Biostatistics ⋈ Illustrations in R

Viewing Ecological data using R graphics

A.B. Dufour & N. Pettorelli

April 9, 2009

Presentation of the principal graphics dealing with discrete or continuous variables. Course proposed at the Institut of Zoology, London, May 2009

Contents

1 Introduction

Data viewing is an important first step to data modelling - think for example about detecting outliers and the possible influence of these values on your analysis.
Many R packages contain graphic options but we will here focus on two libraries. The first one `MASS` is included as a bunddle in the bases; the second one (ade4) allows us introducing graphics which are specific to multivariate analysis.

```
library(MASS)
library(ade4)
```

2 Principal graphics for one variable

2.1 Discrete variable

We will here differentiate two situations, according to the type of data - either you are trying to represent data which have been already summarized (e.g., into frequencies); or the data you are considering are organised in a R vector (e.g., vector of binomial values).

2.1.1 Already summarized information

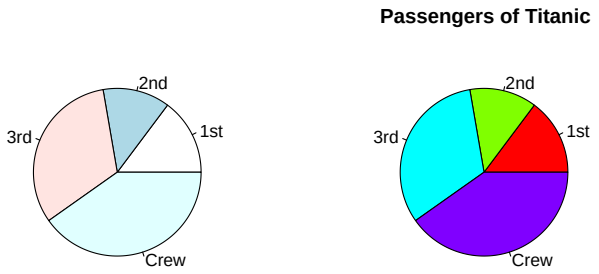
This represents the most common situation.

Example. The dataset 'Titanic' contains the information on the fate of the passengers. We are interested in representing the number of passengers belonging to the different classes who survived the incident.

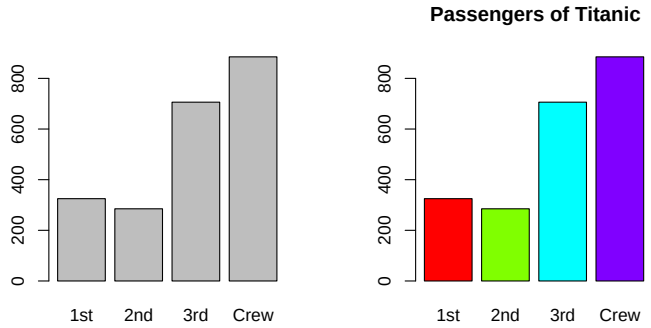
```
library(MASS)
data(Titanic)
titanclas <- apply(Titanic, 1, sum)
```

Based on this information, we can compute two main graphics: the pie chart (`pie`) and the bar plot (`barplot`).

```
par(mfrow = c(1, 2))
pie(titanclas)
pie(titanclas, main = "Passengers of Titanic", col = rainbow(4))
```

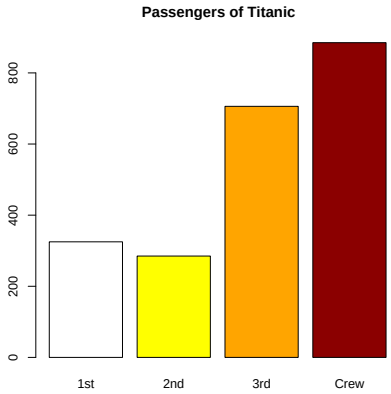


```
par(mfrow = c(1, 2))
barplot(titanclas)
barplot(titanclas, main = "Passengers of Titanic", col = rainbow(4))
```



The most adapted graph is the barplot below since (1) the levels are ordered and (2) the colors follow the class gradient.

```
par(mfrow = c(1, 1))
barplot(titanclas, main = "Passengers of Titanic", col = c("white",
    "yellow", "orange", "darkred"))
```



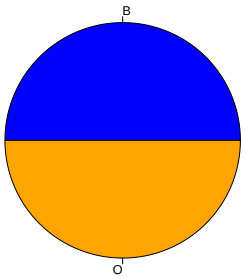
Exercise. Create a graph to represent, using the same data set, the age distribution of the survivors.

2.1.2 Global data

The data are presented as a vector (one column, n individuals). Each element of the vector contains the modality for the considered individual.

Example. The dataset `crabs` contains information on morphological measurements of 200 crabs belonging to two species (orange and blue). Information has been collected on both sex.

```
data(crabs)
varspecies <- summary(crabs$sp)
pie(varspecies, col = c("blue", "orange"))
```



Exercise. Create a graph to represent, using the same data set, the distribution of the measurements according to gender.

