R Package diagram: visualising simple graphs, flowcharts, and webs

Karline Soetaert
Royal Netherlands Institute of Sea Research (NIOZ)
Yerseke, The Netherlands

Abstract

This document describes how to use the **diagram** package (Soetaert 2009a) for plotting small networks, flow charts, and (food) webs.

Together with R-package **shape** (Soetaert 2009b) this package has been written to produce the figures of the book (Soetaert and Herman 2009b).

The electrical network symbols were added to produce a figure of the book (Soetaert, Cash, and Mazzia 2012)

Keywords: diagram, food web, flow chart, arrows, R.

1. Introduction

There are three ways in which package **diagram** can be used:

- function `plotmat` takes as input a matrix with transition coefficients or interaction strengths. It plots the corresponding network consisting of (labeled) boxes (the components) connected by arrows. Each arrow is labeled with the value of the coefficients.
- function `plotweb` takes as input a matrix with (flow) values, and plots a web. Here the components are connected by arrows whose thickness is determined by the value of the coefficients.
- Flowcharts can be made by adding separate objects (textboxes) to the figure and connecting these with arrows.

Three datasets have been included:

- `Rigaweb`, the planktonic food web of the Gulf of Riga (Donali, Olli, Heiskanen, and Andersen 1999).
- `Takapotoweb`, the Takapoto atoll planktonic food web (Niquil, Jackson, Legendre, and Delesalle 1998).
- `Teasel`, the transition matrix describing the population dynamics of Teasel, a European perennial weed ((Caswell 2001; Soetaert and Herman 2009b).

The food webs were generated using R packages **LIM** and **limSolve** (Soetaert, Van den Meersche, and van Oevelen 2009; Soetaert and van Oevelen 2009) which contain functions to read and solve food web problems respectively.

2. plotmat - plotting networks based on matrix input

This is the quickest method of plotting a network. The network is specified in a matrix, which gives the magnitudes of the links (from columns to rows).

The position of the elements (boxes) is specified by argument `pos`. Thus, setting `pos=c(1,2,1)` indicates that the 4 elements will be arranged in three equidistant rows; on the first row one element, on the second row two elements and on the third row one element.

2.1. Simple examples

Below are some simple examples of the use of `plotmat`. In the first graph - four simple boxes are put; no arrows drawn

The second graph contains round boxes with arrows, labeled "flow"

The third graph has diamond-shaped boxes including self-arrows.

The fourth graph has hexagonal-shaped boxes, with curved arrows. The arrows are enlarged and the arrowhead pointing from box 2 to 4 is colored red.

```
> par(mar = c(1, 1, 1, 1), mfrow = c(2, 2))
> #
> #
> names <- c("A", "B", "C", "D")
> M <- matrix(nrow = 4, ncol = 4, byrow = TRUE, data = 0)
> plotmat(M, pos = c(1, 2, 1), name = names, lwd = 1,
+         box.lwd = 2, cex.txt = 0.8, box.size = 0.1,
+         box.type = "square", box.prop = 0.5)
> #
> M[2, 1] <- M[3, 1] <- M[4, 2] <- M[4, 3] <- "flow"
> plotmat(M, pos = c(1, 2, 1), curve = 0, name = names, lwd = 1,
+         box.lwd = 2, cex.txt = 0.8, box.type = "circle", box.prop = 1.0)
> #
> #
> diag(M) <- "self"
> plotmat(M, pos = c(2, 2), curve = 0, name = names, lwd = 1, box.lwd = 2,
+         cex.txt = 0.8, self.cex = 0.5, self.shiftx = c(-0.1, 0.1, -0.1, 0.1),
+         box.type = "diamond", box.prop = 0.5)
> M <- matrix(nrow = 4, ncol = 4, data = 0)
> M[2, 1] <- 1 ; M[4, 2] <- 2 ; M[3, 4] <- 3 ; M[1, 3] <- 4
> Col <- M
> Col[] <- "black"
> Col[4, 2] <- "darkred"
> pp <- plotmat(M, pos = c(1, 2, 1), curve = 0.2, name = names, lwd = 1,
+               box.lwd = 2, cex.txt = 0.8, arr.type = "triangle",
```

```
+               box.size = 0.1, box.type = "hexa", box.prop = 0.25,
+               arr.col = Col, arr.len = 1)
> mtext(outer = TRUE, side = 3, line = -1.5, cex = 1.5, "plotmat")
> #
> par(mfrow = c(1, 1))
```

