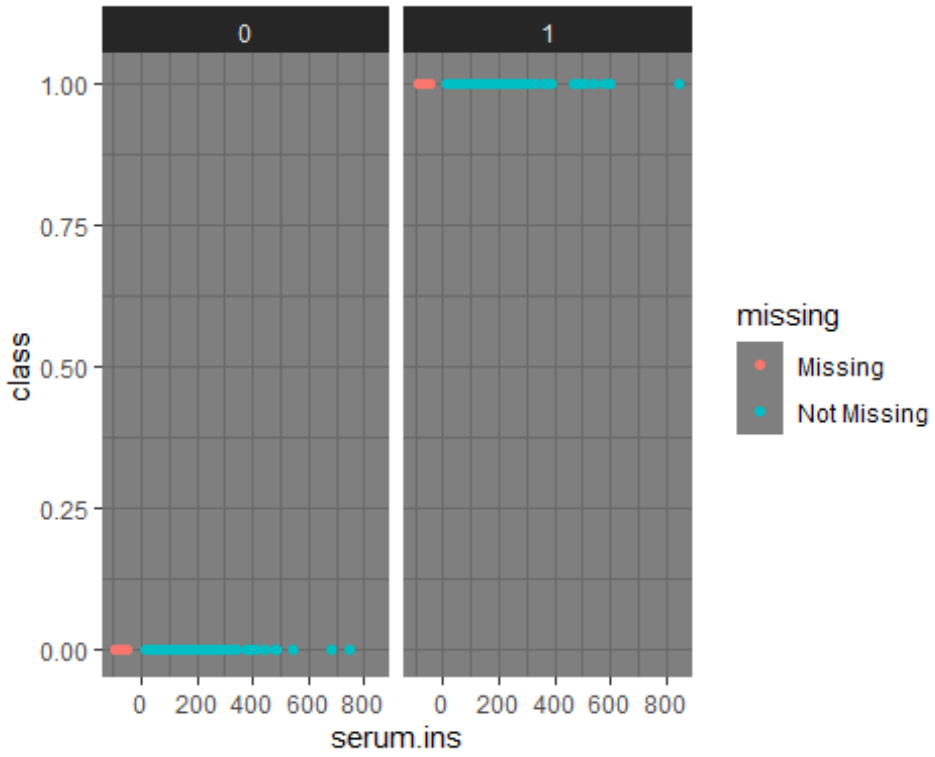

```
# Boxplots for available and missing/imputed data, as well as
univariate scatterplots for missing/imputed values in one variable are
# shown in the plot margins. Imputed values in either of the variables
are highlighted in the scatterplot.
# Furthermore, the frequencies of the missing/imputed values can be
displayed by a number (lower left of the plot). The number in the
# lower left corner is the number of observations that are
missing/imputed in both variables.
# We can see that the distribution of tr.thick is the same when
serum.ins is observed and when serum.ins is missing. If the two
boxplots
# (red and blue) would have been very different it would imply that
when serum.ins is missing the values of tr.thick can be very high or
# very low which lead to suspect the MAR hypothesis.
```

Further data visualizations using ggplot2 package

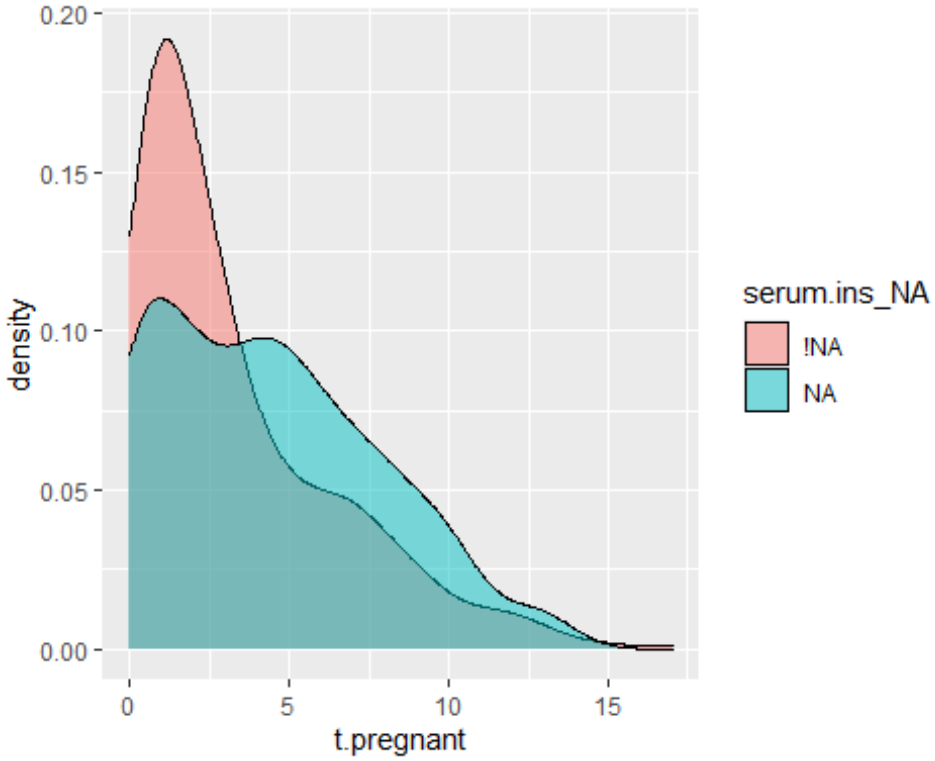
```
library(ggplot2)
ggplot(df,
       aes(x = tr.thick,
           y = class)) +
  geom_miss_point() +
  facet_wrap(~data$class)+
  theme_dark()
```




```
# The function geom_miss_point() is close to the margin plot function
of VIM but within the ggplot framework.


ggplot(df,
       aes(x = serum.ins,
           y = class)) +
  geom_miss_point() +
  facet_wrap(~data$class)+
  theme_dark()
```

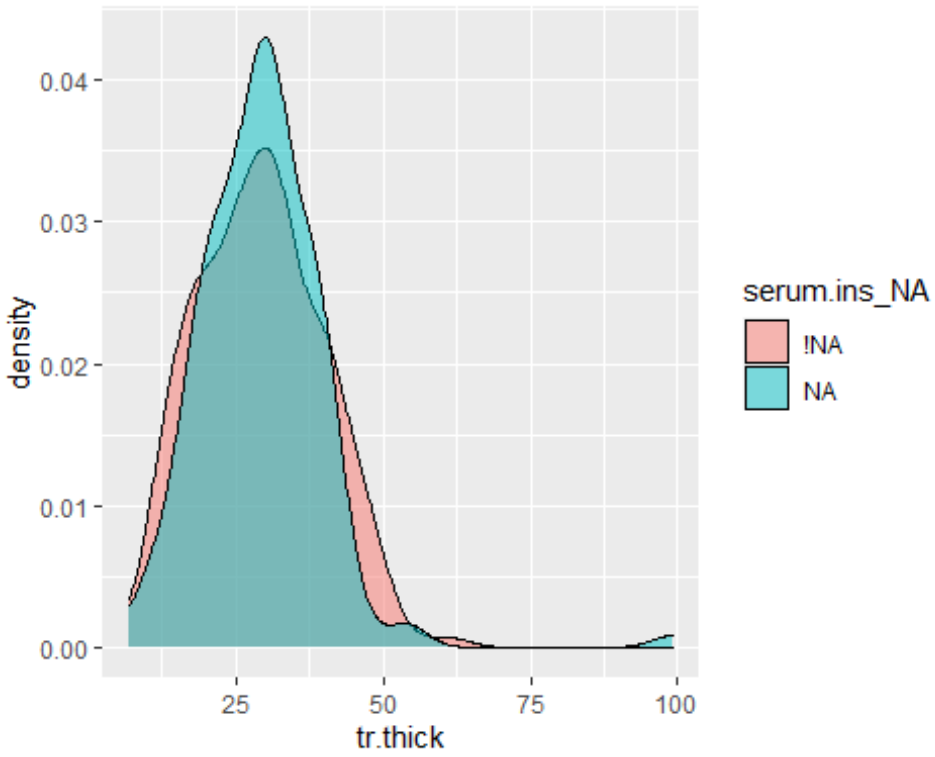



```
ggplot( bind_shadow(df),
        aes(x = t.pregnant,
            fill = serum.ins_NA)) +
  geom_density(alpha=0.5)
```

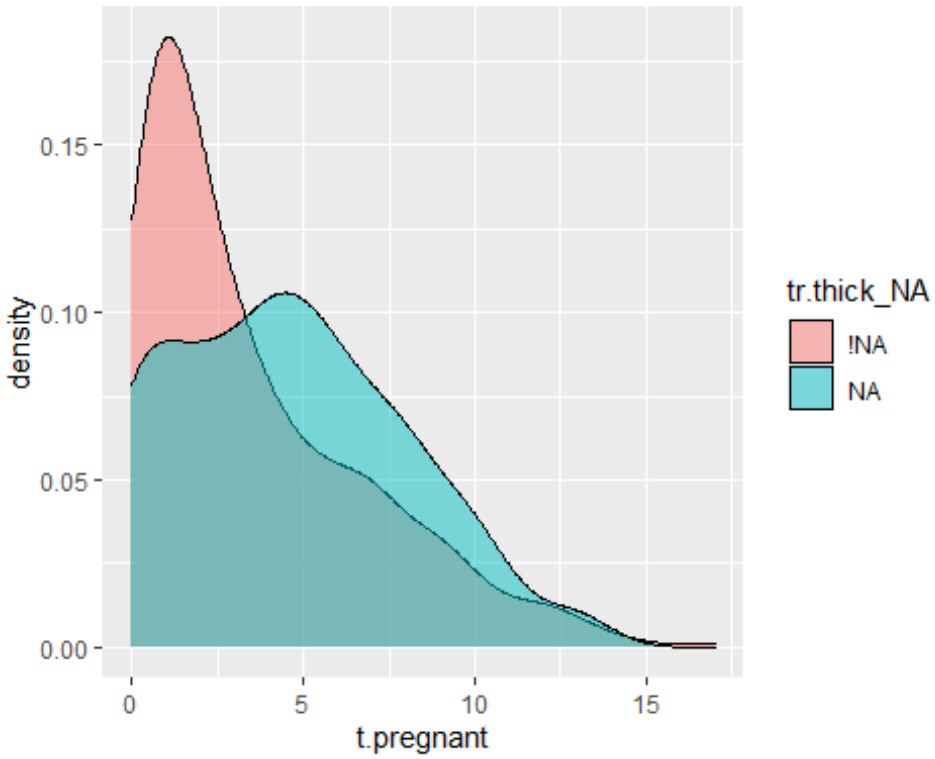



```
# We can plot the distribution of t.pregnant, plotting for values of
t.pregnant when serum.ins is missing, and when it is not missing.


ggplot( bind_shadow(df),
        aes(x = tr.thick,
            fill = serum.ins_NA)) +
  geom_density(alpha=0.5)
```

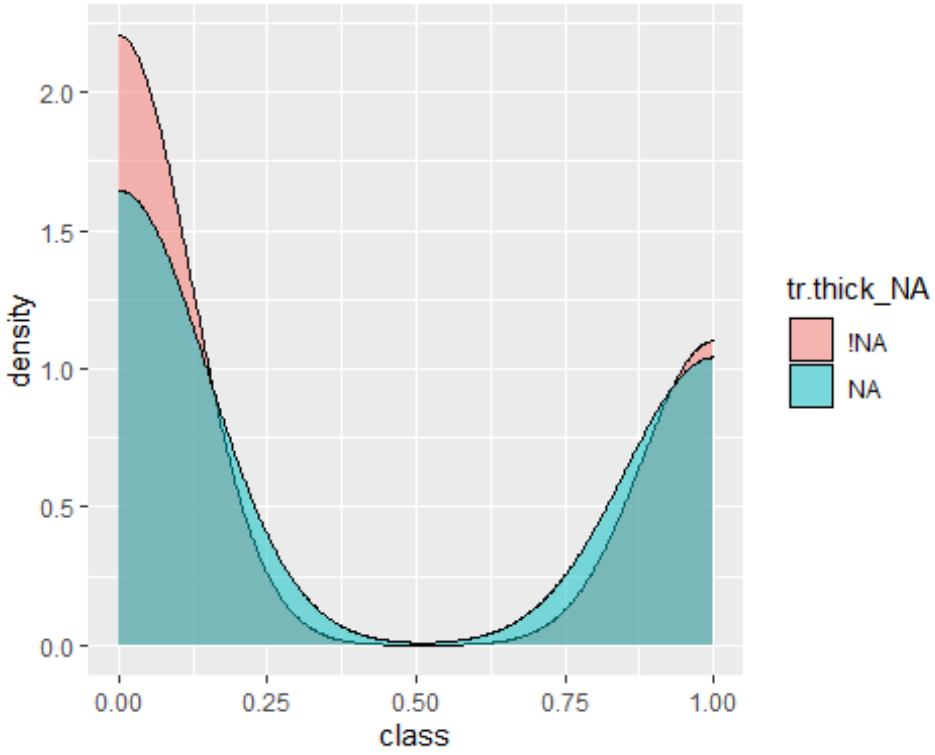
```
## Warning: Removed 227 rows containing non-finite values
(stat_density).
```



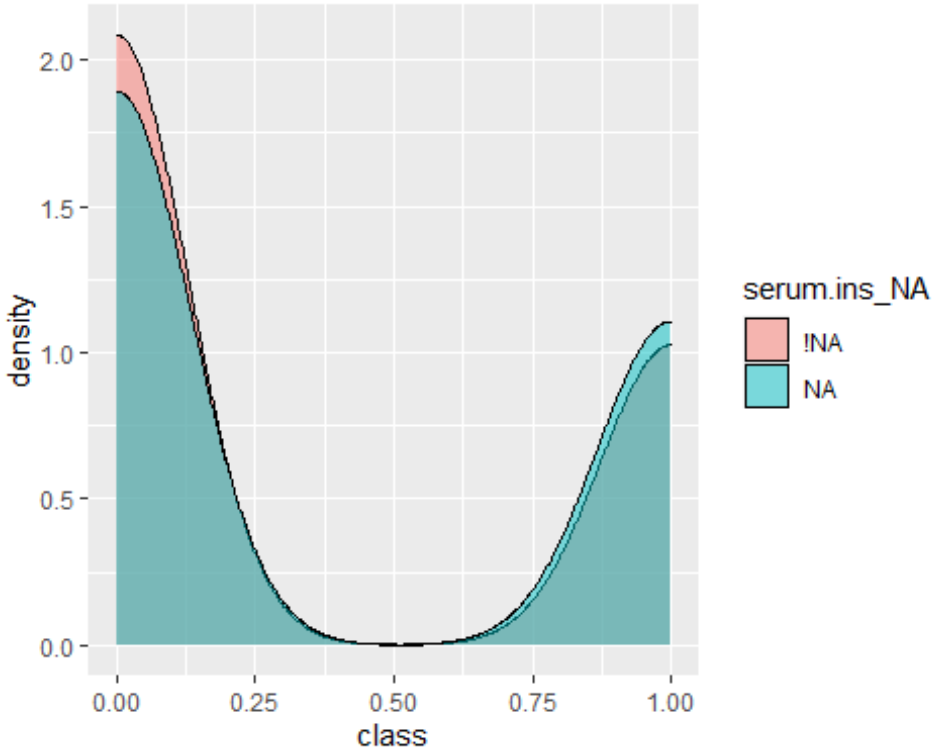

```
# We can plot the distribution of tr.thick, plotting for values of
tr.thick when serum.ins is missing, and when it is not missing.

ggplot( bind_shadow(df),
        aes(x = t.pregnant,
            fill = tr.thick_NA)) +
  geom_density(alpha=0.5)
```




```
# We can plot the distribution of t.pregnant, plotting for values of
t.pregnant when tr.thick is missing, and when it is not missing.

ggplot( bind_shadow(df),
        aes(x = class,
            fill = tr.thick_NA)) +
  geom_density(alpha=0.5)
```




```
# We can plot the distribution of class, plotting for values of class
when tr.thick is missing, and when it is not missing.

ggplot( bind_shadow(df),
        aes(x = class,
            fill = serum.ins_NA)) +
  geom_density(alpha=0.5)
```




```
# We can plot the distribution of class, plotting for values of class
when serum.ins is missing, and when it is not missing.
```