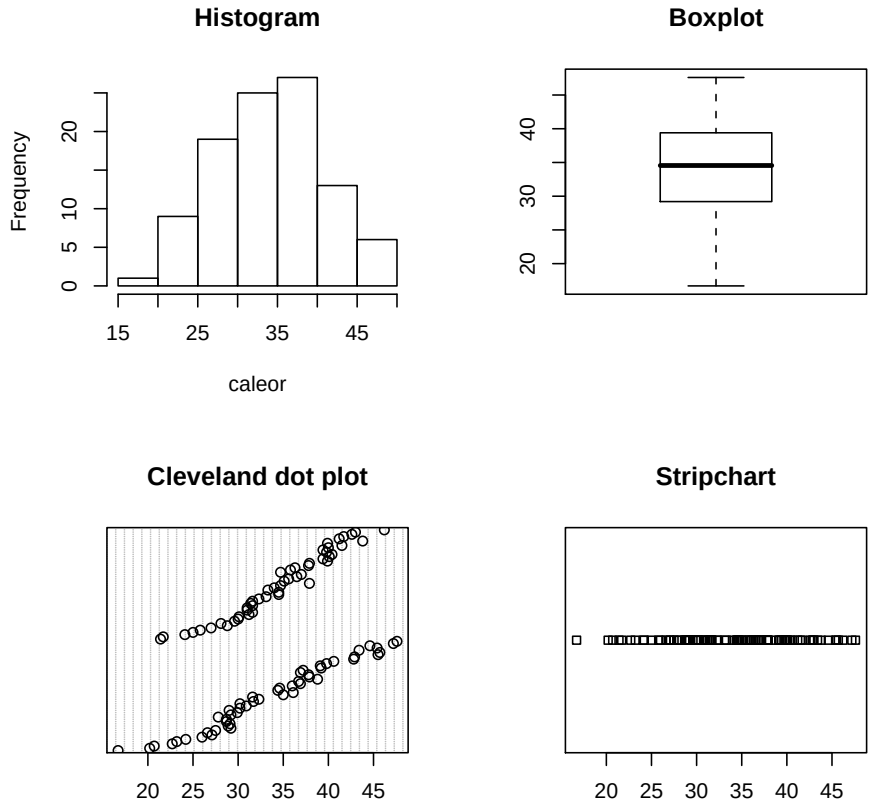
2.2 Continuous variable

We will here focus on:

1. the histogram `hist`
2. the boxplot `boxplot`
3. the Cleveland dot plot `dotchart`
4. a minor one: the one dimension scatter plot `stripchart`

Example. The aim of the following example is to represent the distribution of the carapace length (CL) of the orange crabs.

```
caleor <- crabs$CL[crabs$sp == "O"]
par(mfrow = c(2, 2))
hist(caleor, main = "Histogram")
boxplot(caleor, main = "Boxplot")
dotchart(caleor, main = "Cleveland dot plot")
stripchart(caleor, main = "Stripchart")
```

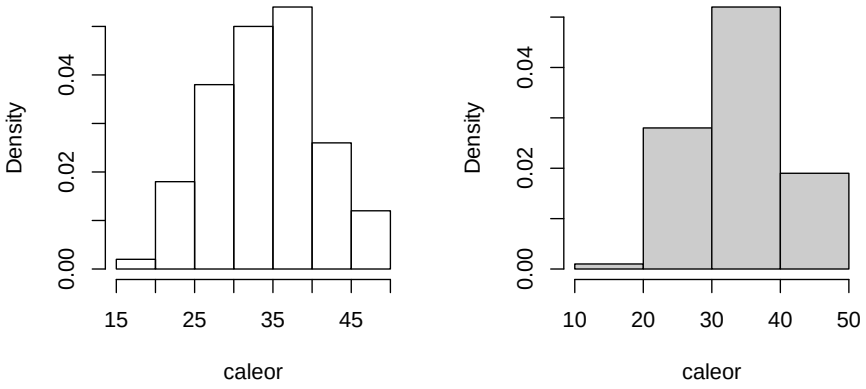


The histogram can be improved using probability distribution (i.e. the superimposition of a theoretical distribution). We can also modified the breaks (see `hist.default` for all the histogram information).

```
par(mfrow = c(1, 2))
hist(caleor, freq = FALSE, main = "")
brex <- seq(10, 50, by = 10)
brex
```

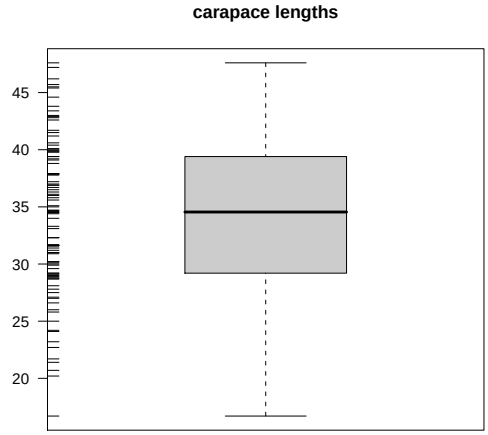
```
[1] 10 20 30 40 50
```

```
hist(caleor, breaks = brex, freq = FALSE, main = "", col = grey(0.8))
```



