

# 3 Graphics

## 3.1 Overview

- a Several R graphics functions have been presented so far:

```
> plot(d.sport[,"kugel"], d.sport[,"speer"],
+ xlab="ball push", ylab="javelin", pch=7)
> plot(punkte ~ kugel+speer, data=d.sport)
> pairs(d.sport)
> boxplot(sleep[1:10,"extra"],sleep[11:20,"extra"],ylab="extra")
```

Many more functions are available and will be introduced:

```
scatter.smooth(), matplot(), image(), ...
lines(), points(), ...
par(), identify(), dev.new(), ...
```

- b There are 6 kinds of **R graphics functions**:

- **High-level plotting functions** such as `plot()`  
⇒ *generate a new graphical display of data.*
- **Low-level plotting functions** such as `lines()`  
⇒ *add further graphical elements to an existing graph.*
- **“Interactive” functions** such as `identify()`  
⇒ *enhance or collect information interactively from a graph.*
- **“Control” functions** such as `par()`  
⇒ *control the appearance of graphs.*
- **“Device” control functions** such as `dev.new()`  
⇒ *to manipulate windows and files that display or store graphs.*

## 3.2 Scatterplot

- a The most common graphical display shows the values of two variables, plotted against each other. The (first) syntax is

```
> plot(x=x, y=y, main="...", xlab="...",
ylab="...", ...)
```

$x, y$  are two numeric vectors.

Example:

```
> plot(x=meuse[,"x"], y=meuse[,"y"], xlab="easting", ylab="northing",
+ main="sampling locations")
```

b Three alternative ways to invoke `plot()`:

- Plot of the values of a single vector against the position indices of the vector elements:  
`> plot(meuse[, "zinc"], ylab = "zinc")`
- Scatterplot of two columns of a matrix or a dataframe  
`> plot(meuse[, c("x", "y")])`
- Use of a model “formula” to select  $y$  and  $x$  variable from a data frame:  
`> plot(zinc~dist, data=meuse)`  
 Note:  $y\sim x$ : The variable to be used on the vertical axis comes first.

c Many **arguments** of `plot()` are **common** to many graphics functions:

- `main="...", xlab="...", ylab="..."`, where  
`...`: any character string  
 $\Rightarrow$  used to set **title** and **labels** of axes
- `log="x", log="y", log="xy"`  
 $\Rightarrow$  for **logarithmic scaling** of axes
- `xlim=c( $x_{\min}$ ,  $x_{\max}$ ), ylim=c( $y_{\min}$ ,  $y_{\max}$ )`  
 $\Rightarrow$  set **ranges** for the values to be displayed.
- `asp= $n$`   
 $\Rightarrow$  set “aspect ratio” of axes, i.e. ratio of lengths of “measurement” units on  $y$  and  $x$  axis. This is most often used to ensure that the scales of both axis are equal when displaying a geographical location or two geometrical measurements of an object, by setting `asp = 1`.
- `pch= $i$  or pch="c"`  
 $\Rightarrow$  select the **plotting symbol(s)**  
`c`: a (vector of) single character(s) or  
`i`: a (vector of) integer(s) to select one of the built-in geometrical symbols (square, circle, triangle, cross, ..., see `?points` for a list).  
 If a vector is given, different symbols will be used for the different points (recycled if necessary).
- `cex= $n$`   
 $\Rightarrow$  choose the **size** of the symbols, relative to the standard size that the `plot` function would choose by default. A vector can be given as with `pch`.
- `col= $i$  or col="color"`  
 $\Rightarrow$  choose the **size** and **color** of symbols  
`color`: name of color in english (`col = "red"`). A vector can again be given as with `pch`.

### 3.3 Boxplot

- a Boxplots show the most important aspects of (the distribution of) a sample in a simple way. **Syntax:**

```
> boxplot(x=x, notch=l, horizontal=l, ...)
```

`notch = TRUE`: “notches” are added to roughly test whether two medians are significantly different

`horizontal = TRUE`: boxplots are shown horizontally. (There is no argument `vertical`.)