Insights regarding the missing values

```
pct_miss(df)
```

```
## [1] 9.43287
```

```
# Percentage of the total missing values in the data set
```

```
n_miss(df)
```

```
## [1] 652
```

```
# Number of the total missing values in the data set
```

```
miss_var_summary(df, order = T)
```

```
## # A tibble: 9 x 3
##   variable   n_miss pct_miss
##   <chr>       <int>    <dbl>
## 1 serum.ins     374   48.7
## 2 tr.thick      227   29.6
## 3 bl.press       35    4.56
## 4 bmi            11    1.43
## 5 plasma          5    0.651
## 6 t.pregnant      0    0
```

```
## 7 diab            0    0
## 8 age             0    0
## 9 class           0    0

# Provide a summary for each variable of the number, percent missings,
and cumulative sum of missings of the order of the variables.
# By default, it orders by the most missings in each variable.

as_shadow(df)

## # A tibble: 768 x 9
##    t.pregnant_NA plasma_NA bl.press_NA tr.thick_NA serum.ins_NA
bmi_NA diab_NA
##    <fct>         <fct>     <fct>       <fct>       <fct>
<fct>  <fct>
##  1 !NA           !NA       !NA         !NA         NA           !NA
!NA
##  2 !NA           !NA       !NA         !NA         NA           !NA
!NA
##  3 !NA           !NA       !NA         NA          NA           !NA
!NA
##  4 !NA           !NA       !NA         !NA         !NA          !NA
!NA
##  5 !NA           !NA       !NA         !NA         !NA          !NA
!NA
##  6 !NA           !NA       !NA         NA          NA           !NA
!NA
##  7 !NA           !NA       !NA         !NA         !NA          !NA
!NA
##  8 !NA           !NA       NA          NA          NA           !NA
!NA
##  9 !NA           !NA       !NA         !NA         !NA          !NA
!NA
## 10 !NA           !NA       !NA         NA          NA           NA
!NA
## # ... with 758 more rows, and 2 more variables: age_NA <fct>,
class_NA <fct>

# A matrix with missing and non missing values

bind_shadow(df)

## # A tibble: 768 x 18
##    t.pregnant plasma bl.press tr.thick serum.ins   bmi  diab   age
class
##         <int>  <int>    <int>    <int>     <int> <dbl> <dbl> <int>
<int>
##  1          6    148       72       35        NA  33.6 0.627    50
1
##  2          1     85       66       29        NA  26.6 0.351    31
0
```

```
##  3           8    183       64       NA        NA  23.3 0.672    32
1
##  4           1     89       66       23        94  28.1 0.167    21
0
##  5           0    137       40       35       168  43.1 2.29     33
1
##  6           5    116       74       NA        NA  25.6 0.201    30
0
##  7           3     78       50       32        88  31   0.248    26
1
##  8          10    115       NA       NA        NA  35.3 0.134    29
0
##  9           2    197       70       45       543  30.5 0.158    53
1
## 10           8    125       96       NA        NA  NA   0.232    54
1
## # ... with 758 more rows, and 9 more variables: t.pregnant_NA <fct>,
## #   plasma_NA <fct>, bl.press_NA <fct>, tr.thick_NA <fct>,
serum.ins_NA <fct>,
## #   bmi_NA <fct>, diab_NA <fct>, age_NA <fct>, class_NA <fct>

# The initial matrix concatenated with the matrix with missing and non
missing values
```

Pairwise correlations check

```
library(Hmisc)

## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, units

mat <- as.matrix(df)
newmat <- mat[,1:8]
rcorr(newmat)

##            t.pregnant plasma bl.press tr.thick serum.ins  bmi  diab
age
## t.pregnant       1.00   0.13     0.21     0.10      0.08 0.02 -0.03
0.54
## plasma           0.13   1.00     0.22     0.23      0.58 0.23  0.14
0.27
## bl.press         0.21   0.22     1.00     0.23      0.10 0.29  0.00
```

```
0.33
## tr.thick         0.10   0.23     0.23     1.00      0.18 0.65  0.12
0.17
## serum.ins        0.08   0.58     0.10     0.18      1.00 0.23  0.13
0.22
## bmi              0.02   0.23     0.29     0.65      0.23 1.00  0.16
0.03
## diab            -0.03   0.14     0.00     0.12      0.13 0.16  1.00
0.03
## age              0.54   0.27     0.33     0.17      0.22 0.03  0.03
1.00
##
## n
##             t.pregnant plasma bl.press tr.thick serum.ins bmi diab
age
## t.pregnant        768    763      733      541       394 757  768
768
## plasma            763    763      728      536       393 752  763
763
## bl.press          733    728      733      539       394 729  733
733
## tr.thick          541    536      539      541       394 539  541
541
## serum.ins         394    393      394      394       394 393  394
394
## bmi               757    752      729      539       393 757  757
757
## diab              768    763      733      541       394 757  768
768
## age               768    763      733      541       394 757  768
768
##
## P
##             t.pregnant plasma bl.press tr.thick serum.ins bmi    diab
age
## t.pregnant            0.0004 0.0000   0.0197   0.1034    0.5507
0.3535 0.0000
## plasma     0.0004            0.0000   0.0000   0.0000    0.0000
0.0001 0.0000
## bl.press   0.0000     0.0000          0.0000   0.0513    0.0000
0.9396 0.0000
## tr.thick   0.0197     0.0000 0.0000           0.0002    0.0000
0.0074 0.0000
## serum.ins  0.1034     0.0000 0.0513   0.0002            0.0000
0.0096 0.0000
## bmi        0.5507     0.0000 0.0000   0.0000   0.0000
0.0000 0.4777
## diab       0.3535     0.0001 0.9396   0.0074   0.0096    0.0000
0.3530
```

```
## age         0.0000     0.0000 0.0000   0.0000   0.0000    0.4777
0.3530

# rcorr returns a list with elements r, the matrix of correlations, n
the matrix of number of observations used in analyzing
# each pair of variables, and P, the asymptotic P-values. Pairs with
fewer than 2 non-missing values have the r values set to NA.
# The diagonals of n are the number of non-NAs for the single variable
corresponding to that row and column. P-values are approximated by
# using the t or F distributions.
```

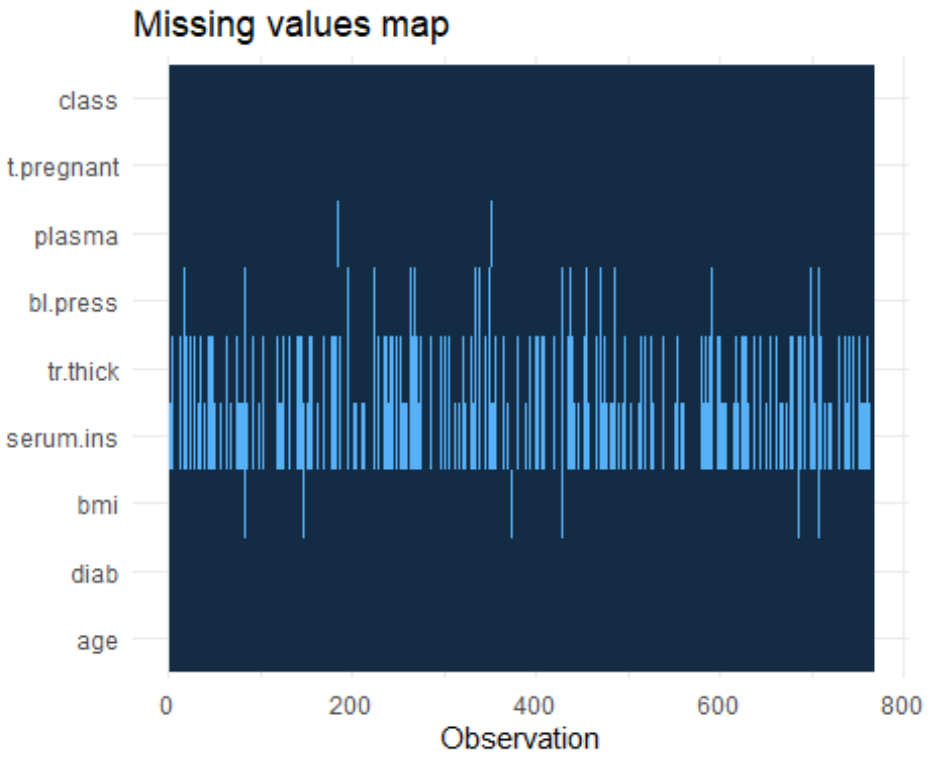
Insights about the dataframe named df using the finalfit package

```
library(finalfit)
expl <-
c("t.pregnant","plasma","bl.press","tr.thick","serum.ins","bmi","diab",
"age")
dep <- c("class")

ff_glimpse(df,dependent=dep,explanatory=expl,digits = 1)

## $Continuous
##                 label var_type   n missing_n missing_percent  mean
sd  min
## class           class    <int> 768         0             0.0   0.3
0.5  0.0
## t.pregnant t.pregnant    <int> 768         0             0.0   3.8
3.4  0.0
## plasma         plasma    <int> 763         5             0.7 121.7
30.5 44.0
## bl.press     bl.press    <int> 733        35             4.6  72.4
12.4 24.0
## tr.thick     tr.thick    <int> 541       227            29.6  29.2
10.5  7.0
## serum.ins   serum.ins    <int> 394       374            48.7 155.5
118.8 14.0
## bmi               bmi    <dbl> 757        11             1.4  32.5
6.9 18.2
## diab             diab    <dbl> 768         0             0.0   0.5
0.3  0.1
## age               age    <int> 768         0             0.0  33.2
11.8 21.0
##            quartile_25 median quartile_75   max
## class              0.0    0.0         1.0   1.0
## t.pregnant         1.0    3.0         6.0  17.0
## plasma            99.0  117.0       141.0 199.0
## bl.press          64.0   72.0        80.0 122.0
## tr.thick          22.0   29.0        36.0  99.0
## serum.ins         76.2  125.0       190.0 846.0
## bmi               27.5   32.3        36.6  67.1
## diab               0.2    0.4         0.6   2.4
```

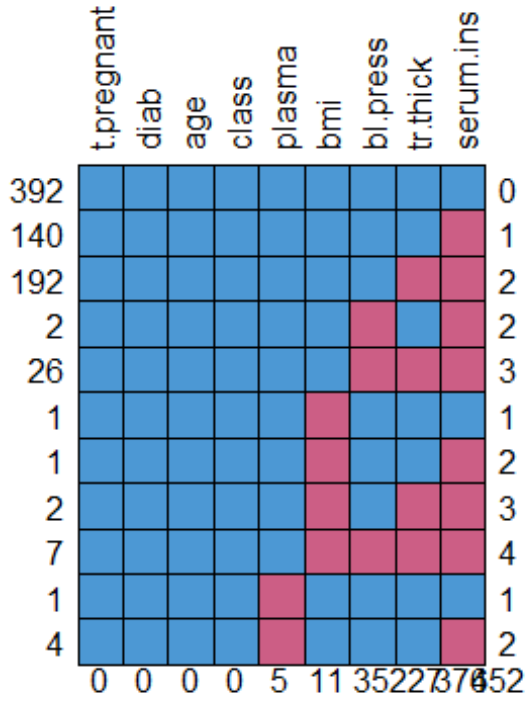
```
## age               24.0   29.0        41.0  81.0
##
## $Categorical
## data frame with 0 columns and 768 rows
```

Heatmap of the missing values in the dataframe named df

```
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:Hmisc':
##
##     src, summarize

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

df %>%
  missing_plot(dependent = dep,explanatory = expl)
```

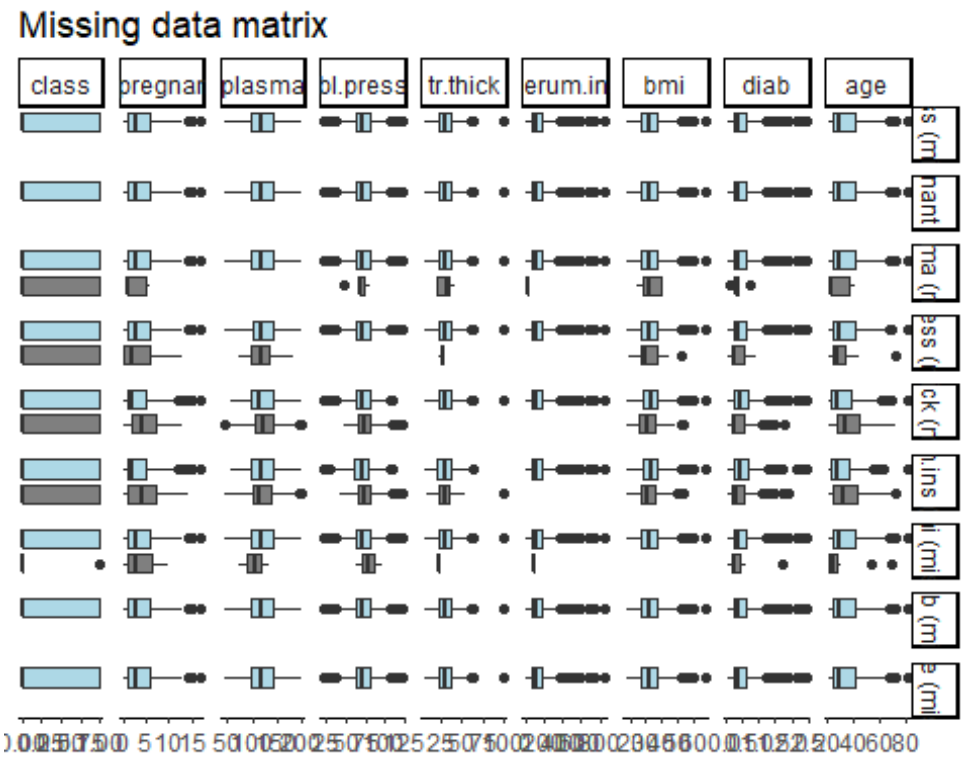


```
# Missing values occurence plot. Heat map of missing values in dataset.

df %>%
  missing_pattern(dependent=dep, explanatory=expl)
```




```
##     t.pregnant diab age class plasma bmi bl.press tr.thick serum.ins
## 392          1    1   1     1      1   1        1        1          1
0
## 140          1    1   1     1      1   1        1        1          0
1
## 192          1    1   1     1      1   1        1        0          0
2
## 2            1    1   1     1      1   1        0        1          0
2
## 26           1    1   1     1      1   1        0        0          0
3
## 1            1    1   1     1      1   0        1        1          1
1
## 1            1    1   1     1      1   0        1        1          0
2
## 2            1    1   1     1      1   0        1        0          0
3
## 7            1    1   1     1      1   0        0        0          0
4
## 1            1    1   1     1      0   1        1        1          1
1
## 4            1    1   1     1      0   1        1        1          0
```

```
2
##               0    0   0     0       5  11       35       227       374
652

# Characterise missing data for finalfit models

df %>%
  missing_pairs(dependent=dep, explanatory=expl)

## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```




```
# Missing values pairs plot. Compare the occurence of missing values in
all variables by each other. Suggest limiting the number
# of variables to a maximum of around six
```

Summary of the missing values in the dataframe named df

```
df %>%
  missing_glimpse(dependent=dep, explanatory=expl)

##                  label var_type   n missing_n missing_percent
## t.pregnant t.pregnant    <int> 768         0             0.0
## plasma         plasma    <int> 763         5             0.7
## bl.press     bl.press    <int> 733        35             4.6
## tr.thick     tr.thick    <int> 541       227            29.6
## serum.ins   serum.ins    <int> 394       374            48.7
```

```
## bmi               bmi    <dbl> 757        11             1.4
## diab             diab    <dbl> 768         0             0.0
## age               age    <int> 768         0             0.0
## class           class    <int> 768         0             0.0
```

Creating two different datasets to work on

```
df1 <- df
# The first dataset that has the missing values df1, which will be
imputed
df2 <- na.omit(df)
# The second dataset, in which we omit the observations with the
missing values
str(df2)

## 'data.frame':    392 obs. of  9 variables:
##  $ t.pregnant: int  1 0 3 2 1 5 0 1 1 3 ...
##  $ plasma    : int  89 137 78 197 189 166 118 103 115 126 ...
##  $ bl.press  : int  66 40 50 70 60 72 84 30 70 88 ...
##  $ tr.thick  : int  23 35 32 45 23 19 47 38 30 41 ...
##  $ serum.ins : int  94 168 88 543 846 175 230 83 96 235 ...
##  $ bmi       : num  28.1 43.1 31 30.5 30.1 25.8 45.8 43.3 34.6 39.3
...
##  $ diab      : num  0.167 2.288 0.248 0.158 0.398 ...
##  $ age       : int  21 33 26 53 59 51 31 33 32 27 ...
##  $ class     : int  0 1 1 1 1 1 1 0 1 0 ...
##  - attr(*, "na.action")= 'omit' Named int [1:376] 1 2 3 6 8 10 11 12
13 16 ...
##   ..- attr(*, "names")= chr [1:376] "1" "2" "3" "6" ...