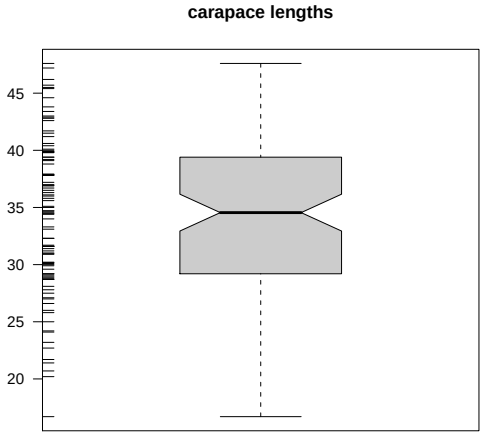
The boxplot is based on the quartiles of the distribution. The function `rug` allows to see each point of the distribution.

```
par(mfrow = c(1, 1))
boxplot(caleor, col = grey(0.8), main = "carapace lengths", las = 1)
rug(caleor, side = 2)
```



It's also possible to add the median confidence interval on the graph (see `boxplot.stats` for more information).

```
boxplot(caleor, notch = TRUE, col = grey(0.8), main = "carapace lengths",
    las = 1)
rug(caleor, side = 2)
```



3 Relationships between two variables

3.1 Two discrete variables

The graphics used to represent the relationships between two discrete variables are:

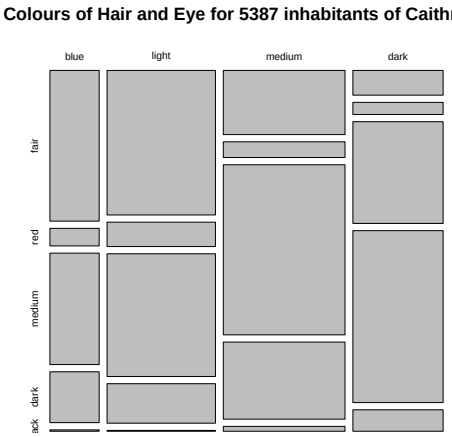
- `mosaicplot()`
- `table.cont()`

Example 1. In Caithness (Scotland), hair and eye colours were determined for 5387 inhabitants.

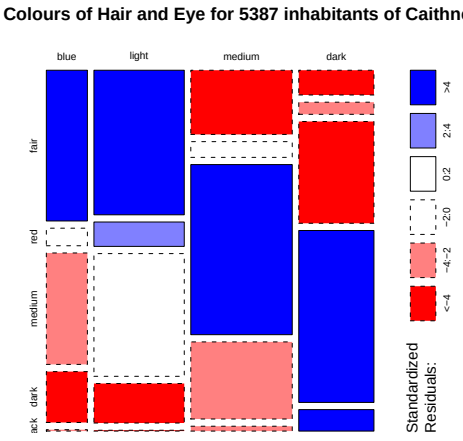
```
data(caith)
caith
```

```
       fair red medium dark black
blue    326  38    241  110     3
light   688 116    584  188     4
medium  343  84    909  412    26
dark     98  48    403  681    85
```

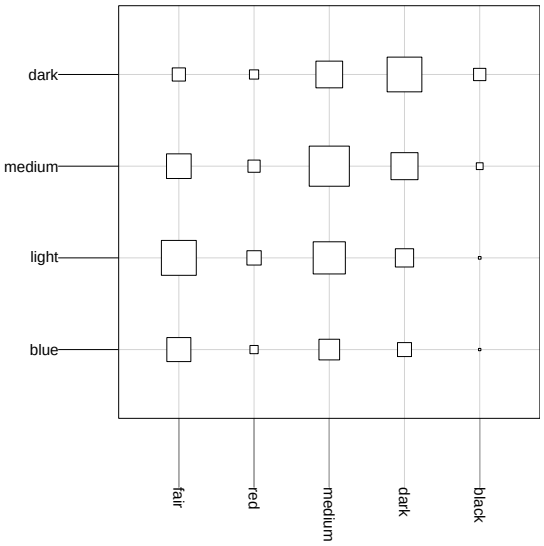
```
mosaicplot(caith, main = "Colours of Hair and Eye for 5387 inhabitants of Caithness")
```




```
mosaicplot(caith, main = "Colours of Hair and Eye for 5387 inhabitants of Caithness",
    shade = T)
```




```
table.cont(caith, csize = 2)
```



Some variables can however have a lot of modalities and the graphics don't help to understand the data structure. In such cases, multivariate data analysis, such as correspondence analysis, is the best way to view the data.