The contents of pp shows the position of the various items.

```
> pp
```

```
$arr
  row col    Angle Value        rad    ArrowX    ArrowY     TextX     TextY
1   2   1  53.1301     1 0.08333333 0.3085298 0.7169284 0.2843333 0.7346667
2   4   2 -53.1301     2 0.08333333 0.3077450 0.2841193 0.2843333 0.2653333
3   1   3 -53.1301     4 0.08333333 0.6918629 0.7164048 0.7156667 0.7346667
4   3   4  53.1301     3 0.08333333 0.6910769 0.2825485 0.7156667 0.2653333

$comp
        x         y
[1,] 0.50 0.8333333
[2,] 0.25 0.5000000
[3,] 0.75 0.5000000
[4,] 0.50 0.1666667

$radii
       x     y
[1,] 0.1 0.025
[2,] 0.1 0.025
[3,] 0.1 0.025
[4,] 0.1 0.025

$rect
     xleft      ybot xright      ytop
[1,]  0.40 0.8083333   0.60 0.8583333
[2,]  0.15 0.4750000   0.35 0.5250000
[3,]  0.65 0.4750000   0.85 0.5250000
[4,]  0.40 0.1416667   0.60 0.1916667
```

2.2. a schematic representation of an ecosystem model

In the example below, first the main components and arrows are drawn (`plotmat`), and the output of this function written in list pp. This contains, a.o. the positions of the components (boxes), arrows, etc.. It is used to draw an arrow from the middle of the arrow connecting fish and zooplankton ("ZOO") to detritus. Function `straightarrow` (see below) is used to draw this arrow.

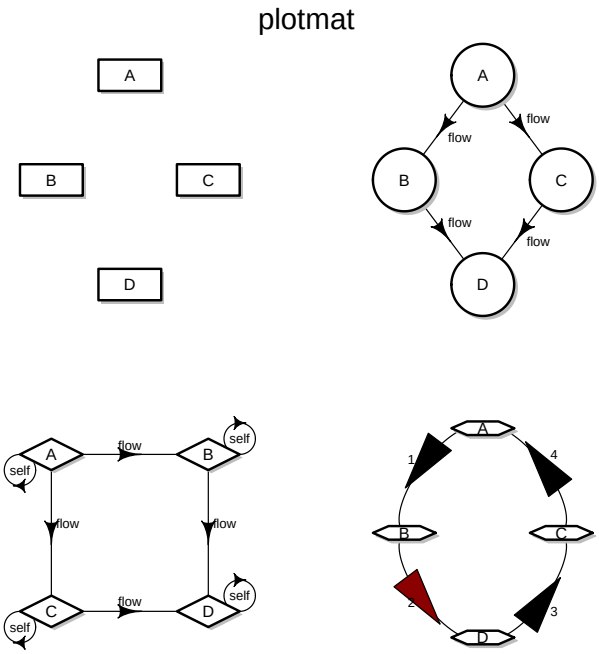

Figure 1: Four simple examples of `plotmat`

```
> names <- c("PHYTO", "NH3", "ZOO", "DETRITUS", "BotDET", "FISH")
> M <- matrix(nrow = 6, ncol = 6, byrow = TRUE, data = c(
+ #   p  n  z   d   b   f
+     0, 1, 0,  0,  0,  0, #p
+     0, 0, 4, 10, 11,  0, #n
+     2, 0, 0,  0,  0,  0, #z
+     8, 0, 13, 0,  0, 12, #d
+     9, 0, 0,  7,  0,  0, #b
+     0, 0, 5,  0,  0,  0  #f
+     ))
> #
> pp <- plotmat(M, pos = c(1, 2, 1, 2), curve = 0, name = names,
+               lwd = 1, box.lwd = 2, cex.txt = 0.8,
+               box.type = "square", box.prop = 0.5, arr.type = "triangle",
+               arr.pos = 0.4, shadow.size = 0.01, prefix = "f",
+               main = "NPZZDD model")
> #
> phyto    <- pp$comp[names=="PHYTO"]
> zoo      <- pp$comp[names=="ZOO"]
> nh3      <- pp$comp[names=="NH3"]
> detritus <- pp$comp[names=="DETRITUS"]
> fish     <- pp$comp[names=="FISH"]
> #
```