dim(df2)

## [1] 392   9

summary(df2)

##    t.pregnant         plasma         bl.press         tr.thick
##  Min.   : 0.000   Min.   : 56.0   Min.   : 24.00   Min.   : 7.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.:21.00
##  Median : 2.000   Median :119.0   Median : 70.00   Median :29.00
##  Mean   : 3.301   Mean   :122.6   Mean   : 70.66   Mean   :29.15
##  3rd Qu.: 5.000   3rd Qu.:143.0   3rd Qu.: 78.00   3rd Qu.:37.00
##  Max.   :17.000   Max.   :198.0   Max.   :110.00   Max.   :63.00
##    serum.ins           bmi             diab             age
##  Min.   : 14.00   Min.   :18.20   Min.   :0.0850   Min.   :21.00
##  1st Qu.: 76.75   1st Qu.:28.40   1st Qu.:0.2697   1st Qu.:23.00
##  Median :125.50   Median :33.20   Median :0.4495   Median :27.00
##  Mean   :156.06   Mean   :33.09   Mean   :0.5230   Mean   :30.86
##  3rd Qu.:190.00   3rd Qu.:37.10   3rd Qu.:0.6870   3rd Qu.:36.00
##  Max.   :846.00   Max.   :67.10   Max.   :2.4200   Max.   :81.00
##      class
```

```
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.3316
##  3rd Qu.:1.0000
##  Max.   :1.0000

which(is.na(df2))

## integer(0)

# As expected the result is zero!
```

Data imputation for variable triceps skinfold thickness(tr.thick)

```
# Data imputation with mean/median has the following advantages:
# Easy and fast.
# Works well with small numerical datasets.
# Cons:
# Doesn't factor the correlations between features. It only works on
the column level.
# Will give poor results on encoded categorical features (do NOT use it
on categorical features).
# Not very accurate.
# Doesn't account for the uncertainty in the imputations.
# Since there are a few outliers in the variable tr.thick we will
impute the missing values using the mean as follows:
df1$tr.thick[is.na(df1$tr.thick)] <- mean(df1$tr.thick,na.rm=T)
summary(df1$tr.thick)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    7.00   25.00   29.15   29.15   32.00   99.00

which(is.na(df1$tr.thick))

## integer(0)

# As expected the result is zero!
```

Data imputation for variable 2-hour serum insulin(serum.ins)

```
# Since there are loads of outliers in the variable serum.ins we will
impute the missing values using the median as follows:
df1$serum.ins[is.na(df1$serum.ins)] <- median(df1$serum.ins,na.rm=T)
summary(df1$serum.ins)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    14.0   121.5   125.0   140.7   127.2   846.0

which(is.na(df1$serum.ins))

## integer(0)
```

```
# As expected the result is zero!
```

Data imputation for variable blood pressure (bl.press)

```
# Since there are loads of outliers in the variable bl.pressure we will
impute the missing values using the median as follows:
df1$bl.press[is.na(df1$bl.press)] <- median(df1$bl.press,na.rm=T)
summary(df1$bl.press)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   24.00   64.00   72.00   72.39   80.00  122.00

which(is.na(df1$bl.press))

## integer(0)

# As expected the result is zero!
```

Data imputation for variable body mass index (bmi)

```
# Since there are loads of outliers in the variable bl.pressure we will
impute the missing values using the median as follows:
df1$bmi[is.na(df1$bmi)] <- median(df1$bmi,na.rm=T)
summary(df1$bmi)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   18.20   27.50   32.30   32.46   36.60   67.10

which(is.na(df1$bmi))

## integer(0)

# As expected the result is zero!
```

Data imputation for variable Plasma glucose concentration a 2 hours in an oral glucose tolerance test (plasma)

```
# Since there aren't outliers in the variable plasma we will impute the
missing values using the mean as follows:
df1$plasma[is.na(df1$plasma)] <- mean(df1$plasma,na.rm=T)
summary(df1$plasma)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   44.00   99.75  117.00  121.69  140.25  199.00

which(is.na(df1$plasma))

## integer(0)

# As expected the result is zero!
```

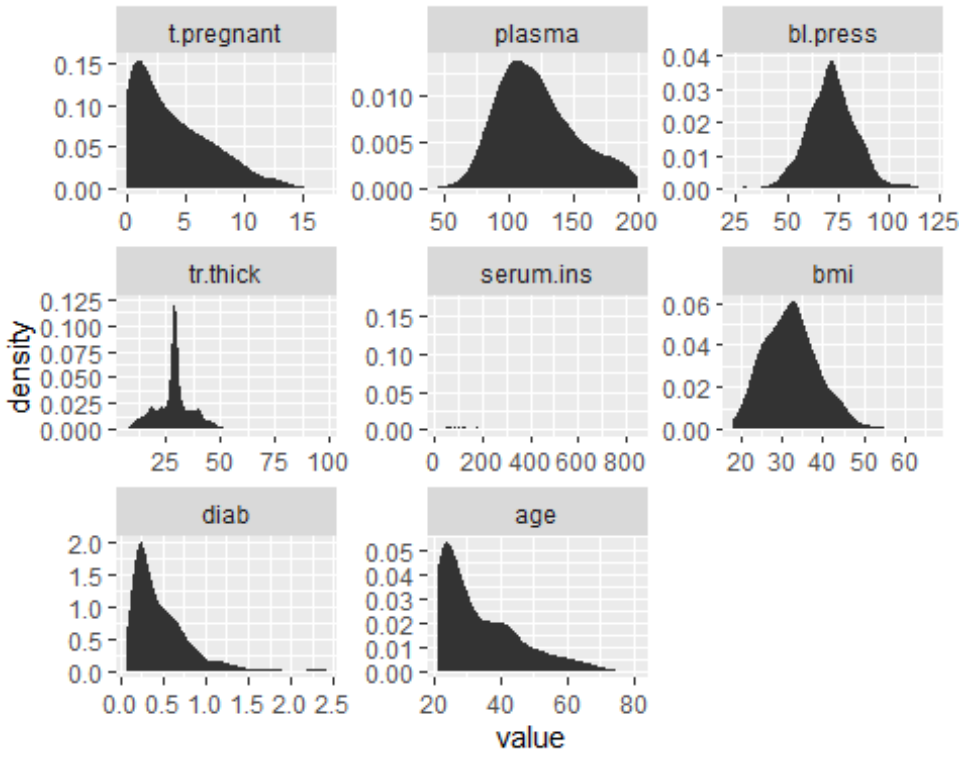
Check that there are no longer missing values

```
ff_glimpse(df1,dependent=dep,explanatory=expl,digits = 1)
```

```
## $Continuous
##                 label var_type   n missing_n missing_percent  mean
sd  min
## class           class    <int> 768         0             0.0   0.3
0.5  0.0
## t.pregnant t.pregnant    <int> 768         0             0.0   3.8
3.4  0.0
## plasma         plasma    <dbl> 768         0             0.0 121.7
30.4 44.0
## bl.press     bl.press    <int> 768         0             0.0  72.4
12.1 24.0
## tr.thick     tr.thick    <dbl> 768         0             0.0  29.2
8.8  7.0
## serum.ins   serum.ins    <dbl> 768         0             0.0 140.7
86.4 14.0
## bmi               bmi    <dbl> 768         0             0.0  32.5
6.9 18.2
## diab             diab    <dbl> 768         0             0.0   0.5
0.3  0.1
## age               age    <int> 768         0             0.0  33.2
11.8 21.0
##             quartile_25 median quartile_75   max
## class               0.0    0.0         1.0   1.0
## t.pregnant          1.0    3.0         6.0  17.0
## plasma             99.8  117.0       140.2 199.0
## bl.press           64.0   72.0        80.0 122.0
## tr.thick           25.0   29.2        32.0  99.0
## serum.ins         121.5  125.0       127.2 846.0
## bmi                27.5   32.3        36.6  67.1
## diab                0.2    0.4         0.6   2.4
## age                24.0   29.0        41.0  81.0
##
## $Categorical
## data frame with 0 columns and 768 rows

# Indeed there aren't any missing values now!
```

Better picture of the variable's distributions via ggplot (using the reshape2 package as well)

```
library(reshape2)
newdf <- melt(df1[,-9])

## No id variables; using all as measure variables

ggplot(data = newdf, aes(x = value)) +
  stat_density() +
  facet_wrap(~variable, scales = "free")
```




```
# Small multiple chart
```

Linear regression for the imputed dataset (df1)

```
n.class1 <- df1$class
m <- lm(n.class1~.,data=df1[,1:8])
summary(m)
```

```
##
## Call:
## lm(formula = n.class1 ~ ., data = df1[, 1:8])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.05571 -0.28879 -0.07463  0.29169  0.98399
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.0245791  0.1040300  -9.849  < 2e-16 ***
## t.pregnant   0.0206324  0.0050620   4.076 5.07e-05 ***
## plasma       0.0065445  0.0005465  11.975  < 2e-16 ***
## bl.press    -0.0012086  0.0013142  -0.920   0.3580
## tr.thick     0.0002124  0.0019534   0.109   0.9134
## serum.ins   -0.0001557  0.0001837  -0.847   0.3971
## bmi          0.0144776  0.0026001   5.568 3.58e-08 ***
## diab         0.1307640  0.0440699   2.967   0.0031 **
## age          0.0020888  0.0015418   1.355   0.1759
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.395 on 759 degrees of freedom
## Multiple R-squared:  0.3214, Adjusted R-squared:  0.3143
## F-statistic: 44.94 on 8 and 759 DF,  p-value: < 2.2e-16

# We observe that there are 4 variables (t.pregnant,plasma,bmi,diab)
statistically significant at a=5% significance level
```

Predict the probabilities with respect to the response variable

```
m.probs=predict(m,type="response")
m.probs[1:10]

##           1           2           3           4           5
6
##  0.64164242 -0.04497041  0.73955869 -0.03847287  0.79706436
0.19462294
##           7           8           9          10
##  0.01600629  0.42324083  0.71931753  0.44001793

# View the first 10 probabilities
m.pred=rep(0,768)
```

Use the regression model (m) to assign the dataset's observations to one of the two classes

```
# If a predicted probability is less that 0.5 the class is predicted as
0, otherwise 1 (which corresponds to non-diabetes, diabetes
respectively)
m.pred[m.probs>.5] = 1
table(m.pred,n.class1)

##       n.class1
## m.pred   0   1
##      0 445 119
##      1  55 149

# The assessment is based on calculated misclassification errors or
rates. We can calculate them after the classification and arrange them
# in a confusion matrix just as above. Thus, we observe that the total
misclassification rate is (55+119)/768, which is equal to 22.6%
# The correct classification rate is it's complementary, 77.4%. The
false positive rate is  55/500, which is equal to 11%, while the false
negative rate is 119/268, which is equal to 44.4% . In our case, what
is of vital importance is the FNR (positive classified as negative) due
to the fact that we are interested in the women that in reality will
develop diabetes, while we predict that they won't!
```

Data preprocessing (for the significant variables only!)

```r
t.pregnant <- df1$t.pregnant
plasma <- df1$plasma
bmi <- df1$bmi
diab <- df1$diab
class <- df1$class

new_df <- data.frame(t.pregnant,plasma,bmi,diab,class)
# Creating a new dataframe for the significant predictors only plus
class, which is the depentent variable
head(new_df,5)
```

```
##   t.pregnant plasma  bmi  diab class
## 1          6    148 33.6 0.627     1
## 2          1     85 26.6 0.351     0
## 3          8    183 23.3 0.672     1
## 4          1     89 28.1 0.167     0
## 5          0    137 43.1 2.288     1
```

```r
tail(new_df,5)
```

```
##     t.pregnant plasma  bmi  diab class
## 764         10    101 32.9 0.171     0
## 765          2    122 36.8 0.340     0
## 766          5    121 26.2 0.245     0
## 767          1    126 30.1 0.349     1
## 768          1     93 30.4 0.315     0
```

```r
library(tseries)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
jarque.bera.test(df1$t.pregnant)
```

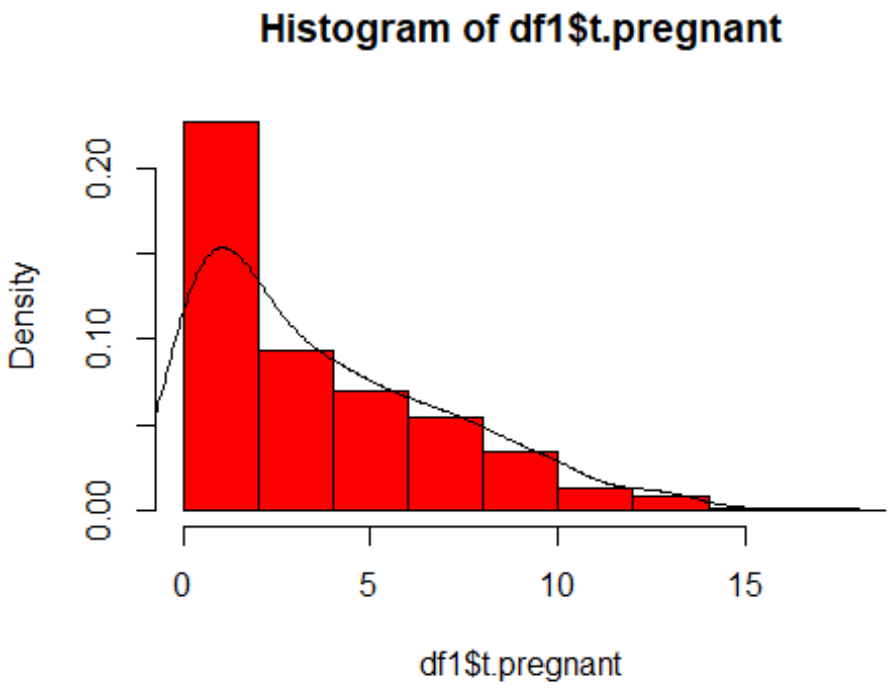
```
##
##  Jarque Bera Test
##
## data:  df1$t.pregnant
## X-squared = 104.38, df = 2, p-value < 2.2e-16
```

```r
# Reject Ho at 5% level of significance hence no normality
shapiro.test(df1$t.pregnant)
```

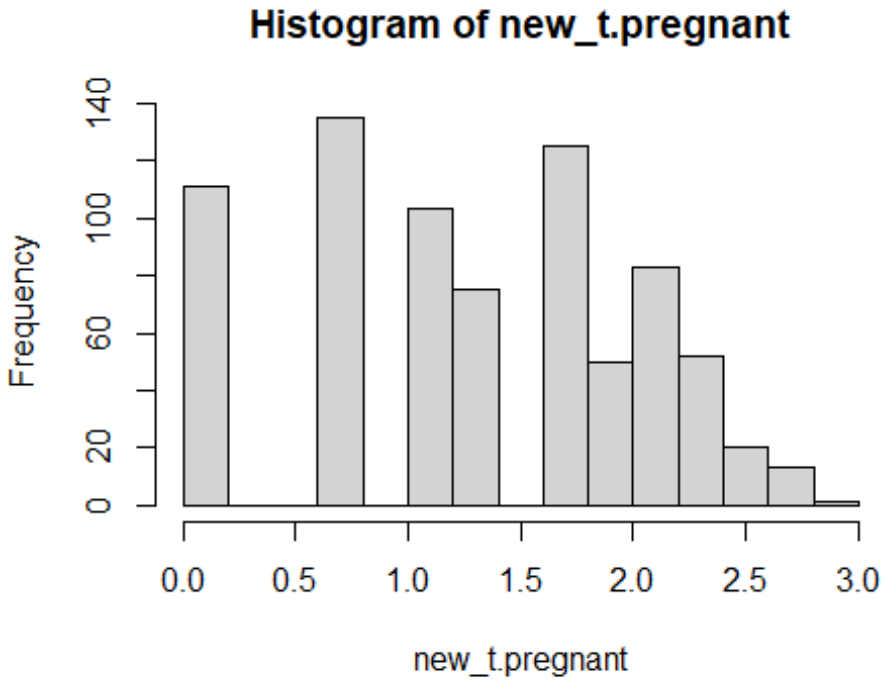
```
##
##  Shapiro-Wilk normality test
##
## data:  df1$t.pregnant
## W = 0.90428, p-value < 2.2e-16
```