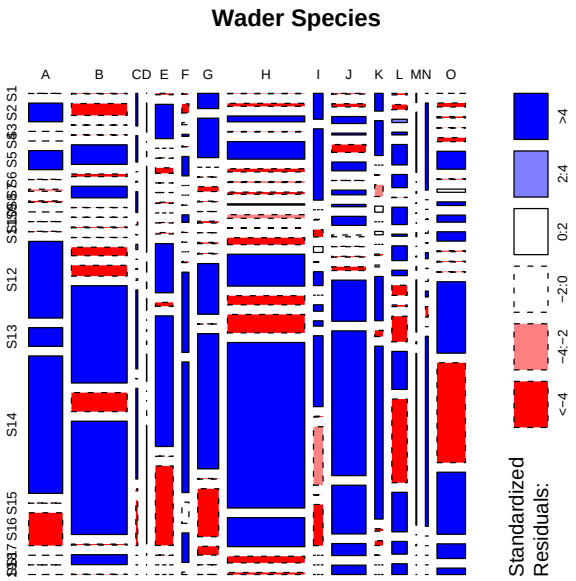
Example 2. The numbers of waders were counted for 19 species at 15 sites in South Africa.

```
data(waders)
waders
```

```
   S1   S2   S3  S4   S5   S6   S7  S8  S9 S10 S11  S12  S13   S14  S15   S16  S17
A  12 2027    0   0 2070   39  219 153   0  15  51 8336 2031 14941   19  3566    0
B  99 2112    9  87 3481  470 2063  28  17 145  31 1515 1917 17321 3378 20164  177
C 197  160    0   4  126   17    1  32   0   2   9  477    1   548   13   273    0
D   0   17    0   3   50    6    4   7   0   1   2   16    0     0    3    69    1
E  77 1948    0  19  310    1    1  64   0  22  81 2792  221  7422   10  4519   12
F  19  203   48  45   20  433    0   0  11 167  12    1    0    26 1790  2916  473
G 1023 2655   0  18  320   49    8 121   9  82  48 3411   14  9101   43  3230  587
H  87  745 1447 125 4330  789  228 529 289 904  34 1710 7869  2247 4558 40880 7166
I 788 2174    0  19  224  178    1 423   0 195 162 2161   25  1784    3  1254    0
J  82  350  760 197  858  962   10 511 251 987 191   34   87   417 4496 15835 5327
K 474  930    0  10  316  161    0  90   0  39  48 1183  166  4626   65   127    4
L  77  249  160 136  999  645   15 851 101 723 266  495   83  1253 1864  4107 1939
M  22  144    0   4    1    1    0  10   0   2   9  125    5   411    0     3    0
N   0  791    0   0    4   38    1  56   1  30  54   95    0  1726    0     0    0
O   0  360  128  43  364 1628   63 287 328 641 850   83   67    48 6499  9094 5647
   S18  S19
A    5    0
B 1759   53
C    0    0
D    0    0
E    0    0
F  658   55
G   10    5
H 1632  498
I    0    0
J 1312 1020
K    0    0
L  623  527
M    0    0
N    0    0
O 1333  582
```

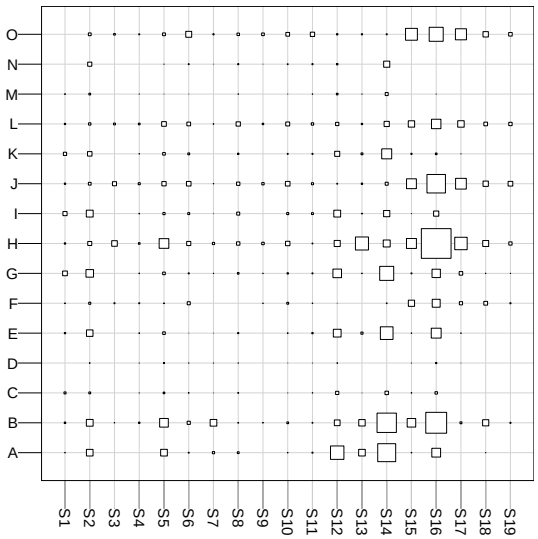
Understanding the structure of such a data set by just looking at the contingency table is clearly impossible. If we try to graphically represent such data set, we obtain:

```
mosaicplot(waders, main = "Wader Species", shade = T)
```



Alternatively, one can try `table.cont`.

```
table.cont(waders)
```

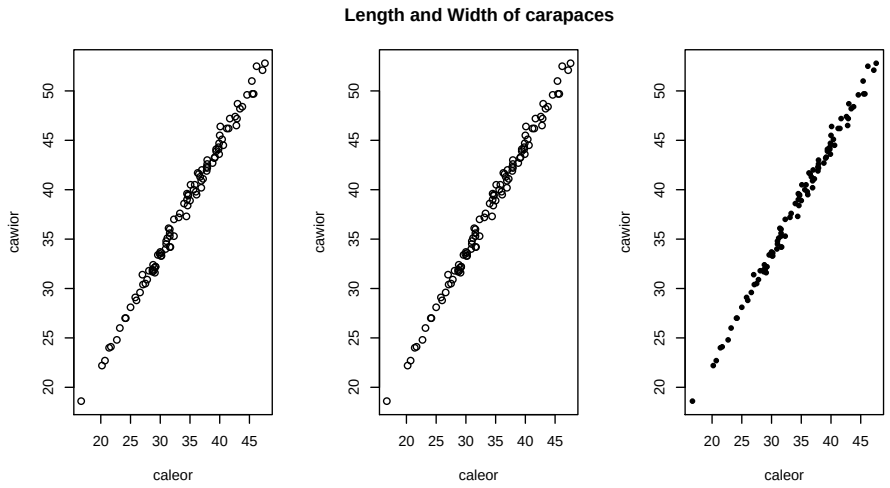


3.2 Two continuous variables

A typical way of representing the relationship between two continuous variables is the scatter plot.