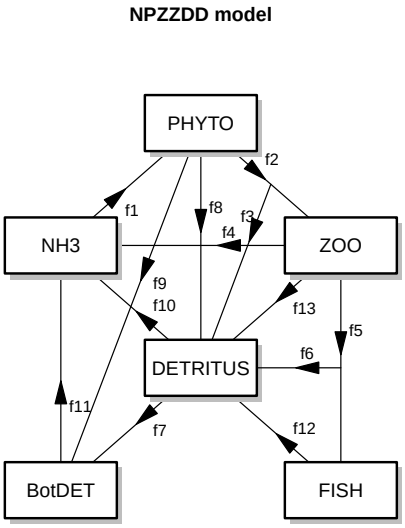

Figure 2: An NPZZDD model

```
> # flow5->detritus
> #
> m2 <- 0.5*(zoo+fish)
> m1 <- detritus
> m1[1] <- m1[1] + pp$radii[4,1]
> mid <- straightarrow (to = m1, from = m2, arr.type = "triangle",
+                       arr.pos = 0.4, lwd = 1)
> text(mid[1], mid[2]+0.03, "f6", cex = 0.8)
> #
> # flow2->detritus
> #
> m2 <- 0.5*(zoo+phyto)
> m1 <- detritus
> m1[1] <-m1[1] + pp$radii[3,1]*0.2
> m1[2] <-m1[2] + pp$radii[3,2]
> mid <- straightarrow (to = m1, from = m2, arr.type = "triangle",
+                       arr.pos = 0.3, lwd = 1)
> text(mid[1]-0.01, mid[2]+0.03, "f3", cex = 0.8)
```

2.3. Plotting a transition matrix

The next example uses formulae to label the arrows [1]. This is done by passing a data.frame rather than a matrix to function `plotmat`

```
> # Create population matrix
> #
> Numgenerations   <- 6
> DiffMat  <- matrix(data = 0, nrow = Numgenerations, ncol = Numgenerations)
> AA <- as.data.frame(DiffMat)
> AA[[1,4]] <- "f[3]"
> AA[[1,5]] <- "f[4]"
> AA[[1,6]] <- "f[5]"
> #
> AA[[2,1]] <- "s[list(0,1)]"
> AA[[3,2]] <- "s[list(1,2)]"
> AA[[4,3]] <- "s[list(2,3)]"
> AA[[5,4]] <- "s[list(3,4)]"
> AA[[6,5]] <- "s[list(4,5)]"
> #
> name  <- c(expression(Age[0]), expression(Age[1]), expression(Age[2]),
+            expression(Age[3]), expression(Age[4]), expression(Age[5]))
> #
> plotmat(A = AA, pos = 6, curve = 0.7, name = name, lwd = 2,
+         arr.len = 0.6, arr.width = 0.25, my = -0.2,
+         box.size = 0.05, arr.type = "triangle", dtext = 0.95,
+         main = "Age-structured population model 1")
```

2.4. Another transition matrix

The data set `Teasel` contains the transition matrix of the population dynamics model of teasel (Dipsacus sylvestris), a European perennial weed, (Caswell 2001; Soetaert and Herman 2009b)

```
> Teasel
```

	DS 1yr	DS 2yr	R small	R medium	R large	F
DS 1yr	0.000	0.00	0.000	0.000	0.000	322.380
DS 2yr	0.966	0.00	0.000	0.000	0.000	0.000
R small	0.013	0.01	0.125	0.000	0.000	3.448
R medium	0.007	0.00	0.125	0.238	0.000	30.170
R large	0.008	0.00	0.038	0.245	0.167	0.862
F	0.000	0.00	0.000	0.023	0.750	0.000

This dataset is plotted using curved arrows; we specify the curvature in a matrix called `curves`.