```r
# Second way, same results
hist(df1$t.pregnant,col='red',freq=F)
```

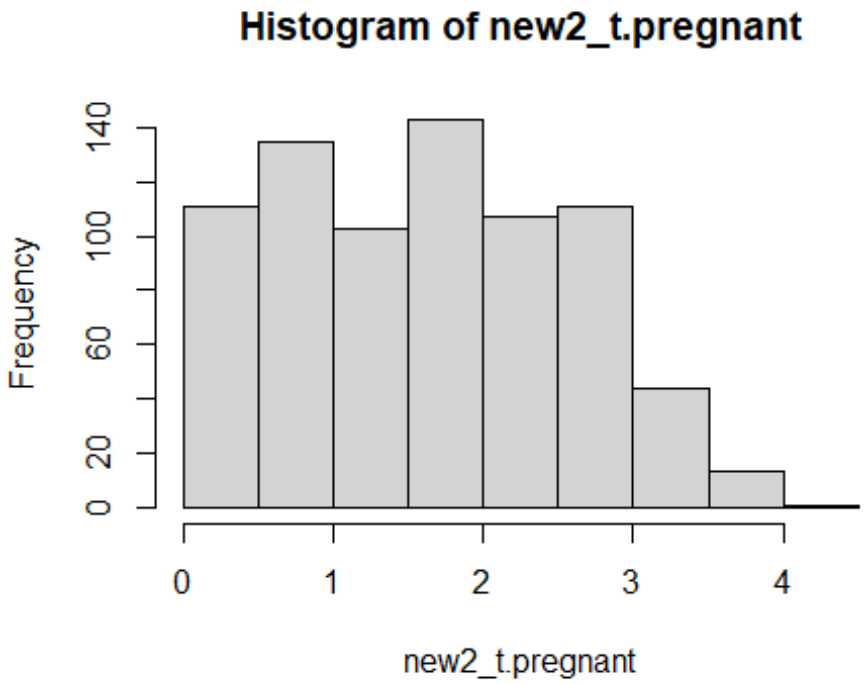
```r
# Third way to check it visually
lines(density(df1$t.pregnant))
```



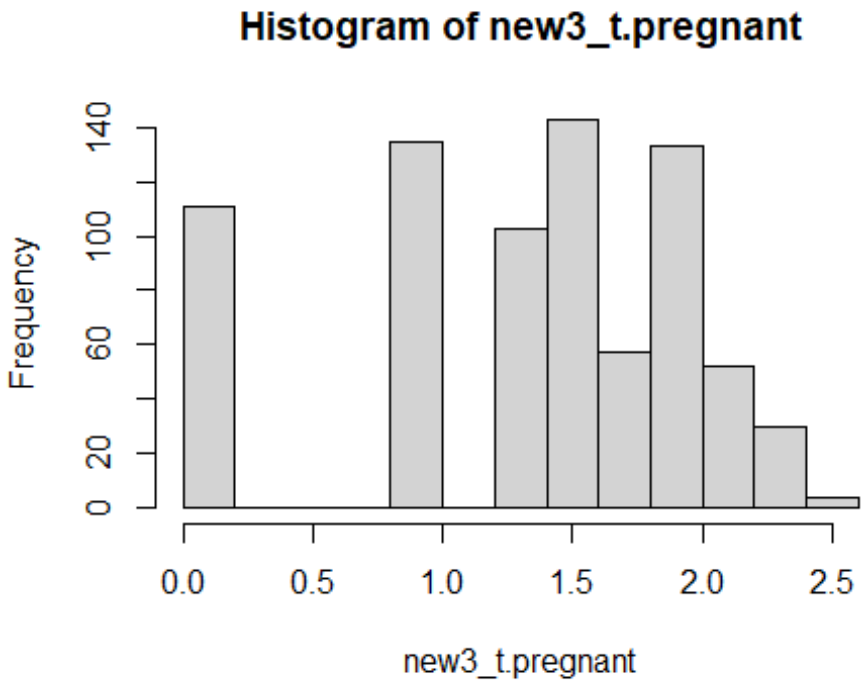

```r
new_t.pregnant <- log(df1$t.pregnant+1)
hist(new_t.pregnant)
```



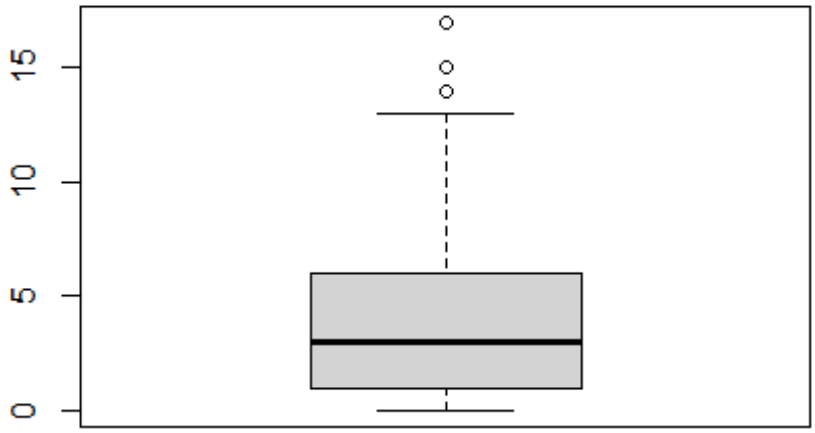

```
# Still no normality
jarque.bera.test(new_t.pregnant)

##
##  Jarque Bera Test
##
## data:  new_t.pregnant
## X-squared = 36.397, df = 2, p-value = 1.249e-08

new2_t.pregnant <- sqrt(df1$t.pregnant)
hist(new2_t.pregnant)
```




```
jarque.bera.test(new2_t.pregnant)

##
##  Jarque Bera Test
##
## data:  new2_t.pregnant
## X-squared = 19.615, df = 2, p-value = 5.504e-05

# Still no normality

new3_t.pregnant <- (df1$t.pregnant)^(1/3)
hist(new3_t.pregnant)
```

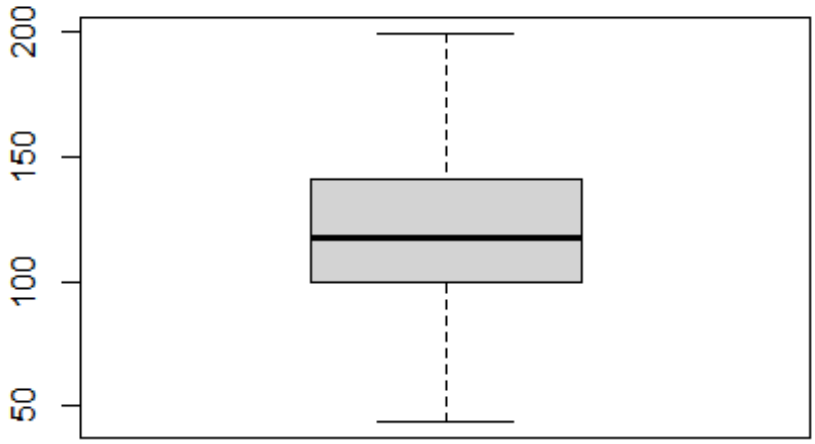



```
jarque.bera.test(new3_t.pregnant)

##
##  Jarque Bera Test
##
## data:  new3_t.pregnant
## X-squared = 81.788, df = 2, p-value < 2.2e-16

# Still no normal
# Since there's no normality transformation for the variable t.pregnant
that means that the multivariate (due to the fact that each and
everyone of the explanatory has to be normal in order for the
multivariate to be normal) isn't going to be normal hence we
# won't have later on at the classification the minimum errors (we
would have the minimum errors under the normality assumption!)

boxplot(t.pregnant)
```

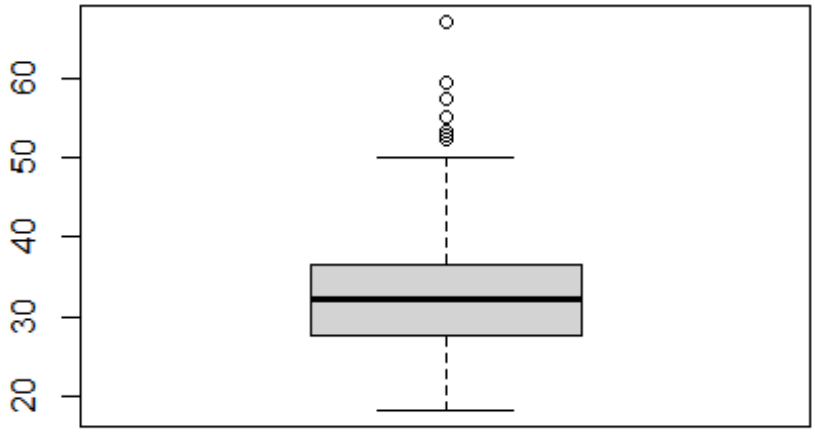



```
# Outliers detected
IQR(t.pregnant)

## [1] 5

# IQR is the range between the first and the third quartiles namely Q1
and Q3: IQR = Q3 - Q1. The data points which fall
# below Q1 - 1.5*IQR or above Q3 + 1.5*IQR are outliers.

Q1_t.pregnant <- quantile(t.pregnant, 0.25)
Q3_t.pregnant <- quantile(t.pregnant, 0.75)
lower_fence_t.pregnant <- Q1_t.pregnant-(3/2)*IQR(t.pregnant)
lower_fence_t.pregnant

##  25%
## -6.5

# Values that fall below this value are outliers
upper_fence_t.pregnant <- Q3_t.pregnant+(3/2)*IQR(t.pregnant)
upper_fence_t.pregnant

##  75%
## 13.5

# Values that fall above this value are outliers
range(t.pregnant)

## [1]  0 17
```

```r
t.pregnant[t.pregnant > 13.5 | t.pregnant < -6.5] <- NA
# Transforming the values that are above the upper fence and below the
lower fence into NAs so we can easily locate them

which(is.na(t.pregnant))

## [1]  89 160 299 456

# The indexes of the values of t.pregnant variable that we assigned to
be NA (in other words the outliers!)
summary(t.pregnant)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.000   1.000   3.000   3.787   6.000  13.000       4

# Checking how many outliers in reality there are

new_df[which(is.na(t.pregnant)),]

##     t.pregnant plasma  bmi  diab class
## 89          15    136 37.1 0.153     1
## 160         17    163 40.9 0.817     1
## 299         14    100 36.6 0.412     1
## 456         14    175 33.6 0.212     1

# View of the new_df rows that there are the outliers for the
explanatory variable t.pregnant

boxplot(plasma)
```




```r
# No outliers detected

boxplot(bmi)
```



```r
# Outliers detected
IQR(bmi)
```

```
## [1] 9.1
```

```r
Q1_bmi <- quantile(bmi, 0.25)
Q3_bmi <- quantile(bmi, 0.75)
lower_fence_bmi <- Q1_bmi-(3/2)*IQR(bmi)
lower_fence_bmi
```

```
##   25%
## 13.85
```

```r
# Values that fall below this value are outliers
upper_fence_bmi <- Q3_bmi+(3/2)*IQR(bmi)
upper_fence_bmi
```

```
##   75%
## 50.25
```

```r
# Values that fall above this value are outliers
range(bmi)
```

```
## [1] 18.2 67.1
```

```r
length(bmi)
```

```
## [1] 768
```

```r
bmi[ bmi > 50.25 | bmi < 13.85] <- NA
head(bmi,10)
```

```
##  [1] 33.6 26.6 23.3 28.1 43.1 25.6 31.0 35.3 30.5 32.3
```

```r
which(is.na(bmi))
```

```
## [1] 121 126 178 194 248 304 446 674
```

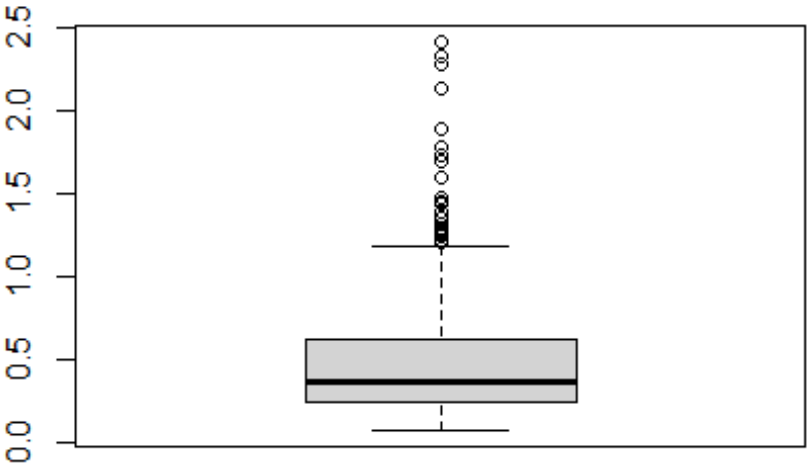
```r
# Which rows of the dataframe will be ignored for the construction of
the classification rule later on
summary(bmi)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   18.20   27.50   32.30   32.21   36.40   50.00       8
```

```r
new_df[which(is.na(bmi)),]
```

```
##     t.pregnant plasma  bmi  diab class
## 121          0    162 53.2 0.759     1
## 126          1     88 55.0 0.496     1
## 178          0    129 67.1 0.319     1
## 194         11    135 52.3 0.578     1
## 248          0    165 52.3 0.427     0
## 304          5    115 52.9 0.209     1
```

```
## 446           0    180 59.4 2.420     1
## 674           3    123 57.3 0.880     0

boxplot(diab)
```




```
# Outliers detected
IQR(diab)

## [1] 0.3825

Q1_diab <- quantile(diab, 0.25)
Q3_diab <- quantile(diab, 0.75)
lower_fence_diab <- Q1_diab-(3/2)*IQR(diab)
lower_fence_diab

##   25%
## -0.33

# Values that fall below this value are outliers
upper_fence_diab <- Q3_diab+(3/2)*IQR(diab)
upper_fence_diab

## 75%
## 1.2

# Values that fall above this value are outliers
range(diab)

## [1] 0.078 2.420
```

```r
diab[ diab > 1.2 | diab < -0.33 ] <- NA
which(is.na(diab))
```

```
##  [1]   5  13  40  46  59 101 148 188 219 229 244 246 260 293 309 331
371 372 384
## [20] 396 446 535 594 607 619 622 623 660 662
```

```r
summary(diab)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  0.0780  0.2380  0.3560  0.4298  0.5870  1.1910      29
```

```r
new_df[which(is.na(diab)),]
```

```
##     t.pregnant plasma  bmi  diab class
## 5            0    137 43.1 2.288     1
## 13          10    139 27.1 1.441     0
## 40           4    111 37.1 1.390     1
## 46           0    180 42.0 1.893     1
## 59           0    146 40.5 1.781     0
## 101          1    163 39.0 1.222     1
## 148          2    106 30.5 1.400     0
## 188          1    128 32.0 1.321     1
## 219          5     85 29.0 1.224     1
## 229          4    197 36.7 2.329     0
## 244          6    119 27.1 1.318     1
## 246          9    184 30.0 1.213     1
## 260         11    155 33.3 1.353     1
## 293          2    128 43.3 1.224     1
## 309          0    128 30.5 1.391     1
## 331          8    118 23.1 1.476     0
## 371          3    173 38.4 2.137     1
## 372          0    118 32.3 1.731     0
## 384          1     90 25.1 1.268     0
## 396          2    127 27.7 1.600     0
## 446          0    180 59.4 2.420     1
## 535          1     77 33.3 1.251     0
## 594          2     82 28.5 1.699     0
## 607          1    181 40.0 1.258     1
## 619          9    112 28.2 1.282     1
## 622          2     92 24.2 1.698     0
## 623          6    183 40.8 1.461     0
## 660          3     80 34.2 1.292     1
## 662          1    199 42.9 1.394     1
```

```r
rows_to_be_ignored <-
c(which(is.na(t.pregnant)),which(is.na(bmi)),which(is.na(diab)))
rows_to_be_ignored
```

```
##  [1]  89 160 299 456 121 126 178 194 248 304 446 674   5  13  40  46
59 101 148
```

```
## [20] 188 219 229 244 246 260 293 309 331 371 372 384 396 446 535 594
607 619 622
## [39] 623 660 662

df_without_outliers <- new_df[-c(rows_to_be_ignored),]
dim(df_without_outliers)

## [1] 728   5

str(df_without_outliers)