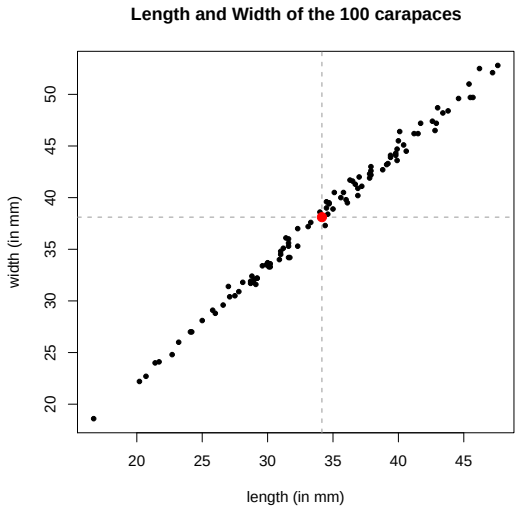
Example 1. Let's study, for instance, the relationship between the length (in mm) and the width (in mm) of the carapaces, focusing here on the orange crab species.

```
cawior <- crabs$CW[crabs$sp == "O"]
par(mfrow = c(1, 3))
plot(caleor, cawior)
plot(cawior ~ caleor, main = "Length and Width of carapaces")
plot(cawior ~ caleor, pch = 20)
```



Notice that if we want to add some information on the graph, we first need to compute a `plot()`.

```
par(mfrow = c(1, 1))
plot(cawior ~ caleor, main = "Length and Width of the 100 carapaces",
    pch = 20, xlab = "length (in mm)", ylab = "width (in mm)")
abline(h = mean(cawior), lty = 2, col = grey(0.6))
abline(v = mean(caleor), lty = 2, col = grey(0.6))
points(mean(caleor), mean(cawior), col = "red", pch = 20, cex = 2)
```

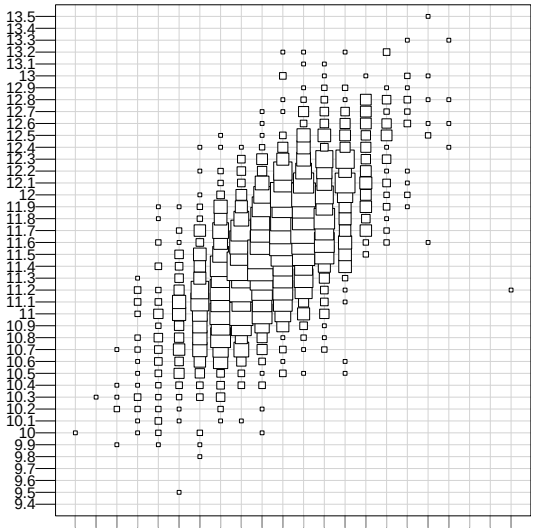


Exercise. Choose two other continuous variables in this data set and draw the best graph.

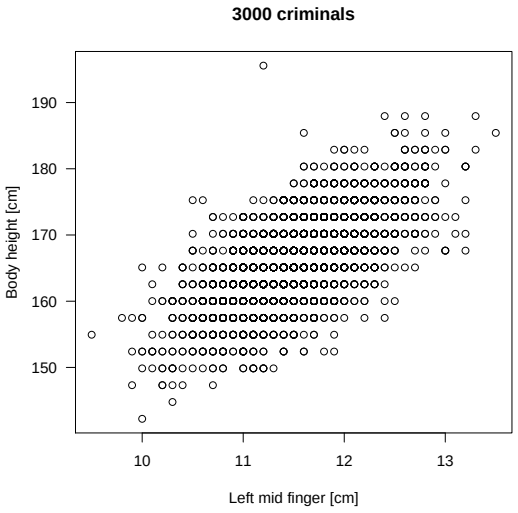
Sometimes, however, several individuals can get the same values (especially in the case of fixed values). In the classical plot, these individuals will only be represented by one single point. So if we want to see the superimposition of points, we need to use specific arguments in the previous graph or built a new one such as the `sunflowerplot`.

Example 2. Let's examine here the famous data set used by W.S. Gosset (aka Student) to develop the t-test: the body heights and the mid-finger lengths of 3000 criminals.

```
data(crimtab)
table.cont(crimtab)
```

