

Using R for Data Analysis and Graphics

4. Graphics

4.1 Overview

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(t.y1,t.y2,ylab="extra")
```

Many more functions to come:

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

4.1 Overview

Six kinds of R graphics functions:

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

4.2 Scatterplot

Display of the values of two variables plotted against each other.

Syntax:

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

x, y : two numeric vectors

Example:

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

4.2 Scatterplot

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")], asp=1)
```
- Use of a formula to select y- and x-variable out of a data frame

```
> plot(zinc~dist, data=meuse)
```

4.2 Scatterplot

Arguments of `plot()` **common** to many graphics functions:

- `main="..."`, `xlab="..."`, `ylab="..."`
... : any character string
⇒ to set **title** and **labels** of axes
- `log="x"`, `log="y"`, `log="xy"`
⇒ for **logarithmic scaling** of axes
- `xlim=c(xmin, xmax)`, `ylim=c(ymin, ymax)`
`xmin`, `xmax`, `ymin`, `ymax` : numeric scalars
⇒ to set **ranges** displayed on axes

4.2 Scatterplot

Common arguments of `plot()` (continued):

- `asp=n`
`n` : numeric scalar
⇒ to set **range** aspect ratio of axes
- `pch=i` or `pch="c"`
`i` : an integer; `c` : a single character
⇒ for selecting **symbol type**
- `cex=n`
⇒ for choosing **size** and **color** of symbols
- `col=i` or `col="color"`
`color` : key word
⇒ for choosing **size** and **color** of symbols

4.3 Boxplot

Syntax:

```
boxplot(x=x, notch=1, horizontal=1, ...)
```

1 (logical): controls whether

- “notches” are added to roughly test whether two medians are significantly different
- whether boxplots are horizontal

Example:

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

4.3 Boxplot

Variants to invoke `boxplot()` :

- 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)
```

4.4 Adding points and lines to a graphic

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

Syntax:

```
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)
```

4.4 Adding points and lines to a graphic

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

Syntax:

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

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

`n` : (relative) line width

Example:

```
> plot(y~x, meuse, asp=1)
```

```
> lines(meuse.riv, lty="dotted", lwd=2.5,  
+ col= "cyan")
```

4.4 Adding points and lines to a graphic

Straight lines can be added by `abline()` to a graph.

Syntax:

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

Provide either values to `a` (intercept) and `b` (slope) **or** `h=v` as the position(s) of *horizontal* **or** `v=v` as the 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)
```

4.4 Adding points and lines to a graphic

Straight line segments are added by `segments()`.

Syntax:

```
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] )
```

4.4 Adding points and lines to a graphic

Polygons are added by `polygon()` .

Syntax:

```
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)
```

4.5 Adding text to a graphic

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

Syntax:

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

Provide a vector with 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"])
```

4.5 Adding text to a graphic

More sophisticated **text annotation** is added by `legend()` .

Syntax:

```
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 keywords such as "bottomleft" (cf. `?legend` for details). For the remaining arguments provide vectors of the same length with text (`legend`), symbol and line (`pch` , `lty`) types and color (`col`).

4.5 Adding text to a graphic

Legend continued.

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"))
```

4.6 Interacting with a graphic

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

Syntax:

```
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. Right-click to stop identifying.

Example:

```
> plot(meuse[,c("x", "y")], asp=1,  
+ cex=sqrt(meuse[, "zinc"])/10)  
> identify(meuse[,c("x", "y")],  
+ labels=meuse[, "zinc"])
```

4.6 Interacting with a graphic

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

Syntax:

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

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())
```