[1] This is now possible thanks to Yvonnick Noel, Univ. Rennes, France

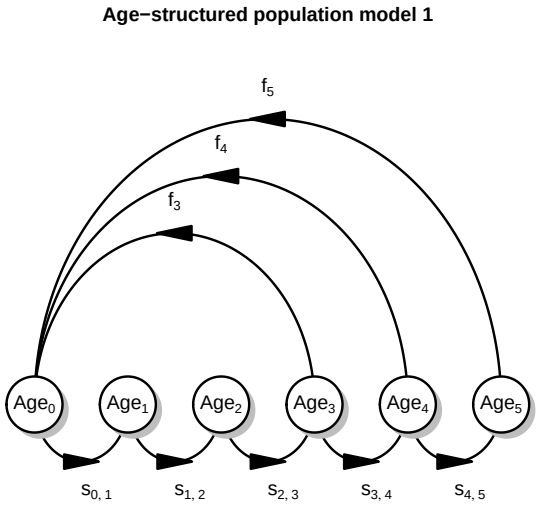

Figure 3: A transition matrix

```
> curves <- matrix(nrow = ncol(Teasel), ncol = ncol(Teasel), 0)
> curves[3, 1] <- curves[1, 6] <- -0.35
> curves[4, 6] <- curves[6, 4] <- curves[5, 6] <- curves[6, 5] <- 0.08
> curves[3, 6] <-  0.35
> plotmat(Teasel, pos = c(3, 2, 1), curve = curves,
+         name = colnames(Teasel), lwd = 1, box.lwd = 2,
+         cex.txt = 0.8, box.cex = 0.8, box.size = 0.08,
+         arr.length = 0.5, box.type = "circle", box.prop = 1,
+         shadow.size = 0.01, self.cex = 0.6, my = -0.075, mx = -0.01,
+         relsize = 0.9, self.shiftx = c(0, 0, 0.125, -0.12, 0.125, 0),
+         self.shifty = 0, main = "Teasel population model")
```

3. plotweb - plotting webs based on matrix input

Given a matrix containing flow values (from rows to columns), function `plotweb` will plot a web. The elements are positioned on a circle, and connected by arrows; the magnitude of web flows determines the thickness of the arrow.

This function is less flexible than `plotmat`, although it does allow to color the arrows differently.

```
> BB <- matrix(nrow = 20, ncol = 20, 1:20)
> diag(BB) <- 0
```