## 'data.frame':    728 obs. of  5 variables:
##  $ t.pregnant: int  6 1 8 1 5 3 10 2 8 4 ...
##  $ plasma    : num  148 85 183 89 116 78 115 197 125 110 ...
##  $ bmi       : num  33.6 26.6 23.3 28.1 25.6 31 35.3 30.5 32.3 37.6
...
##  $ diab      : num  0.627 0.351 0.672 0.167 0.201 0.248 0.134 0.158
0.232 0.191 ...
##  $ class     : int  1 0 1 0 0 1 0 1 1 0 ...

which(is.na(df_without_outliers))

## integer(0)

# As expected, zero values hence the outliers have been handled. Thus,
we are ready for the classification
```

Usual correlation check for the (imputed) dataset that has no longer outliers (df_without_outliers)

```
cor(df_without_outliers)

##             t.pregnant     plasma        bmi        diab     class
## t.pregnant 1.000000000 0.13017261 0.04633065 0.003960739 0.2206087
## plasma     0.130172608 1.00000000 0.20951857 0.086820076 0.5043113
## bmi        0.046330654 0.20951857 1.00000000 0.127556451 0.2938781
## diab       0.003960739 0.08682008 0.12755645 1.000000000 0.1649550
## class      0.220608683 0.50431132 0.29387815 0.164955024 1.0000000

# We observe that the correlation values are small, hence the
classification is going to be even harder!
```

Linear regression for the imputed dataset (df_without_outliers)

```
n.class <- df_without_outliers$class
m1 <- lm(n.class~.,data=df_without_outliers[,1:4])
summary(m1)

##
## Call:
## lm(formula = n.class ~ ., data = df_without_outliers[, 1:4])
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -1.01528 -0.27772 -0.07712  0.27708  1.03093
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.0938098  0.0867287 -12.612  < 2e-16 ***
## t.pregnant   0.0223283  0.0044437   5.025 6.36e-07 ***
## plasma       0.0068662  0.0004956  13.854  < 2e-16 ***
## bmi          0.0132909  0.0022973   5.786 1.08e-08 ***
## diab         0.1947848  0.0582661   3.343 0.000871 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3883 on 723 degrees of freedom
## Multiple R-squared:  0.3251, Adjusted R-squared:  0.3213
## F-statistic: 87.06 on 4 and 723 DF,  p-value: < 2.2e-16

# We observe that there are 4 variables (t.pregnant,plasma,bmi,diab)
statistically significant at a=5% significance level
```

Predict the probabilities with respect to the response variable

```
m1.probs=predict(m1,type="response")
m1.probs[1:10]

##           1           2           3           4           6
7
##  0.62506767 -0.06594356  0.78191099 -0.05438264  0.19371417 -
0.03093323
##           8           9          10          11
##  0.41436105  0.73964365  0.41758281  0.28773188

# View the first 10 probabilities
m1.pred=rep(0,728)
```

Use the regression model (m1) to assign the dataset's observations to one of the two classes

```
# If a predicted probability is less that 0.5 the class is predicted as
0, otherwise 1 (which corresponds to non-diabetes, diabetes
respectively)
m1.pred[m1.probs>.5] = 1
table(m1.pred,n.class)

##        n.class
## m1.pred   0   1
##       0 435 110
##       1  51 132

# The assessment is based on calculated misclassification errors or
rates. We can calculate them after the classification and arrange them
in a confusion matrix just as above. Thus, we observe that the total
```

```
misclassification rate is (51+110)/728, which is equal to 22.1%
# The correct classification rate is it's complementary, 77.9%. The
false positive rate is 51/486, which is equal to 10.4%, while the false
negative rate is 110/242, which is equal to 47.4%. In our case, what is
of vital importance is the FNR (positive classified as negative) due to
the fact that we are interested in the women that in reality will
develop diabetes, while we predict that they won't!
```

Splitting the dataset to account for bias using package caret

```
# To detect a machine learning model behavior, we need to use
observations that aren't used in the training process. Otherwise, the
evaluation of the model would be biased. Instead of using the whole
data set we will use the train-test split. The train-test split is a
technique for evaluating the performance of a machine learning
algorithm. It can be used for classification or regression problems and
can be used for any supervised learning algorithm. The procedure
involves taking a dataset and dividing it into two subsets. The first
subset is used to fit the model and is referred to as the training
dataset. The second subset is not used to train the model; instead, the
input element of the dataset is provided to the model, then predictions
are made and compared to the expected values. This second dataset is
referred to as the test dataset.
# Train Dataset: Used to fit the machine learning model.
# Test Dataset: Used to evaluate the fit machine learning model.
# The objective is to estimate the performance of the machine learning
model on new data: data not used to train the model. This is how we
expect to use the model in practice. Namely, to fit it on available
data with known inputs and outputs, then make predictions on new
examples in the future where we do not have the expected output or
target values. The train-test procedure is appropriate when there is a
sufficiently large dataset available. There is no optimal split
percentage, we will use a common split, such as 70(train)-30(test):
library(caret)

##
## Attaching package: 'caret'

## The following object is masked from 'package:survival':
##
##     cluster

set.seed(50)
train.index <- createDataPartition(n.class, p = .7, list = F)
# By default, createDataPartition does a stratified random split of the
data.
train_df_wth_out <- df_without_outliers[ train.index,]
test_df_wth_out  <- df_without_outliers[-train.index,]
```

Linear regression for the training dataset (train_df1)

```r
m2 <- lm(n.class~.,data=df_without_outliers[,1:4]
,subset=as.numeric(rownames(train_df_wth_out)))
# Subset refers to the rows we will pick from the entire dataset
summary(m2)
```

```
##
## Call:
## lm(formula = n.class ~ ., data = df_without_outliers[, 1:4],
##     subset = as.numeric(rownames(train_df_wth_out)))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.99465 -0.26841 -0.06579  0.26374  1.04824
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.1304562  0.1050931 -10.757  < 2e-16 ***
## t.pregnant   0.0196754  0.0055907   3.519 0.000474 ***
## plasma       0.0073360  0.0005928  12.376  < 2e-16 ***
## bmi          0.0130754  0.0027613   4.735 2.89e-06 ***
## diab         0.1840383  0.0707255   2.602 0.009550 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3821 on 480 degrees of freedom
##   (25 observations deleted due to missingness)
## Multiple R-squared:  0.342,  Adjusted R-squared:  0.3366
## F-statistic: 62.38 on 4 and 480 DF,  p-value: < 2.2e-16
```

Predict the probabilities with respect to the response variable

```r
m2.probs=predict(m2,test_df_wth_out,type="response")
m2.probs[1:5]
```

```
##           7           9          14          16          17
## -0.04824482  0.78195884  0.74253476  0.22220558  0.43544772
```

```r
# View the first 5 probabilities
m2.pred=rep(0,218)
```

Use the linear regression model (m2) to assign the test dataset (test_df_wth_out) observations to one of the two classes

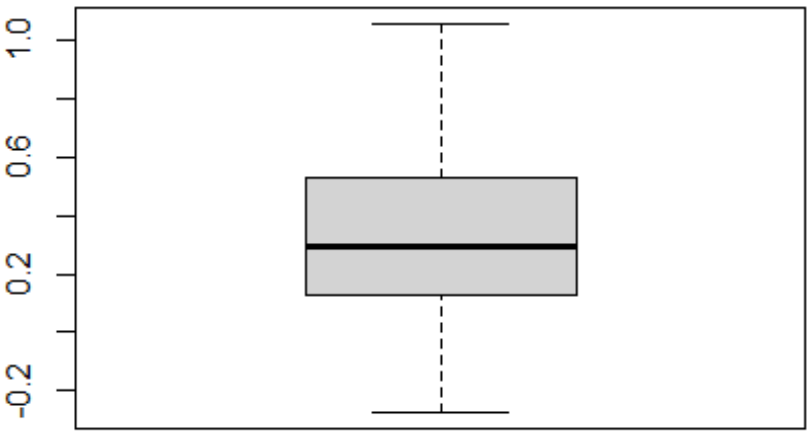
```r
# If a predicted probability is less that 0.5 the class is predicted as
0, otherwise 1 (which correspond to non-diabetes, diabetes
respectively)
m2.pred[m2.probs>.5] = 1
new_class <- test_df_wth_out$class
mean(m2.pred!=new_class)
```

```
## [1] 0.2293578
```

```
# The total misclassification error
table(m2.pred,new_class)

##         new_class
## m2.pred   0   1
##       0 126  34
##       1  16  42

# We observe that the total misclassification rate is (16+34)/218,
which is equal to 22.9% and the correct classification rate is it's
complementary, in other words 77.1%. Now, the FPR is 16/142, which is
equal to 11.2%, while the FNR is 34/76, which is equal to 44.7%
```

Comments on the linear regression model

```
summary(m2.probs)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.2744  0.1292  0.2931  0.3335  0.5259  1.0576

# We observe that there are negative values as well as values greater
than 1, fact that indicates that the linear model isn't the most
appropriate one.
boxplot(m2.probs)
```




```
# We observe the same conclusions as above hence the logistic model
seems as a more appropriate fit, in comparison to the linear one.
```

Fitting a logistic regression model

```
m3 <-
glm(n.class~.,data=df_without_outliers[,1:4],subset=as.numeric(rownames
(train_df_wth_out)),family=binomial)
summary(m3)
```

```
##
## Call:
## glm(formula = n.class ~ ., family = binomial, data =
df_without_outliers[,
##     1:4], subset = as.numeric(rownames(train_df_wth_out)))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5713  -0.6721  -0.3653   0.6394   2.5997
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -10.297372   0.993165 -10.368  < 2e-16 ***
## t.pregnant    0.130511   0.038143   3.422 0.000622 ***
## plasma        0.044123   0.004757   9.275  < 2e-16 ***
## bmi           0.090913   0.020274   4.484 7.32e-06 ***
## diab          1.215839   0.485514   2.504 0.012272 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 612.21  on 484  degrees of freedom
## Residual deviance: 423.60  on 480  degrees of freedom
##   (25 observations deleted due to missingness)
## AIC: 433.6
##
## Number of Fisher Scoring iterations: 5
```

Predict the probabilities with respect to the response variable

```
m3.probs=predict(m3,test_df_wth_out,type="response")
m3.probs[1:5]
```

```
##          7          9         14         16         17
## 0.03407832 0.83490425 0.80102838 0.16034673 0.43606969
```

```
# View the first 5 probabilities
m3.pred=rep(0,218)
```

Use the logistic regression model (m3) to assign the test dataset (test_df_wth_out) observations to one of the two classes

```
# If a predicted probability is less that 0.5 the class is predicted as
0, otherwise 1 (which correspond to non-diabetes, diabetes
respectively)
m3.pred[m3.probs>.5] = 1
# Threshold at 0.5
mean(m3.pred!=new_class)
```

```
## [1] 0.233945
```

```
# The total misclassification rate
table(m3.pred,new_class)
```

```
##         new_class
## m3.pred   0   1
##       0 124  33
##       1  18  43
```

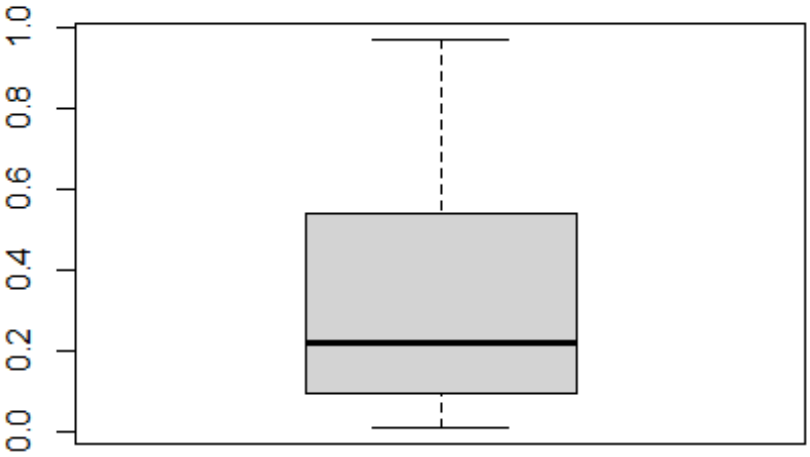
```
# We observe that the total misclassification rate is (18+33)/218,
which is equal to 23.3% and the correct classification rate is the
complementary, in other words it is 76.7%. Now, the FPR is 18/142,
which is equal to 12.6% while the FNR is 33/76, which is equal to 43.4%
```

Comments on the logistic regression model

```
summary(m3.probs)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.008643 0.092304 0.217642 0.335017 0.537736 0.972431
```

```
boxplot(m3.probs)
```




```
# We observe that all values lie in the interval (0,1) which is more
appropriate in our case. Slight improvements observed in comparison
with the linear regression model!
```

Decreasing the FNR

```
m4.probs <- predict(m3,test_df_wth_out,type="response")
m4.probs[1:5]
```

```
##          7          9         14         16         17
## 0.03407832 0.83490425 0.80102838 0.16034673 0.43606969
```

```
# View the first 5 probabilities
m4.pred=rep(0,218)
m4.pred[m4.probs>.3] = 1
# Threshold at 0.3
mean(m4.pred!=new_class)
```

```
## [1] 0.2568807
```

```
# The total misclassification rate
table(m4.pred,new_class)
```

```
##         new_class
## m4.pred   0   1
##       0 110  24
##       1  32  52
```

```
# We observe that the FNR is 24/76, which is equal to 31.5% ( while
before it was 43.4% !)
```

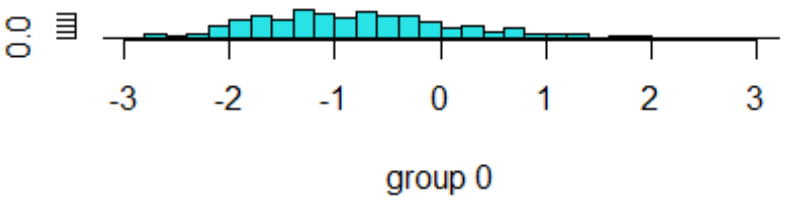
Further decreasing the FNR

```
m5.probs <- predict(m3,test_df_wth_out,type="response")
m5.probs[1:5]

##          7          9         14         16         17
## 0.03407832 0.83490425 0.80102838 0.16034673 0.43606969

# View the first 5 probabilities
m5.pred=rep(0,218)
m5.pred[m5.probs>.1] = 1
# Setting the threshold at 0.1
table(m5.pred,new_class)

##        new_class
## m5.pred  0  1
##       0 55  3
##       1 87 73

# We observe that the FNR is 3/76, which is equal to 0.04%! Now ,the
total misclassification rate is (87+3)/218, which is equal to 41.2%
```

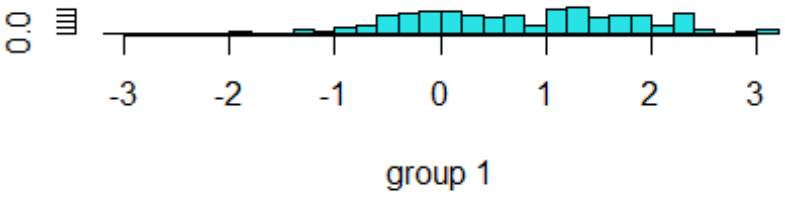
Linear Discriminant Analysis

```
library(MASS)

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select

lda.fit <-
lda(n.class~.,data=df_without_outliers[,1:4],subset=as.numeric(rownames
(train_df_wth_out)) )
lda.fit

## Call:
## lda(n.class ~ ., data = df_without_outliers[, 1:4], subset =
as.numeric(rownames(train_df_wth_out)))
##
## Prior probabilities of groups:
##         0         1
## 0.6742268 0.3257732
##
## Group means:
##   t.pregnant   plasma      bmi      diab
## 0   3.192661 109.8003 30.83578 0.3983089
```

```
## 1   4.531646 143.7745 34.80759 0.4772278
##
## Coefficients of linear discriminants:
##                    LD1
## t.pregnant 0.08831365
## plasma     0.03292777
## bmi        0.05868955
## diab       0.82606317

plot(lda.fit)
```



group 0



group 1

```
# The plot of scores of train data set on the (one in this case) linear
discriminant functions
lda.pred=predict(lda.fit, test_df_wth_out)
names(lda.pred)

## [1] "class"     "posterior" "x"

lda.class=lda.pred$class
table(lda.class,new_class)

##          new_class
## lda.class   0   1
##         0 124  33
##         1  18  43

# We observe that the total misclassification rate is (18+33)/218,
which is equal to 23.3%. The FNR is 33/76, which is equal to 43.4% and
```

```
# the FPR is 18/142, which is equal to 12.6%
mean(lda.class==new_class)