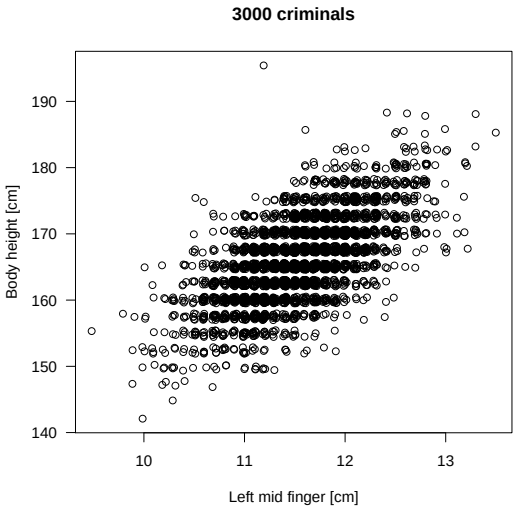


```
crimtab.dft <- as.data.frame(crimtab)
expand.dft <- function(x, na.strings = "NA", as.is = FALSE, dec = ".") {
    DF <- sapply(1:nrow(x), function(i) x[rep(i, each = x$Freq[i]),
        ], simplify = FALSE)
    DF <- subset(do.call("rbind", DF), select = -Freq)
    for (i in 1:ncol(DF)) {
        DF[[i]] <- type.convert(as.character(DF[[i]]), na.strings = na.strings,
            as.is = as.is, dec = dec)
    }
    DF
}
crimtab.raw <- expand.dft(crimtab.dft)
x <- crimtab.raw[, 1]
y <- crimtab.raw[, 2]
plot(x, y, las = 1, main = "3000 criminals", ylab = "Body height [cm]",
    xlab = "Left mid finger [cm]")
```



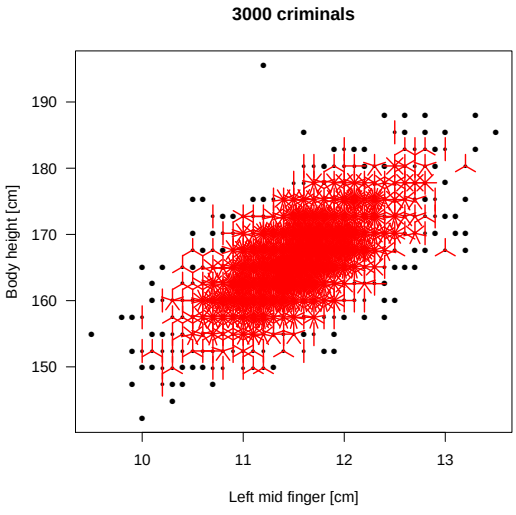
If we want to see all the points, one possible solution is to jitter the data.

```
plot(jitter(x), jitter(y), las = 1, main = "3000 criminals", ylab = "Body height [cm]",
    xlab = "Left mid finger [cm]")
```



Another option is the sunflowerplot. The number of repetitions is represented by the number of petals around a point.

```
sunflowerplot(x, y, las = 1, main = "3000 criminals", ylab = "Body height [cm]",
    xlab = "Left mid finger [cm]")
```



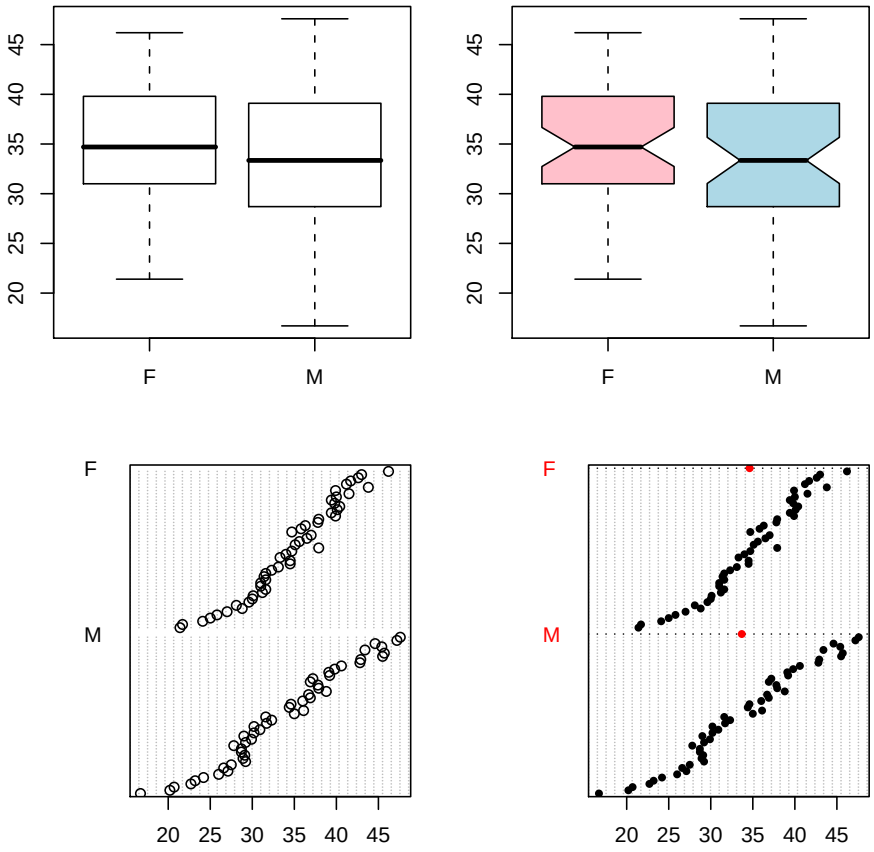
Exercise. Is there any visible repetition when exploring the relationship between length and width of carapaces (focusing again on orange crabs)?