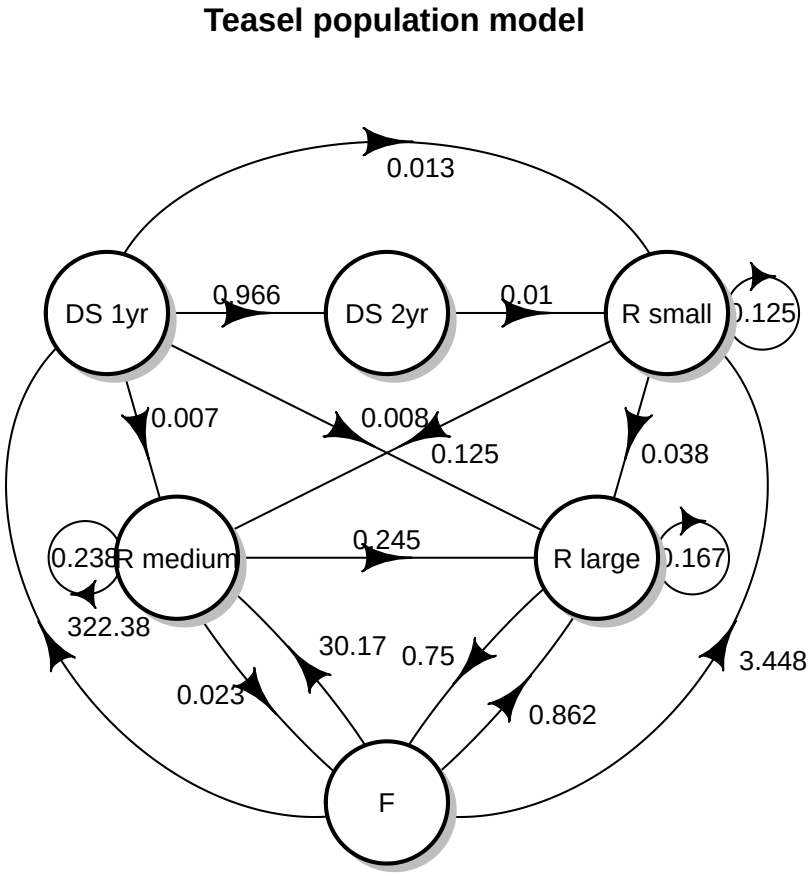

Figure 4: The Teasel data set

```
NULL
```

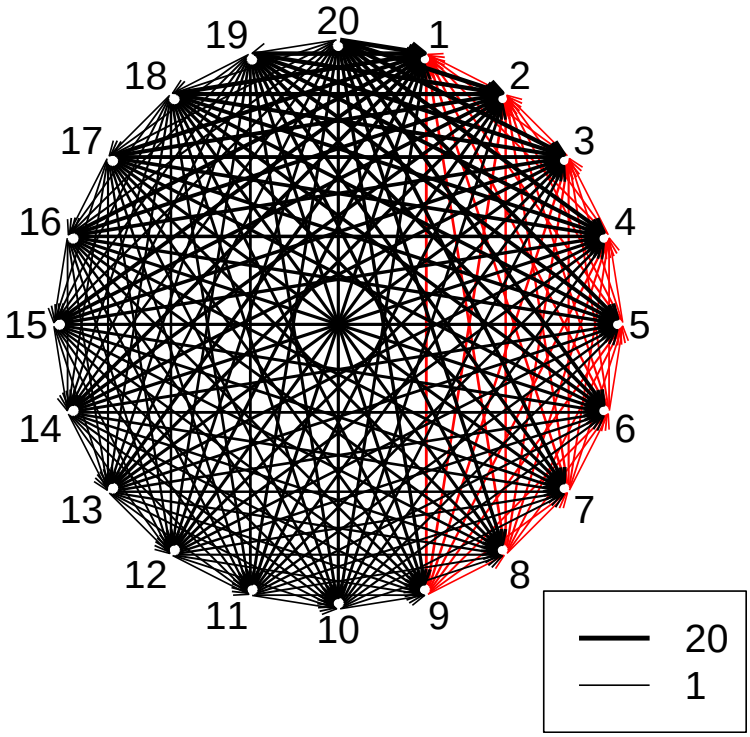

Figure 5: Plotweb

```
> Col <- BB
> Col[] <- "black"
> Col[BB<10]<- "red"
> plotweb(BB, legend = TRUE, maxarrow = 3, arr.col = Col)
```

```
NULL
```

```
> par(mfrow = c(1, 1))
```

3.1. Foodwebs

Dataset `Rigaweb` ((Donali *et al.* 1999) contains flow values for the food web of the Gulf of Riga planktonic system.

```
> Rigaweb
```

```
                    P1      P2        B       N        Z        D      DOC       CO2
P1              0.0000  0.0000   0.0000 4.12297 10.49431 0.000000 1.565910  17.22501
P2              0.0000  0.0000   0.0000 0.00000 16.79755 4.457164 2.723090  29.95399
B               0.0000  0.0000   0.0000 9.44000  0.00000 0.000000 0.000000 244.99223
N               0.0000  0.0000   0.0000 0.00000  0.00000 0.000000 0.000000  13.40297
Z               0.0000  0.0000   0.0000 0.00000  0.00000 3.183226 3.963226  30.19580
D               0.0000  0.0000   0.0000 0.00000 12.34039 0.000000 0.000000   0.00000
DOC             0.0000  0.0000 261.1822 0.00000  0.00000 0.000000 0.000000   0.00000
CO2            31.3182 54.4618   0.0000 0.00000  0.00000 0.000000 0.000000   0.00000
Sedimentation   0.0000  0.0000   0.0000 0.00000  0.00000 0.000000 0.000000   0.00000
               Sedimentation
P1                      0.10
P2                      0.34
B                       0.00
N                       0.00
Z                       0.78
D                      13.92
DOC                     0.00
CO2                     0.00
Sedimentation           0.00
```

```
> plotweb(Rigaweb, main = "Gulf of Riga food web",
+         sub = "mgC/m3/d", val = TRUE)
```

4. functions to create flow charts

The various functions are given in table (1) [2].