## [1] 0.766055

# The correct test classification rate (which is the complementary of
the total missclassification rate)
mean(lda.class!=new_class)

## [1] 0.233945

# The total misclassification rate
```

Second way using package hmeasure

```
library(hmeasure)
lda.counts <- misclassCounts(lda.class,new_class)
# Computes a set of classification performance metrics that rely on a
set of predicted labels and a set of true labels as input.
# All the measures computed here are scalar summaries of the confusion
matrix, which consists of the number of True Positives (TPs),
# False Positives (FPs), True Negatives (TNs), and False Negatives
(FNs). The most common such summary is the Error Rate (ER).
# Additionally the following metrics are reported: the True Positive
Rate (TPR) and the False Positive Rate (FPR), Sensitivity
# (same as TPR) versus Specificity (given by 1-FPR), and yet another
such pair is Precision versus Recall (same as Sensitivity).
# Finally, the measure and the Youden index are scalar measures that
attempt to take a more balanced view of the two different objectives
# than ER does. The former is given by the harmonic mean of Precision
and Recall, and the latter by Sens+Spec-1.
lda.counts$conf.matrix

##          pred.1 pred.0
## actual.1     43     33
## actual.0     18    124

print(lda.counts$metrics,digits=3)

##      ER  Sens  Spec Precision Recall   TPR   FPR     F Youden
## 1 0.234 0.566 0.873     0.705  0.566 0.566 0.127 0.628  0.439
```

Quadratic Discriminant Analysis

```
qda.fit=qda(n.class~.,data=df_without_outliers[,1:4],subset=as.numeric(
rownames(train_df_wth_out)) )
qda.fit

## Call:
## qda(n.class ~ ., data = df_without_outliers[, 1:4], subset =
as.numeric(rownames(train_df_wth_out)))
##
```

```
## Prior probabilities of groups:
##         0         1
## 0.6742268 0.3257732
##
## Group means:
##   t.pregnant   plasma      bmi      diab
## 0   3.192661 109.8003 30.83578 0.3983089
## 1   4.531646 143.7745 34.80759 0.4772278

qda.class=predict(qda.fit,test_df_wth_out)$class
table(qda.class,new_class)

##          new_class
## qda.class   0   1
##         0 127  34
##         1  15  42

# We observe that the total misclassification rate is (15+34)/218,
which is equal to 22.4%. The FNR is 34/76, which is equal to 44.7% and
the FPR is 15/142, which is equal to 10.5%
mean(qda.class==new_class)

## [1] 0.7752294

# The correct test classification rate
mean(qda.class!=new_class)

## [1] 0.2247706

# The total classification rate
```

Second way using package hmeasure

```
qda.counts <- misclassCounts(qda.class,new_class)
qda.counts$conf.matrix

##          pred.1 pred.0
## actual.1     42     34
## actual.0     15    127

print(qda.counts$metrics,digits = 3)

##      ER  Sens  Spec Precision Recall   TPR   FPR     F Youden
## 1 0.225 0.553 0.894     0.737  0.553 0.553 0.106 0.632  0.447

HMeasure(new_class,test_df_wth_out[,1:4], severity.ratio = NA,
threshold=0.5, level=0.95)$metrics

##                    H      Gini       AUC      AUCH        KS
MER
## t.pregnant 0.05926238 0.1848592 0.5924296 0.6160119 0.1843958
0.3394495
## plasma     0.37008339 0.6191623 0.8095812 0.8257969 0.4772053
```

```
0.2110092
## bmi        0.17877987 0.3256116 0.6628058 0.7030671 0.2839140
0.3073394
## diab       0.04251774 0.0313195 0.5156597 0.5726001 0.1289844
0.3440367
##                  MWL Spec.Sens95 Sens.Spec95        ER      Sens
Spec
## t.pregnant 0.3704234  0.08544601  0.04342105 0.5963303 0.8815789
0.1478873
## plasma     0.2374379  0.36901408  0.49868421 0.6513761 1.0000000
0.0000000
## bmi        0.3252251  0.30422535  0.19605263 0.6513761 1.0000000
0.0000000
## diab       0.3955896  0.05774648  0.02631579 0.4266055 0.2763158
0.7323944
##            Precision    Recall       TPR       FPR         F
Youden TP  FP
## t.pregnant 0.3563830 0.8815789 0.8815789 0.8521127 0.5075758
0.029466271 67 121
## plasma     0.3486239 1.0000000 1.0000000 1.0000000 0.5170068
0.000000000 76 142
## bmi        0.3486239 1.0000000 1.0000000 1.0000000 0.5170068
0.000000000 76 142
## diab       0.3559322 0.2763158 0.2763158 0.2676056 0.3111111
0.008710156 21  38
##             TN FN
## t.pregnant  21  9
## plasma       0  0
## bmi          0  0
## diab       104 55

# The HMeasure function outputs an object of class "hmeasure", with one
field named "metrics"that reports several performance metrics in the
form of a data frame with one row per classifier, and an attribute
named "data" which preserves useful information (such as the empirical
scoring distributions) for plotting purposes. The H-measure naturally
requires as input a severity ratio, which represents how much more
severe misclassifying a class 0 instance is than misclassifying a class
1 instance. Formally, this determines the mode of the prior over costs
that underlies the H-measure (see package vignette or references for
more information). We may write SR = c_0/c_1, where c_0 > 0 is the cost
of misclassifying a class 0 datapoint as class 1. It is sometimes more
convenient to consider instead the normalised cost c = c_0/(c_0 + c_1),
so that SR = c/(1-c) where c is in [0,1]. For instance, severity.ratio
= 2 implies that a False Positive costs twice as much as a False
Negative. By default the severity ratio is set to be reciprocal of
relative class frequency, i.e., severity.ratio = pi1/pi0, so that
misclassifying the rare class is considered a graver mistake. See Hand
2012 for a more detailed motivation of this default. The metrics
reported can be broken down into two types. The first type consists of
```

```
metrics that measure the match between a set of predicted labels and
the true labels. We obtain these predictions using the scores provided
and employing a user-specified threshold (or thresholds, one per
classifier), if provided, otherwise a default of 0.5. See
help(misclassCounts) for a list of the metrics computed. The second
type of measures are aggregate measures of performance that do not rely
on the user to specify the threshold, and instead operate directly on
the classification scores themselves. In this sense, they are more
useful for performance comparisons across classifiers and datasets. The
aggregate metrics currently reported include: the Area under the ROC
Curve (AUC), the H-measure, the Area under the Convex Hull of the ROC
Curve (AUCH), the Gini coefficient, the Kolmogorov-Smirnoff (KS)
statistic, the Minimum Weighted Loss (MWL), the Minimum Total Loss
(MTL), as well as the Sensitivity at 95% Specificity ("Sens95"), and
the Specificity at 95% Sensitivity ("Spec95"). For these latter
measures, a 95% level is the default, but alternative or additional
values may be specified using the "level" argument. The package
vignette contains a very a detailed explanation of each of the above
metrics, and their relationships with each other.
```

KNN

```
library(class)
set.seed(20)
train.class=train_df_wth_out$class
knn.pred1=knn(train_df_wth_out,test_df_wth_out,train.class,k=1)
# Number of neighbours considered k=1
table(knn.pred1,new_class)
```

```
##          new_class
## knn.pred1   0   1
##         0 115  31
##         1  27  45
```

```
# We observe that the total misclassification rate is (27+31)/218,
which is equal to 26.6%. The FNR is 31/76, which is equal to 40.7%
while the FPR is 27/142, which is equal to 19%
mean(knn.pred1==new_class)
```

```
## [1] 0.733945
```

```
# The correct classification rate
mean(knn.pred1!=new_class)
```

```
## [1] 0.266055
```

```
# The total misclassification rate


knn.pred2=knn(train_df_wth_out,test_df_wth_out,train.class,k=3)
```

```
# Number of neighbours considered k=3
table(knn.pred2,new_class)
```

```
##          new_class
## knn.pred2   0   1
##         0 112  29
##         1  30  47
```

```
# We observe that the total misclassification rate is (30+29)/218,
which is equal to 27%. The FNR is 29/76, which is equal to 38.1%  while
the FPR is 30/142, which is equal to 21.1%
mean(knn.pred2==new_class)
```

```
## [1] 0.7293578
```

```
# The correct classification rate
mean(knn.pred2!=new_class)
```

```
## [1] 0.2706422
```

```
# The total misclassification rate
```

Second dataset (df2) analysis

```
head(df2,5)
```

```
##    t.pregnant plasma bl.press tr.thick serum.ins  bmi  diab age
class
## 4           1     89       66       23        94 28.1 0.167  21
0
## 5           0    137       40       35       168 43.1 2.288  33
1
## 7           3     78       50       32        88 31.0 0.248  26
1
## 9           2    197       70       45       543 30.5 0.158  53
1
## 14          1    189       60       23       846 30.1 0.398  59
1
```

```
which(is.na(df2))
```

```
## integer(0)
```

```
# Zero, as expected there are no missing values in the dataframe
summary(df2)
```

```
##    t.pregnant         plasma         bl.press         tr.thick
##  Min.   : 0.000   Min.   : 56.0   Min.   : 24.00   Min.   : 7.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.:21.00
##  Median : 2.000   Median :119.0   Median : 70.00   Median :29.00
##  Mean   : 3.301   Mean   :122.6   Mean   : 70.66   Mean   :29.15
##  3rd Qu.: 5.000   3rd Qu.:143.0   3rd Qu.: 78.00   3rd Qu.:37.00
##  Max.   :17.000   Max.   :198.0   Max.   :110.00   Max.   :63.00
```

```
##    serum.ins           bmi             diab             age        
##  Min.   : 14.00   Min.   :18.20   Min.   :0.0850   Min.   :21.00  
##  1st Qu.: 76.75   1st Qu.:28.40   1st Qu.:0.2697   1st Qu.:23.00  
##  Median :125.50   Median :33.20   Median :0.4495   Median :27.00  
##  Mean   :156.06   Mean   :33.09   Mean   :0.5230   Mean   :30.86  
##  3rd Qu.:190.00   3rd Qu.:37.10   3rd Qu.:0.6870   3rd Qu.:36.00  
##  Max.   :846.00   Max.   :67.10   Max.   :2.4200   Max.   :81.00  
##      class       
##  Min.   :0.0000  
##  1st Qu.:0.0000  
##  Median :0.0000  
##  Mean   :0.3316  
##  3rd Qu.:1.0000  
##  Max.   :1.0000
```

```
cor(df2)
```

```
##              t.pregnant    plasma   bl.press  tr.thick  serum.ins
bmi
## t.pregnant  1.000000000 0.1982910  0.2133548 0.0932094 0.07898363 -
0.02534728
## plasma      0.198291043 1.0000000  0.2100266 0.1988558 0.58122301
0.20951592
## bl.press    0.213354775 0.2100266  1.0000000 0.2325712 0.09851150
0.30440337
## tr.thick    0.093209397 0.1988558  0.2325712 1.0000000 0.18219906
0.66435487
## serum.ins   0.078983625 0.5812230  0.0985115 0.1821991 1.00000000
0.22639652
## bmi        -0.025347276 0.2095159  0.3044034 0.6643549 0.22639652
1.00000000
## diab        0.007562116 0.1401802 -0.0159711 0.1604985 0.13590578
0.15877104
## age         0.679608470 0.3436415  0.3000389 0.1677611 0.21708199
0.06981380
## class       0.256565956 0.5157027  0.1926733 0.2559357 0.30142922
0.27011841
##                    diab        age     class
## t.pregnant  0.007562116 0.67960847 0.2565660
## plasma      0.140180180 0.34364150 0.5157027
## bl.press   -0.015971104 0.30003895 0.1926733
## tr.thick    0.160498526 0.16776114 0.2559357
## serum.ins   0.135905781 0.21708199 0.3014292
## bmi         0.158771043 0.06981380 0.2701184
## diab        1.000000000 0.08502911 0.2093295
## age         0.085029106 1.00000000 0.3508038
## class       0.209329511 0.35080380 1.0000000
```

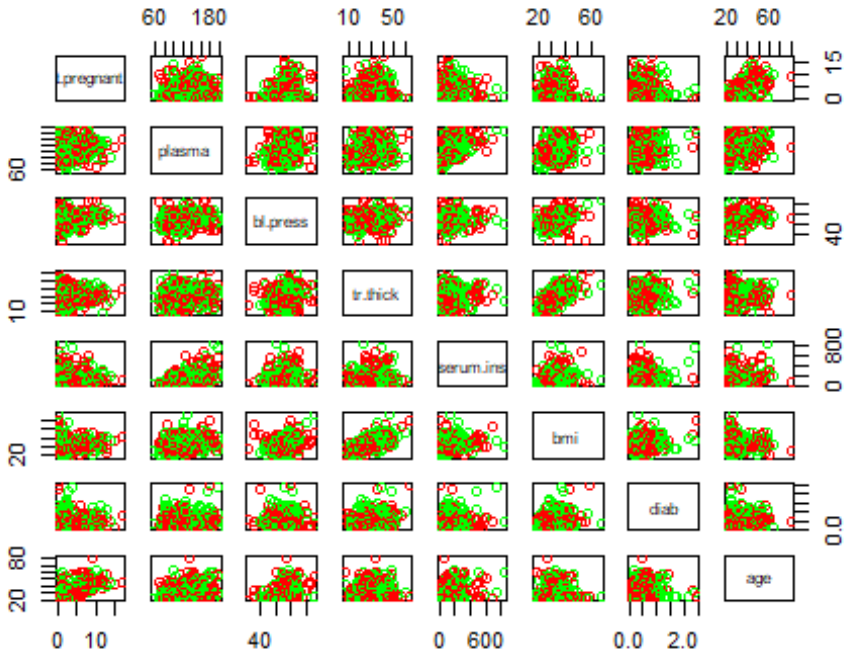
```
dim(df2)
```

```
## [1] 392   9
```

```
# The correlation values are small hence the classification is going to
be even harder!
```

Pairwise colorful plots

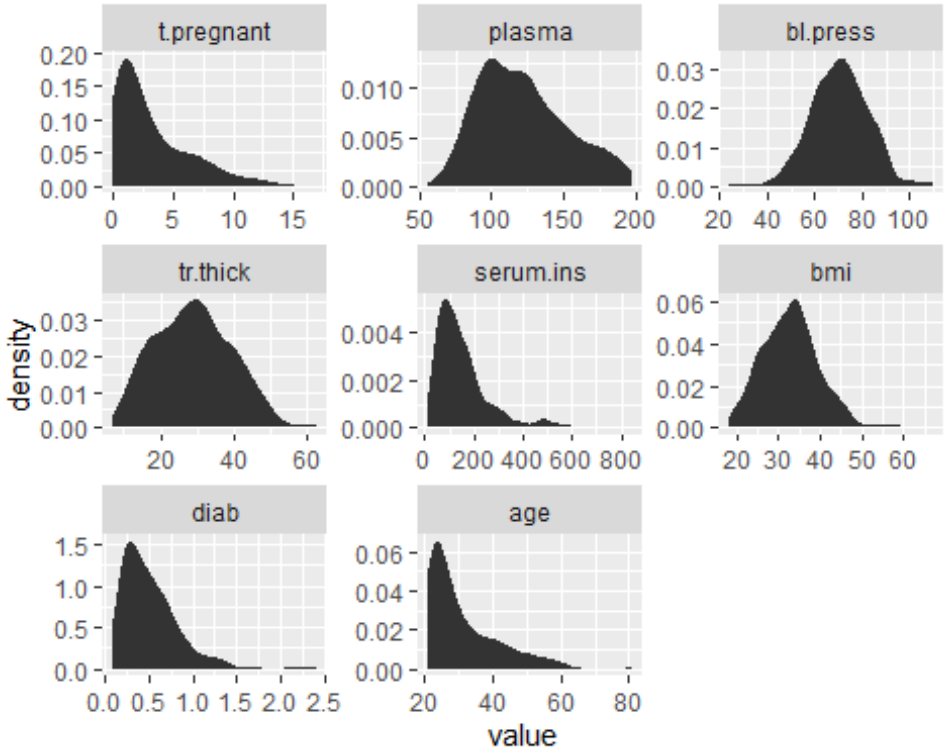
```
cols2 <- character(nrow(df2))
cols2[] <- "black"
cols2[df2$class %in% c(0,1)] <- c("green","red")
pairs(df2[,-9],col=cols2)
```




```
# We observe that there's no conspicuous discrimination available
```

Data visualization using ggplot

```
library(reshape2)
newdataframe <- melt(df2[,-9])