3.3 One discrete and one continuous variables

The main displays to link discrete and continuous variables are the `boxplot` or the `dotchart` graphics.

Example. Let's explore the relationship between carapace length and sex in orange crabs.

```
crabsex <- crabs$sex[crabs$sp == "O"]
par(mfrow = c(2, 2))
par(mar = c(3, 2, 2, 2))
boxplot(caleor ~ crabsex)
boxplot(caleor ~ crabsex, col = c("pink", "lightblue"), notch = TRUE)
dotchart(caleor, groups = crabsex)
dotchart(caleor, groups = crabsex, gdata = tapply(caleor, crabsex,
    mean), pch = 20, gpch = 20, gcol = "red")
par(mfrow = c(1, 1))
```



Exercise. The data contains 4 other continuous variables. Produce a representation, in the same graphical device, of the relationship between each variable and sex.

4 Adding an information: two continuous variables and a factor

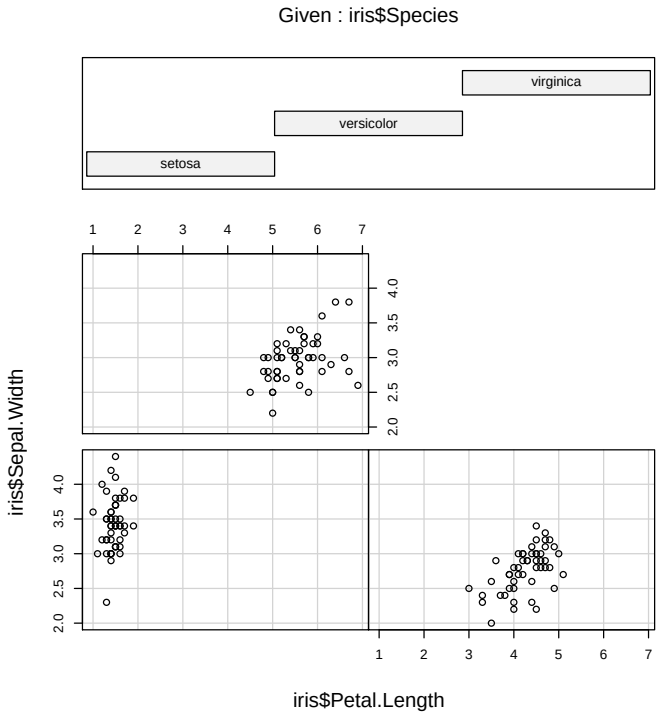
We are here interested in computing three variables together (e.g., relationship between sepal length and width of the famous Iris dataset, for the three species

- *Iris setosa*, *Iris versicolor* and *Iris virginica*). Two main representations can be used: `coplot` and `s.class` of the `ade4` package.

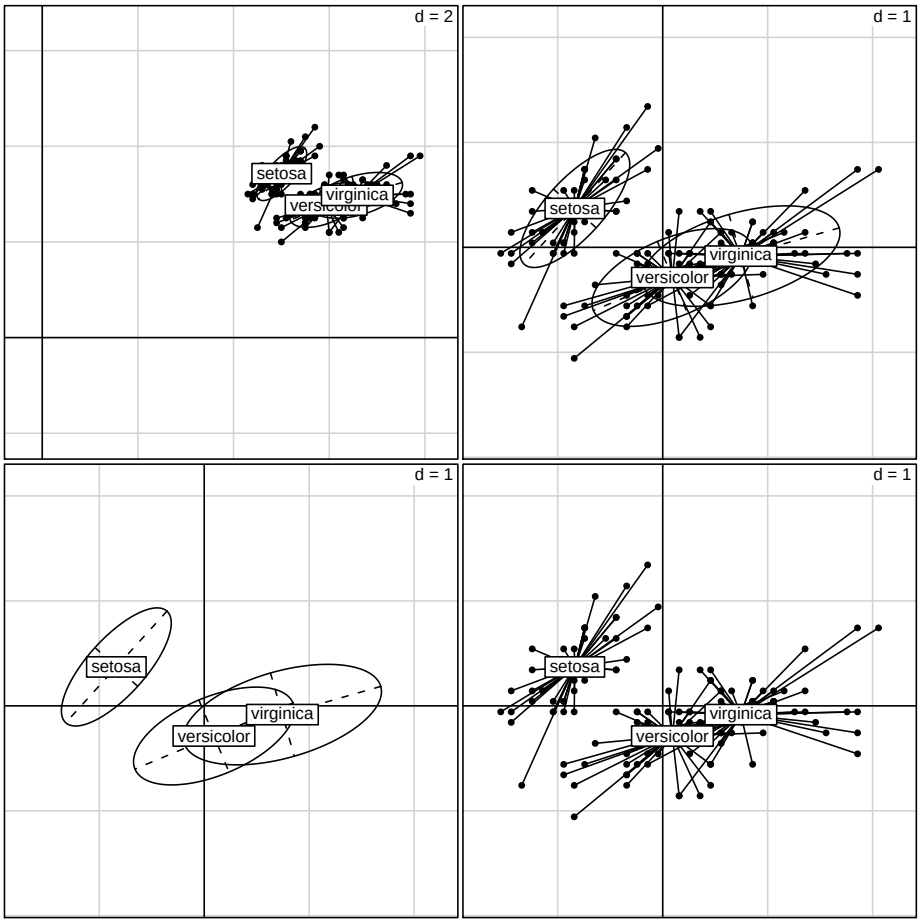
```
data(iris)
names(iris)
```

```
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