The code below generates a flow chart

```
> par(mar = c(1, 1, 1, 1))
> openplotmat()
> elpos  <- coordinates (c(1, 1, 2, 4))
> fromto <- matrix(ncol = 2, byrow = TRUE,
+                  data = c(1, 2, 2, 3, 2, 4, 4, 7, 4, 8))
> nr     <- nrow(fromto)
> arrpos <- matrix(ncol = 2, nrow = nr)
> for (i in 1:nr)
+  arrpos[i, ] <- straightarrow (to = elpos[fromto[i, 2], ],
+                                from = elpos[fromto[i, 1], ],
+                                lwd = 2, arr.pos = 0.6, arr.length = 0.5)
> textellipse(elpos[1,], 0.1,       lab = "start",           box.col = "green",
+             shadow.col = "darkgreen", shadow.size = 0.005, cex = 1.5)
> textrect   (elpos[2,], 0.15, 0.05,lab = "found term?",     box.col = "blue",
+             shadow.col = "darkblue", shadow.size = 0.005, cex = 1.5)
```

[2] textparallel was implemented by Michael Folkes, Canada

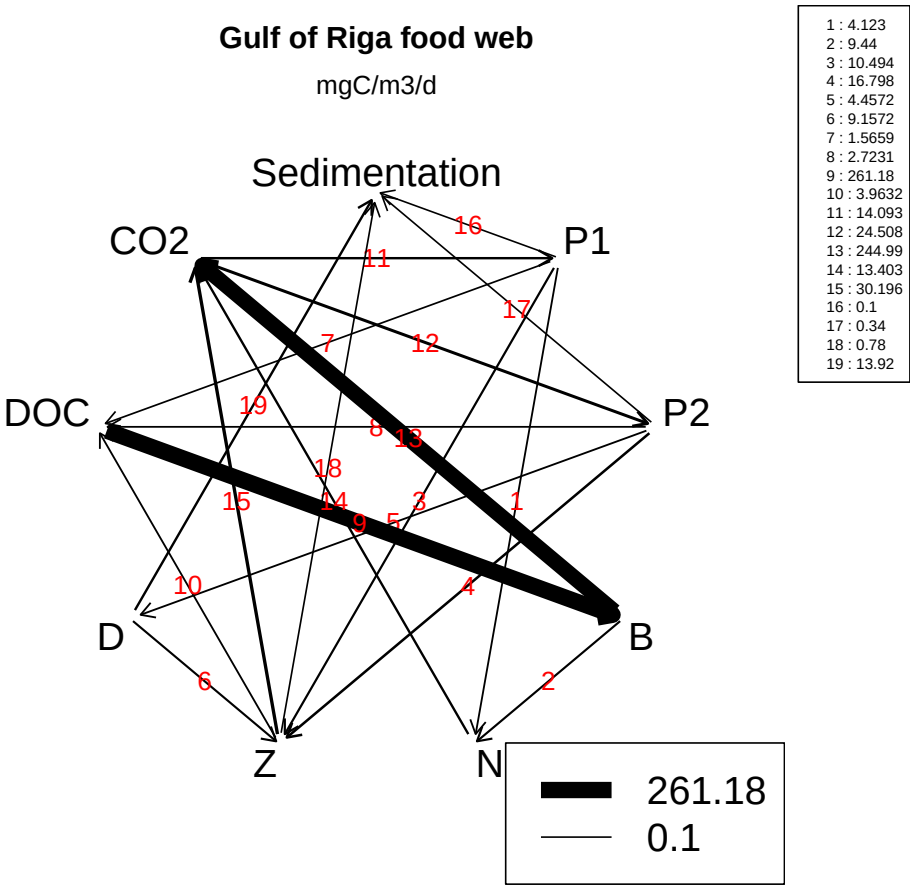

Figure 6: The Gulf of Riga data set

Table 1: Summary of flowchart functions

Function	Description
openplotmat	creates an empty plot
coordinates	calculates coordinates of elements, neatly arranged in rows/columns
bentarrow	adds 2-segmented arrow between two points
curvedarrow	adds curved arrow between two points
segmentarrow	adds 3-segmented arrow between two points
selfarrow	adds a circular self-pointing arrow
splitarrow	adds a branched arrow between several points
straightarrow	adds straight arrow between two points
treearrow	adds dendrogram-like branched arrow between several points
shadowbox	adds a box with a shadow to a plot
textdiamond	adds lines of text in a diamond-shaped box to a plot