## No id variables; using all as measure variables

ggplot(data = newdataframe, aes(x = value)) +
  stat_density() +
  facet_wrap(~variable, scales = "free")
```




```
# Small multiple chart
```

Linear regression for the second dataset (df2)

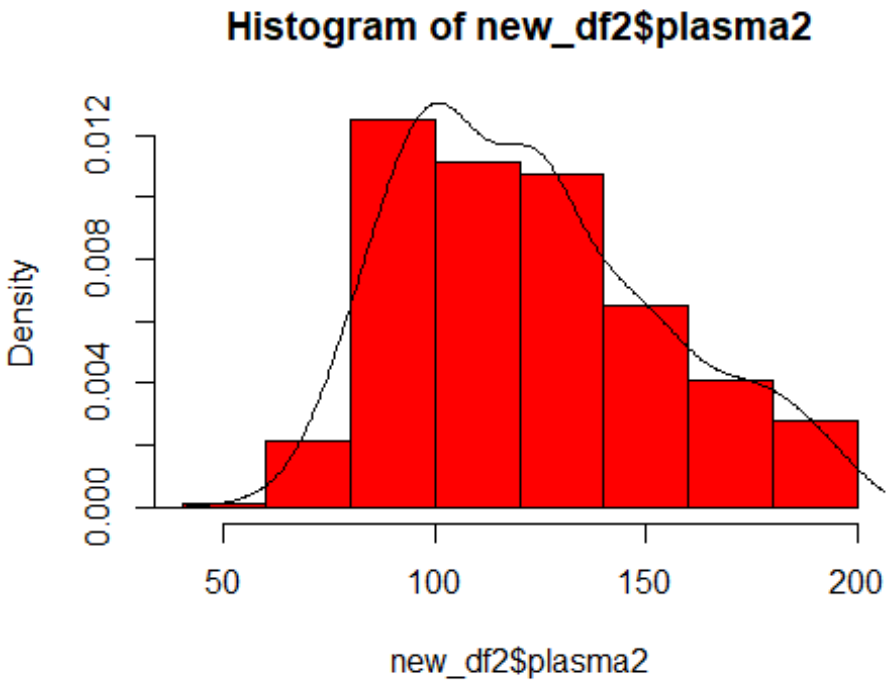
```
class2 <- df2$class
model1 <- lm(class2~.,data=df2[,1:8])
summary(model1)
```

```
##
## Call:
## lm(formula = class2 ~ ., data = df2[, 1:8])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.07966 -0.25711 -0.06177  0.25851  1.03750
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.103e+00  1.436e-01  -7.681 1.34e-13 ***
## t.pregnant   1.295e-02  8.364e-03   1.549  0.12230
## plasma       6.409e-03  8.159e-04   7.855 4.07e-14 ***
## bl.press     5.465e-05  1.730e-03   0.032  0.97482
## tr.thick     1.678e-03  2.522e-03   0.665  0.50631
## serum.ins   -1.233e-04  2.045e-04  -0.603  0.54681
## bmi          9.325e-03  3.901e-03   2.391  0.01730 *
## diab         1.572e-01  5.804e-02   2.708  0.00707 **
## age          5.878e-03  2.787e-03   2.109  0.03559 *
```

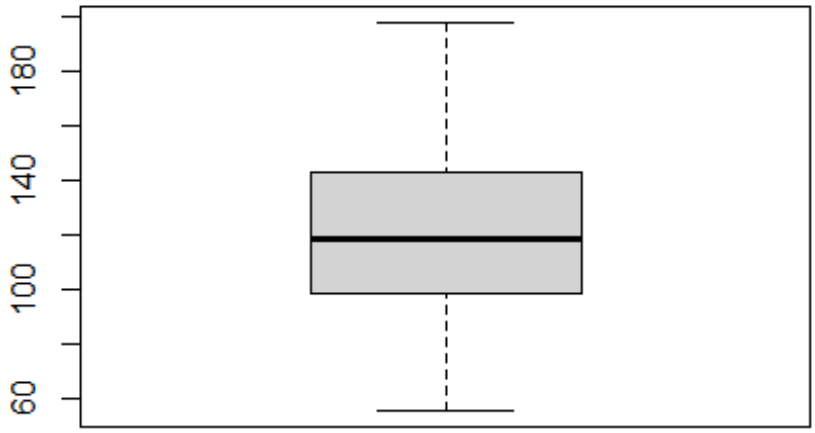
```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3853 on 383 degrees of freedom
## Multiple R-squared:  0.3458, Adjusted R-squared:  0.3321
## F-statistic:  25.3 on 8 and 383 DF,  p-value: < 2.2e-16

# We observe that there are 4 variables (plasma,bmi,diab,age)
statistically significant at a=5% significance level
plasma2 <- df2$plasma
bmi2 <- df2$bmi
diab2 <- df2$diab
age2 <- df2$age
class2 <- df2$class
new_df2 <- data.frame(plasma2,bmi2,diab2,age2,class2)
```

Data preprocessing for the significant explanatory variables of df2

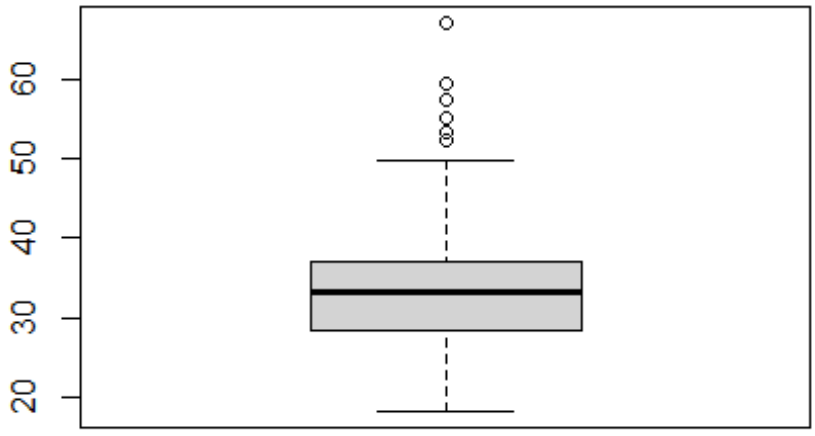
```
jarque.bera.test(new_df2$plasma2)

##
##  Jarque Bera Test
##
## data:  new_df2$plasma2
## X-squared = 21.346, df = 2, p-value = 2.316e-05

# Reject Ho at 5% level of significance hence no normality
shapiro.test(new_df2$plasma2)

##
##  Shapiro-Wilk normality test
##
## data:  new_df2$plasma2
## W = 0.96423, p-value = 3.442e-08

# Second way, same results
hist(new_df2$plasma2,col='red',freq=F)
# Third way to check it visually
lines(density(new_df2$plasma2))
```




```
# Since there's no normality transformation for the variable t.pregnant
that means that the multivariate (due to the fact that each and
everyone of the explanatory has to be normal in order for the
multivariate to be normal) isn't going to be normal hence we won't have
later on at the classification the minimum errors (we would have the
minimum errors under the normality assumption!)


boxplot(plasma2)
```




```r
# No outliers detected
boxplot(bmi2)
```



```r
# Outliers detected
IQR(bmi2)
```

```
## [1] 8.7
```

```r
Q1_bmi2 <- quantile(bmi2, 0.25)
Q3_bmi2 <- quantile(bmi2, 0.75)
lower_fence_bmi2 <- Q1_bmi2-(3/2)*IQR(bmi2)
lower_fence_bmi2
```

```
##   25%
## 15.35
```

```r
# Values that fall below this value are outliers
upper_fence_bmi2 <- Q3_bmi2+(3/2)*IQR(bmi2)
upper_fence_bmi2
```

```
##   75%
## 50.15
```

```r
# Values that fall above this value are outliers
range(bmi2)
```

```
## [1] 18.2 67.1
```

```r
bmi2[bmi2 > 50.15 | bmi2 <15.35 ] <- NA
# Transforming the values that are above the upper fence and below the
lower fence into NAs so we can easily locate them

which(is.na(bmi2))
```

```
## [1]  56  58  87 120 226 349
```

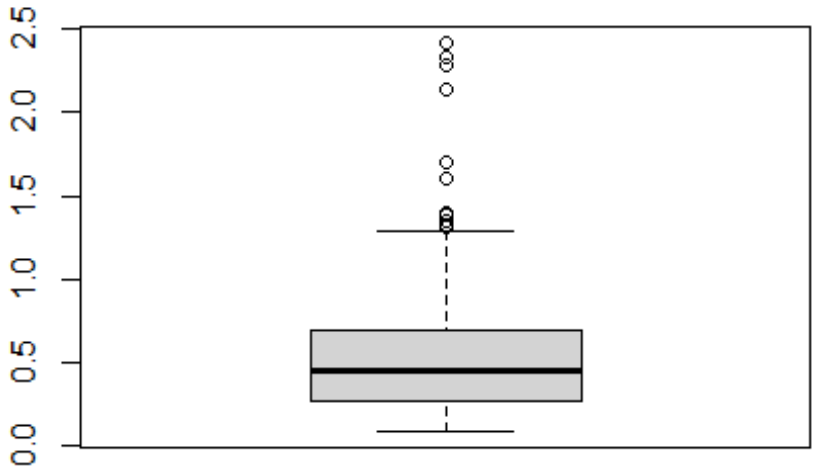
```r
# The indexes of the values of diab2 variable that we assigned to be NA
(in other words the outliers!)
summary(bmi2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   18.20   28.32   33.10   32.71   36.88   49.70       6
```

```r
# Checking how many outliers in reality there are

new_df2[which(is.na(bmi2)),]
```

```
##     plasma2 bmi2 diab2 age2 class2
## 56      162 53.2 0.759   25      1
## 58       88 55.0 0.496   26      1
## 87      129 67.1 0.319   26      1
## 120     165 52.3 0.427   23      0
## 226     180 59.4 2.420   25      1
## 349     123 57.3 0.880   22      0
```

```r
# View of the new_df2 rows that there are the outliers for the
explanatory variable bmi2


boxplot(diab2)
```




```r
# Outliers detected
IQR(diab2)

## [1] 0.41725

Q1_diab2 <- quantile(diab2, 0.25)
Q3_diab2 <- quantile(diab2, 0.75)
lower_fence_diab2 <- Q1_diab2-(3/2)*IQR(diab2)
lower_fence_diab2

##       25%
## -0.356125

# Values that fall below this value are outliers
upper_fence_diab2 <- Q3_diab2+(3/2)*IQR(diab2)
upper_fence_diab2

##      75%
## 1.312875

# Values that fall above this value are outliers
range(diab2)
```

```
## [1] 0.085 2.420
```

```
diab2[diab2 > 1.312875 | diab2 < -0.356125  ] <- NA
# Transforming the values that are above the upper fence and below the
lower fence into NAs so we can easily locate them

which(is.na(diab2))
```

```
##  [1]   2  18  71  90 111 118 125 153 185 203 226 305
```

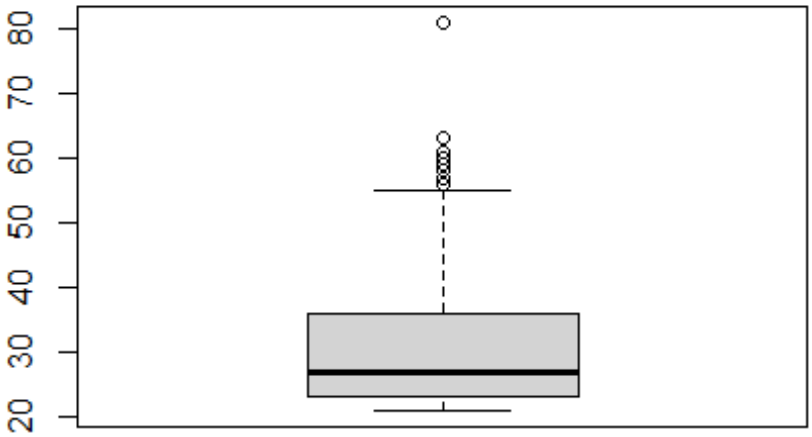
```
# The indexes of the values of diab2 variable that we assigned to be NA
(in other words the outliers!)
summary(diab2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  0.0850  0.2680  0.4365  0.4852  0.6587  1.2920      12
```

```
# Checking how many outliers in reality there are

new_df2[which(is.na(diab2)),]
```

```
##     plasma2 bmi2 diab2 age2 class2
## 2       137 43.1 2.288   33      1
## 18      111 37.1 1.390   56      1
## 71      106 30.5 1.400   34      0
## 90      128 32.0 1.321   33      1
## 111     197 36.7 2.329   31      0
## 118     119 27.1 1.318   33      1
## 125     155 33.3 1.353   51      1
## 153     128 30.5 1.391   25      1
## 185     173 38.4 2.137   25      1
## 203     127 27.7 1.600   25      0
## 226     180 59.4 2.420   25      1
## 305      82 28.5 1.699   25      0
```

```
# View of the new_df2 rows that there are the outliers for the
explanatory variable diab2

boxplot(age2)
```




```
# Outliers detected
IQR(age2)

## [1] 13

Q1_age2 <- quantile(age2, 0.25)
Q3_age2 <- quantile(age2, 0.75)
lower_fence_age2 <- Q1_age2-(3/2)*IQR(age2)
lower_fence_age2

## 25%
## 3.5

# Values that fall below this value are outliers
upper_fence_age2 <- Q3_age2+(3/2)*IQR(age2)
upper_fence_age2

##  75%
## 55.5

# Values that fall above this value are outliers
range(age2)

## [1] 21 81

age2[age2 > 55.5 | age2 < 3.5  ] <- NA
# Transforming the values that are above the upper fence and below the
lower fence into NAs so we can easily locate them
```

```
which(is.na(age2))
```

```
##  [1]   5  14  18  24  89 100 108 189 198 236 251 266 391
```

```
# The indexes of the values of age2 variable that we assigned to be NA
(in other words the outliers!)
summary(age2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   21.00   23.00   27.00   29.85   34.00   55.00      13
```

```
# Checking how many outliers in reality there are

new_df2[which(is.na(age2)),]
```

```
##     plasma2 bmi2 diab2 age2 class2
## 5       189 30.1 0.398   59      1
## 14      145 22.2 0.245   57      0
## 18      111 37.1 1.390   56      1
## 24      176 33.7 0.467   58      1
## 89      181 30.1 0.615   60      1
## 100     196 37.5 0.605   57      1
## 108     142 28.8 0.687   61      0
## 189     140 39.2 0.528   58      1
## 198     144 32.0 0.452   58      1
## 236     134 25.9 0.460   81      0
## 251     173 46.5 1.159   58      0
## 266     129 19.6 0.582   60      0
## 391     101 32.9 0.171   63      0
```

```
# View of the new_df2 rows (that there are the outliers) for the
explanatory variable age2

rows_to_be_ignored2 <-
c(which(is.na(bmi2)),which(is.na(diab2)),which(is.na(age2)))
rows_to_be_ignored2
```

```
##  [1]  56  58  87 120 226 349   2  18  71  90 111 118 125 153 185 203
226 305   5
## [20]  14  18  24  89 100 108 189 198 236 251 266 391
```

```
df2_without_outliers <- new_df2[-c(rows_to_be_ignored2),]
dim(df2_without_outliers)
```

```
## [1] 363   5
```

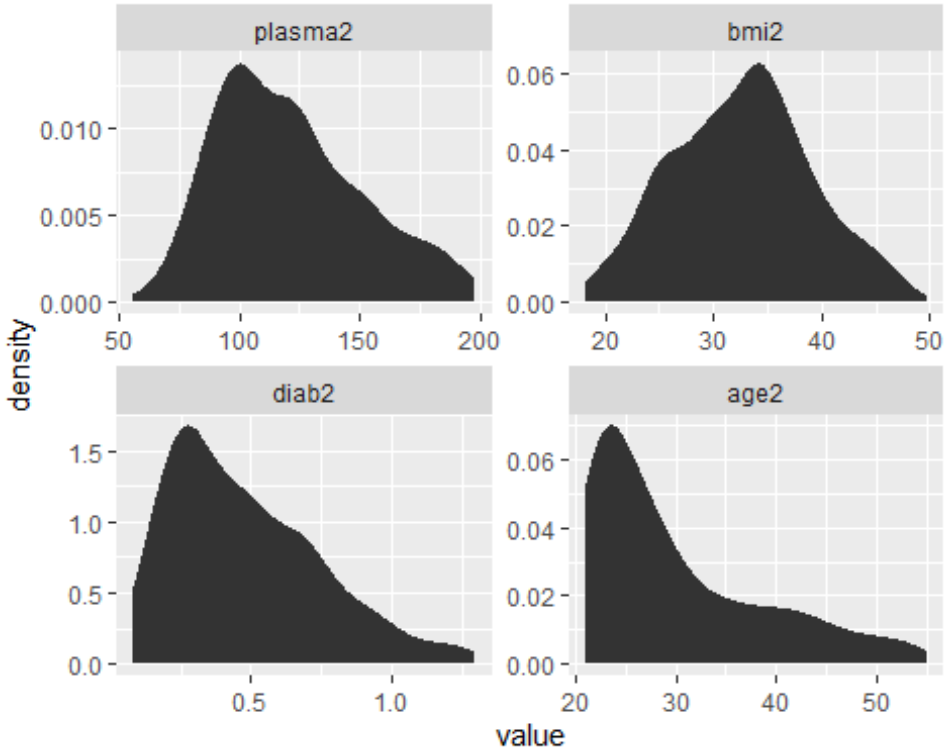
```
str(df2_without_outliers)
```

```
## 'data.frame':    363 obs. of  5 variables:
##  $ plasma2: int  89 78 197 166 118 103 115 126 143 125 ...
##  $ bmi2   : num  28.1 31 30.5 25.8 45.8 43.3 34.6 39.3 36.6 31.1 ...
```

```
##  $ diab2  : num  0.167 0.248 0.158 0.587 0.551 0.183 0.529 0.704
0.254 0.205 ...
##  $ age2   : int  21 26 53 51 31 33 32 27 51 41 ...
##  $ class2 : int  0 1 1 1 1 0 1 0 1 1 ...