```
coplot(iris$Sepal.Width ~ iris$Petal.Length | iris$Species)
```




```
par(mfrow = c(2, 2))
s.class(iris[, 1:2], iris$Species)
s.class(scale(iris[, 1:2], center = T, scale = F), iris$Species)
s.class(scale(iris[, 1:2], center = T, scale = F), iris$Species,
    cstar = 0, cpoint = 0)
s.class(scale(iris[, 1:2], center = T, scale = F), iris$Species,
    cellipse = 0)
```



Exercise. The data set contains 3 species and 4 measurements. Produce a representation of the relationship between two measurements and species.

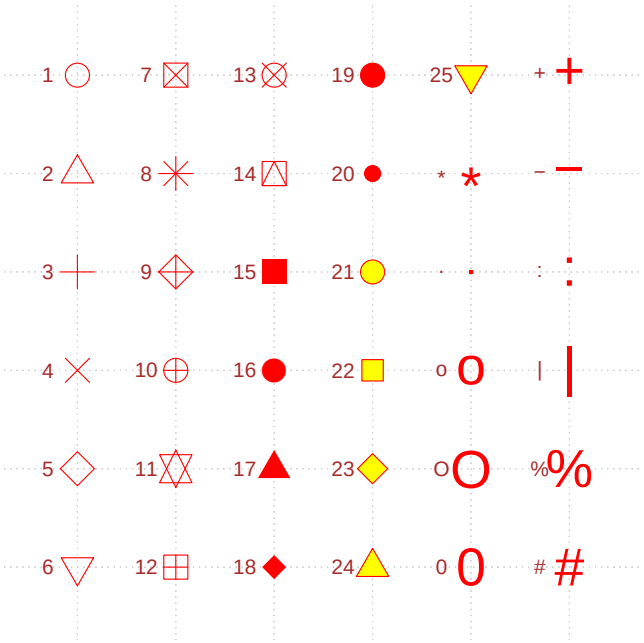
5 Conclusion

Data viewing is a crucial first step to data modelling. Remembering functions can be challenging.
R provides some help such as:

1. `help.start()` opens a window with an interface for HTML help.
2. `RSiteSearch("hist")` looks for information on the R website (manuals, documentation, archives,...)
3. `help.search("hist")` opens a page to inform where to find good explanations
4. `help("hist")` opens a page giving all the information about the research object.

6 Appendice: Types of point and line

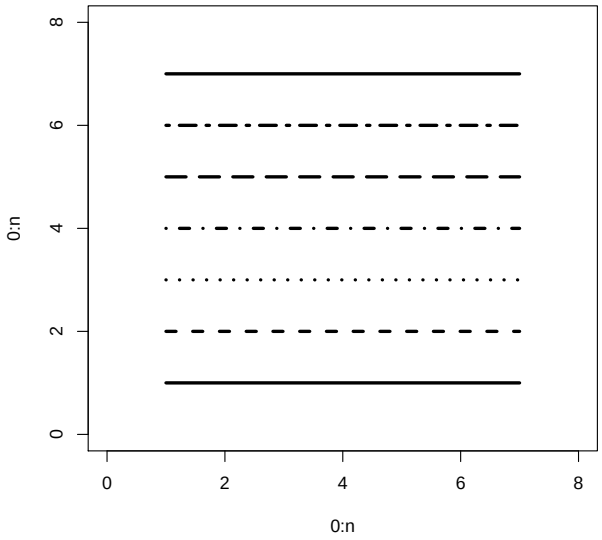
6.1 Point Types



1. The point type is given by the `pch` argument. For instance, empty circles (by default) is `pch = 1` (see more information on the appendix of the document).
2. The colour of a point is given by the `col` argument. For instance, we used the red to compute the previous graphic using `col = "red"`.
3. The inside colour of a point is given by the `bg` argument. For instance, we used the yellow to compute the previous graphic using `bg = "yellow"`.
4. For more details, look at the `points()` function.

6.2 Line Types

```
n <- 7
plot(0:n, 0:n, type = "n", xlim = c(0, n + 1), ylim = c(0, n + 1))
for (i in 1:n) {
    lines(1:n, rep(i, n), lwd = 3, lty = i)
}
```



Line types are given by the `lty` argument. For instance, the continuous line type (by default) is `lty = 1` (see more information on the appendix of the document).

You may find useful to read the *Graphical procedures* chapter from *An Introduction to R* (following the menu `Help`, `Manuals`).