textellipse	adds lines of text in a ellipse-shaped box to a plot
textempty	adds lines of text on a colored background to a plot
texthexa	adds lines of text in a hexagonal box to a plot
textmulti	adds lines of text in a multigonal box to a plot
textparallel	adds lines of text in a parallelogram to a plot
textplain	adds lines of text to a plot
textrect	adds lines of text in a rectangular-shaped box to a plot
textround	adds lines of text in a rounded box to a plot

```
> textrect   (elpos[4,], 0.15, 0.05,lab = "related?",        box.col = "blue",
+             shadow.col = "darkblue", shadow.size = 0.005, cex = 1.5)
> textellipse(elpos[3,], 0.1, 0.1,  lab = c("other","term"), box.col = "orange",
+             shadow.col = "red", shadow.size = 0.005, cex = 1.5)
> textellipse(elpos[3,], 0.1, 0.1,  lab = c("other","term"), box.col = "orange",
+             shadow.col = "red", shadow.size = 0.005, cex = 1.5)
> textellipse(elpos[7,], 0.1, 0.1,  lab = c("make","a link"),box.col = "orange",
+             shadow.col = "red", shadow.size = 0.005, cex = 1.5)
> textellipse(elpos[8,], 0.1, 0.1,  lab = c("new","article"),box.col = "orange",
+             shadow.col = "red", shadow.size = 0.005, cex = 1.5)
> #
> dd <- c(0.0, 0.025)
> text(arrpos[2, 1] + 0.05, arrpos[2, 2], "yes")
> text(arrpos[3, 1] - 0.05, arrpos[3, 2], "no")
> text(arrpos[4, 1] + 0.05, arrpos[4, 2] + 0.05, "yes")
> text(arrpos[5, 1] - 0.05, arrpos[5, 2] + 0.05, "no")
> #
```

The different types of text boxes are generated with the following code:

```
> openplotmat(main = "textbox shapes")
> rx <- 0.1
> ry <- 0.05
```