**Example:**

```
> boxplot(x=meuse[, "zinc"], notch=TRUE, horizontal=TRUE, log="x",
+ xlab="zinc content")
```

- b Variations of boxplots:

- Boxplot of several variables in same graph:
 

```
> boxplot(meuse[,c("zinc", "lead")], horizontal=TRUE, log="x")
```
- Boxplots of several groups for one variable:
 

```
> boxplot(zinc~ffreq, data=meuse)
```

### 3.4 Adding Points and Lines to a Plot

- a Use `points()` to add further **points** to a graph created before by a high-level graphics function.

```
> points(x=x, y=y, pch=i, col= , cex=n, ...)
```

Example:

```
> plot(lead~dist.m, data=meuse, ylim=c(10,1000), log="y")
> points(meuse[,c("dist.m", "copper")],
+ col="red", pch=3)
```

- b **Lines** can be added by `lines()` to a graph.

```
> lines(x=x, y=y, col= , lty= , lwd=n, ...)
```

`lty=i` or `lty="linetype"` (keyword): type of line

`n`: line width relative to default value

Example:

```
> plot(y~x, meuse, asp=1)
> lines(meuse.riv, lty="dotted", lwd=2.5, col="cyan")
```

- c **Straight lines** through the whole plotting area can be added by `abline()` to a graph.

```
> abline(a=n, b=n, h=v, v=v, ...)
```

Provide either values to `a` (intercept) and `b` (slope) **or** `h=v` as the (y) position(s) of *horizontal* **or** `v=v` as the (x) position(s) of *vertical* straight line(s).

Example:

```
> plot(lead~dist.m, meuse, asp=1)
> abline(h=c(200,500), lty="dotted", col=c("orange", "red"))
> abline(a=300, b=-1, lty="3313", lwd=3)
```

- d **Line segments** are added by `segments()`.

```
> segments(x0=v, y0=v, x1=v, x2=v, ...)
```

Line segments are drawn *from the points*  $(x_0, y_0)$  *to the points*  $(x_1, y_1)$  ( $v$  numeric vectors of same length).

Example:

```
> plot(y~x, meuse)
> t.xr <- range(meuse[, "x"])
> t.yr <- range(meuse[, "y"])
> segments(x0=rep(t.xr[1], 2), y0=t.yr, x1=rep(t.xr[2], 2), y1=t.yr[2:1] )
```

- e **Polygons** are added by `polygon()`.

```
> polygon(x=x, y=y, density=n, angle=n, border= , col= , ...)
```

Provide values to `density` and `angle` for **hachuring** and to `border` and `col` for **border** and **fill color** of polygons. ???

Example:

```
> plot(y~x, meuse, asp=1)
> polygon(meuse.riv, border="blue", col="cyan", angle=135, density=10)
```

### 3.5 Adding Text to a Plot

- a **Points** in a scatterplot are **labelled** by `text()`.

```
> text(x=x, y=y, labels= , pos=i, ...)
```

Provide a vector of character strings to `labels`. The argument `pos` controls whether the text is plotted below (1), to the left (2), above (3) or to the right (4) of the points.

Example:

```
> plot(y~x, meuse, asp=1, type="n")
> text(meuse[, c("x", "y")], cex=0.7, labels=meuse[, "landuse"])
```

- b **Legend.** Place a legend explaining any different symbols or line types used by `legend()`.

```
> legend(x= , y=n, xjust=n, yjust=n,
  legend= , col= , lty= , pch= , ...)
```

The position of the legend is either specified by `x` and `y` in combination with the arguments `xjust`, `yjust` or by a keyword such as `"bottomleft"` as the first argument (cf. `?legend` for details). `legend` contains the explanatory text as a vector of character strings. One or more of the remaining arguments will be specified by a vector of the same length as `legend` and will give the symbols and line types (`pch` and `lty`), the symbol sizes (`cex`), or the colors (`col`).

Example:

```
> plot(meuse[, c("x", "y")], asp=1, col=as.numeric(meuse[, "ffreq"]),
+ cex=sqrt(meuse[, "zinc"])/10)
> legend(x="topleft", pch=c(NA, rep(1, 3)),
+ col=c("black", "black", "red", "green"),
+ legend=c("flooding", "often", "intermediate", "rarely"))
```

## 3.6 Interacting with a Plot

- a **Points** in plots are identified and queried interactively by `identify()`.

```
> identify(x=x, y=y, labels= )
```

Provide a character or numeric vector of the same length as `x` and `y` to `labels` for labelling identified points accordingly. Click the right hand mouse button to stop identifying.

Example:

```
> plot(meuse[,c("x","y")], asp=1, cex=sqrt(meuse[,"zinc"]/10))
> t.i <- identify(meuse[,c("x","y")], labels=meuse[,"zinc"])
```

Now, the sequence numbers of the identified points are available as `t.i` for further use.

- b Read the coordinates of interactively selected points from plots by using `locator()`.

```
> locator(n=i, type="p")
```

You can either specify the number `i` of points to locate in advance or right-click to stop locating points.

Example:

```
> plot(meuse[,c("x","y")], asp=1)
> polygon(locator())
```