# 363 obs

which(is.na(df2_without_outliers))

## integer(0)

# As expected, zero values hence the outliers have been handled. Thus,
we are ready for the classification

unique(rows_to_be_ignored2)

##  [1]  56  58  87 120 226 349   2  18  71  90 111 118 125 153 185 203
305   5  14
## [20]  24  89 100 108 189 198 236 251 266 391

# Check that we got the right amount of data

length(rows_to_be_ignored2)

## [1] 31

length(unique(rows_to_be_ignored2))

## [1] 29

# As expected the difference between the rows_to_be_ignored2 and the
unique(rows_to_be_ignored2) is 2
```

Data visualization after the data processing

```
library(reshape2)
newdataframe2 <- melt(df2_without_outliers[,-5])

## No id variables; using all as measure variables

ggplot(data = newdataframe2, aes(x = value)) +
  stat_density() +
  facet_wrap(~variable, scales = "free")
```




```
# Small multiple chart
```

Splitting the dataset (df2) to account for bias using package caret

```
library(caret)
set.seed(50)
n.class2 <- df2_without_outliers$class2
train.index2 <- createDataPartition(n.class2, p = .7, list = F)
# By default, createDataPartition does a stratified random split of the
data.
train2_df_wth_out <- df2_without_outliers[ train.index2,]
test2_df_wth_out  <- df2_without_outliers[-train.index2,]
```

Linear regression for the training dataset (train2_df1)

```
model2 <- lm(n.class2~.,data=df2_without_outliers[,1:4]
,subset=as.numeric(rownames(train2_df_wth_out)))
# Subset refers to the rows we will pick from the entire dataset
summary(model2)

##
## Call:
## lm(formula = n.class2 ~ ., data = df2_without_outliers[, 1:4],
##     subset = as.numeric(rownames(train2_df_wth_out)))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.88612 -0.22467 -0.04813  0.19442  0.97754
```

```
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.1268881  0.1415466  -7.961 7.78e-14 ***
## plasma2      0.0067877  0.0007851   8.646 9.24e-16 ***
## bmi2         0.0037987  0.0038214   0.994   0.3212
## diab2        0.2004602  0.0866847   2.313   0.0216 *
## age2         0.0130392  0.0027305   4.775 3.19e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3545 on 230 degrees of freedom
##   (20 observations deleted due to missingness)
## Multiple R-squared:  0.4212, Adjusted R-squared:  0.4112
## F-statistic: 41.85 on 4 and 230 DF,  p-value: < 2.2e-16
```

Predict the probabilities with respect to the response variable

```
model2.probs=predict(model2,test2_df_wth_out,type="response")
model2.probs[1:5]

##        20        25        27        32        40
## 1.0549110 0.7146527 0.3267138 0.2157513 0.2002764

# View the first 5 probabilities
model2.pred=rep(0,108)
```

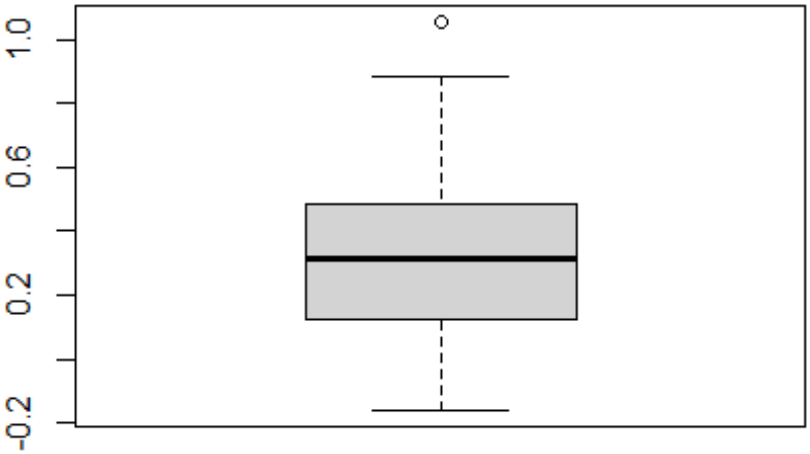
Use the linear regression model (model2) to assign the test dataset (test2_df_wth_out) observations to one of the two classes

```
# If a predicted probability is less that 0.5 the class is predicted as
0, otherwise 1 (which correspond to non-diabetes, diabetes
respectively)
model2.pred[model2.probs>.5] = 1
new_class2 <- test2_df_wth_out$class
mean(model2.pred!=new_class2)

## [1] 0.2777778

# The total misclassification error
table(model2.pred,new_class2)

##            new_class2
## model2.pred  0  1
##           0 62 19
##           1 11 16
```

Comments on the linear regression model

```
summary(model2.probs)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.1604  0.1221  0.3106  0.3189  0.4747  1.0549

# We observe that there are negative values as well as values greater
than 1, fact that indicates that the linear model isn't the most
# appropriate one.
boxplot(model2.probs)
```




```
# We observe the same conclusions as above hence the logistic model
seems as a more appropriate fit, in comparison to the linear one.
```

Fitting a logistic regression model

```
model3 <-
glm(n.class2~.,data=df2_without_outliers[,1:4],subset=as.numeric(rownam
es(train2_df_wth_out)),family=binomial)
summary(model3)

##
## Call:
## glm(formula = n.class2 ~ ., family = binomial, data =
df2_without_outliers[,
##     1:4], subset = as.numeric(rownames(train2_df_wth_out)))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3684  -0.5310  -0.2949   0.4167   2.5599
##
```

```
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -11.475340   1.616719  -7.098 1.27e-12 ***
## plasma2       0.046052   0.007312   6.298 3.01e-10 ***
## bmi2          0.034705   0.032996   1.052   0.2929
## diab2         1.713657   0.731953   2.341   0.0192 *
## age2          0.089217   0.022221   4.015 5.94e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 289.60  on 234  degrees of freedom
## Residual deviance: 175.29  on 230  degrees of freedom
##   (20 observations deleted due to missingness)
## AIC: 185.29
##
## Number of Fisher Scoring iterations: 5
```

Predict the probabilities with respect to the response variable

```
model3.probs=predict(model3,test2_df_wth_out,type="response")
model3.probs[1:5]
```

```
##        20        25        27        32        40
## 0.9825080 0.8339924 0.3032803 0.1487336 0.1267438
```

```
# View the first 5 probabilities
model3.pred=rep(0,108)
```

Use the logistic regression model (model3) to assign the test dataset (test2_df_wth_out) observations to one of the two classes

```
# If a predicted probability is less that 0.5 the class is predicted as
0, otherwise 1 (which correspond to non-diabetes, diabetes
respectively)
model3.pred[model3.probs>.5] = 1
# Threshold at 0.5
mean(model3.pred!=new_class2)
```

```
## [1] 0.2777778
```

```
# The total misclassification rate
table(model3.pred,new_class2)
```

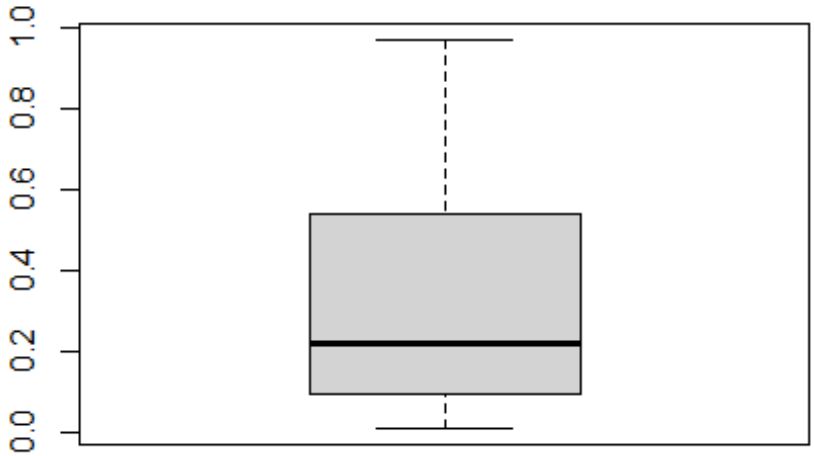
```
##            new_class2
## model3.pred  0  1
##           0 62 19
##           1 11 16
```

```
# We observe that the total misclassification rate is (11+19)/108,
which is equal to 27.7% and the correct classification rate is the
```

```
complementary, in other words it is 72.3%. Now, the FPR is 11/73, which
is equal to 15%, while the FNR is 19/35, which is equal to 54.2%
```

Comments on the logistic regression model

```
summary(m3.probs)

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.008643 0.092304 0.217642 0.335017 0.537736 0.972431

boxplot(m3.probs)
```

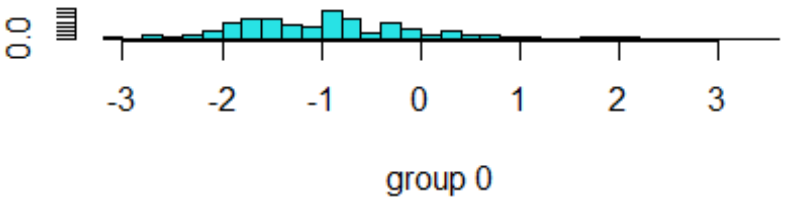



```
# We observe that all values lie in the interval (0,1) which is more
appropriate in our case. Slight improvements observed in comparison
with the linear regression model!
```

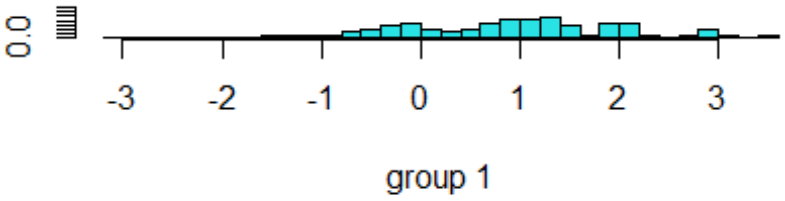
Linear Discriminant Analysis

```
library(MASS)
lda.fit2 <-
lda(n.class2~.,data=df2_without_outliers[,1:4],subset=as.numeric(rownam
es(train2_df_wth_out)) )
lda.fit2

## Call:
## lda(n.class2 ~ ., data = df2_without_outliers[, 1:4], subset =
as.numeric(rownames(train2_df_wth_out)))
##
```

```
## Prior probabilities of groups:
##         0         1
## 0.693617 0.306383
##
## Group means:
##    plasma2     bmi2     diab2     age2
## 0 108.1902 31.74110 0.4422822 27.65644
## 1 148.3056 35.04861 0.5347500 36.69444
##
## Coefficients of linear discriminants:
##                LD1
## plasma2 0.02969336
## bmi2    0.01661757
## diab2   0.87693227
## age2    0.05704112

plot(lda.fit2)
```



group 0



group 1

```
# The plot of scores of train data set on the (one in this case) linear discriminant functions
lda.pred2=predict(lda.fit2, test2_df_wth_out)
names(lda.pred2)

## [1] "class"     "posterior" "x"

lda.class2=lda.pred2$class
table(lda.class2,new_class2)
```

```
##           new_class2
## lda.class2  0  1
##          0 62 19
##          1 11 16
```

```
# We observe that the total misclassification rate is (11+19)/108,
which is equal to 27.7% and the correct classification rate is the
complementary, in other words it is 72.3%. Now, the FPR is 11/73, which
is equal to 15%, while the FNR is 19/35, which is equal to 54.2%
```

```
mean(lda.class2==new_class2)
```

```
## [1] 0.7222222
```

```
# The correct test classification rate (which is the complementary of
the total missclassification rate)
mean(lda.class2!=new_class2)
```

```
## [1] 0.2777778
```

```
# The total misclassification rate
```

Quadratic Discriminant Analysis

```
qda.fit2=qda(n.class2~.,data=df2_without_outliers[,1:4],subset=as.numer
ic(rownames(train2_df_wth_out)) )
qda.fit2
```

```
## Call:
## qda(n.class2 ~ ., data = df2_without_outliers[, 1:4], subset =
as.numeric(rownames(train2_df_wth_out)))
##
## Prior probabilities of groups:
##        0        1
## 0.693617 0.306383
##
## Group means:
##    plasma2     bmi2     diab2     age2
## 0 108.1902 31.74110 0.4422822 27.65644
## 1 148.3056 35.04861 0.5347500 36.69444
```

```
qda.class2=predict(qda.fit2,test2_df_wth_out)$class
table(qda.class2,new_class2)
```

```
##           new_class2
## qda.class2  0  1
##          0 62 17
##          1 11 18
```

```
# We observe that the total misclassification rate is (11+17)/108,
which is equal to 25.9%. The FNR is 17/35, which is equal to 48.5% and
```

```
the FPR is 11/73, which is equal to 15%
mean(qda.class2==new_class2)

## [1] 0.7407407

# The correct test classification rate
mean(qda.class2!=new_class2)

## [1] 0.2592593

# The total classification rate
```

KNN

```
library(class)
set.seed(20)
train.class2=train2_df_wth_out$class
knn2.pred=knn(train2_df_wth_out,test2_df_wth_out,train.class2,k=1)
# Number of neighbours considered k=1
table(knn2.pred,new_class2)

##          new_class2
## knn2.pred  0  1
##         0 56 19
##         1 17 16

# We observe that the total misclassification rate is (17+19)/108,
which is equal to 33.3%. The FNR is 19/35, which is equal to 54.2%
while the FPR is 17/73, which is equal to 23.2%
mean(knn2.pred==new_class2)

## [1] 0.6666667

# The correct classification rate
mean(knn2.pred!=new_class2)

## [1] 0.3333333

# The total misclassification rate
```

Percentage matrix for the misclassification rate and the false negative rate (FNR) for the first dataset (with the imputed values!)

	Total Misclassification Rate	FNR
Linear Regression	22.01835	44.7
Logistic Regression (threshold at 0.5)	20.18349	43.4
Logistic Regression (threshold at 0.1)	41.20000	4.0
LDA	21.55963	43.4
QDA	22.47706	44.7
KNN1	28.44037	40.7
KNN3	23.39450	39.1

Our case refers to medical data, hence the FNR Is of top importance (in fact even more than the total misclassification error). The reason lies in the fact that it represents the women that will in fact develop diabetes while they are classified in the group that won't develop diabetes (positive classified as negative!). Therefore, in our assessment we will choose the method that provides the lowest FNR, logistic regression with a threshold at 0.1 being the case (with a total misclassification rate at 41.2%).

Percentage matrix for the misclassification rate and the false negative rate (FNR) for the second dataset (with the omitted values!)

	Total Misclassification Error	FNR
Linear Regression	19.44444	54.2
Logistic Regression (threshold at 0.5)	18.51852	54.2
LDA	18.51852	54.2
QDA	23.14815	48.5
KNN1	27.77778	54.2

From the above models we would pick the one with the lowest FNR, that is the QDA. However, in the case of the second dataset, due to the fact that we omitted the missing values we were left with approximately half the observations. We notice that the result is much worse! All of the above models have much higher FNRs in comparison with the first (imputed) dataset. Therefore, we would definitely pick the method, in which we do not omit the missing values, rather we impute them!