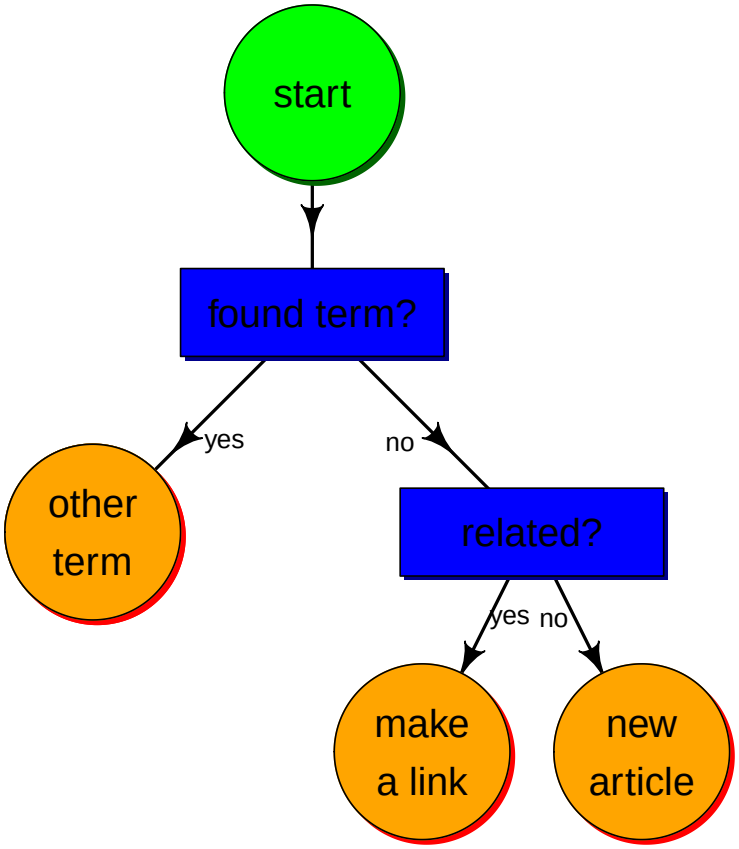

Figure 7: A flow chart

```
> pos <- coordinates(c(1, 1, 1, 1, 1, 1,1 ), mx = -0.2)
> textdiamond(mid = pos[1,], radx = rx, rady = ry, lab = LETTERS[1],
+             cex = 2, shadow.col = "lightblue")
> textellipse(mid = pos[2,], radx = rx, rady = ry, lab = LETTERS[2],
+             cex = 2, shadow.col = "blue")
> texthexa(mid = pos[3,], radx = rx, rady = ry, lab = LETTERS[3],
+          cex = 2, shadow.col = "darkblue")
> textmulti(mid = pos[4,], nr = 7, radx = rx, rady = ry, lab = LETTERS[4],
+          cex = 2, shadow.col = "red")
> textrect(mid = pos[5,], radx = rx, rady = ry, lab = LETTERS[5],
+          cex = 2, shadow.col = "darkred")
> textround(mid = pos[6,], radx = rx, rady = ry, lab = LETTERS[6],
+          cex = 2, shadow.col = "black")
> textparallel(mid = pos[7,], radx = rx, rady = ry, lab = LETTERS[7],
+          cex = 2, theta = 40, shadow.col = "black")
> textempty(mid = pos[8,], lab = LETTERS[8], cex = 2, box.col = "yellow")
> pos[ ,1] <- pos[ ,1] + 0.5
> text(pos[ ,1],pos[ ,2], c("textdiamond", "textellipse", "texthexa",
+                   "textmulti", "textrect", "textround",
+                   "textparallel", "textempty"))
```

The different types of arrows are generated with the following code:

```
> par(mar = c(1, 1, 1, 1))
> openplotmat(main = "Arrowtypes")
> elpos <- coordinates (c(1, 2, 1), mx = 0.1, my = -0.1)
> curvedarrow(from = elpos[1, ], to = elpos[2, ], curve = -0.5,
+             lty = 2, lcol = 2)
> straightarrow(from = elpos[1, ], to = elpos[2, ], lty = 3, lcol = 3)
> segmentarrow (from = elpos[1, ], to = elpos[2, ], lty = 1, lcol = 1)
> treearrow    (from = elpos[2:3, ], to = elpos[4, ], lty = 4, lcol = 4)
> bentarrow (from = elpos[3, ], to = elpos[3, ]-c(0.1, 0.1),
+            arr.pos = 1, lty = 5, lcol = 5)
> bentarrow(from = elpos[1, ], to = elpos[3, ], lty = 5, lcol = 5)
> selfarrow(pos = elpos[3, ], path = "R",lty = 6, curve = 0.075, lcol = 6)
> splitarrow(from = elpos[1, ], to = elpos[2:3, ], lty = 1,
+            lwd = 1, dd = 0.7, arr.side = 1:2, lcol = 7)
> for ( i in 1:4)
+   textrect (elpos[i, ], 0.05, 0.05, lab = i, cex = 1.5)
> legend("topright", lty = 1:7, legend = c("segmentarrow",
+        "curvedarrow", "straightarrow", "treearrow", "bentarrow",
+        "selfarrow", "splitarrow"), lwd = c(rep(2, 6), 1), col = 1:7)
```

5. functions to draw electrical networks

Since version 1.6, it is possible to use `diagram` to draw electrical networks. Below I give an example of a small transistor circuit.

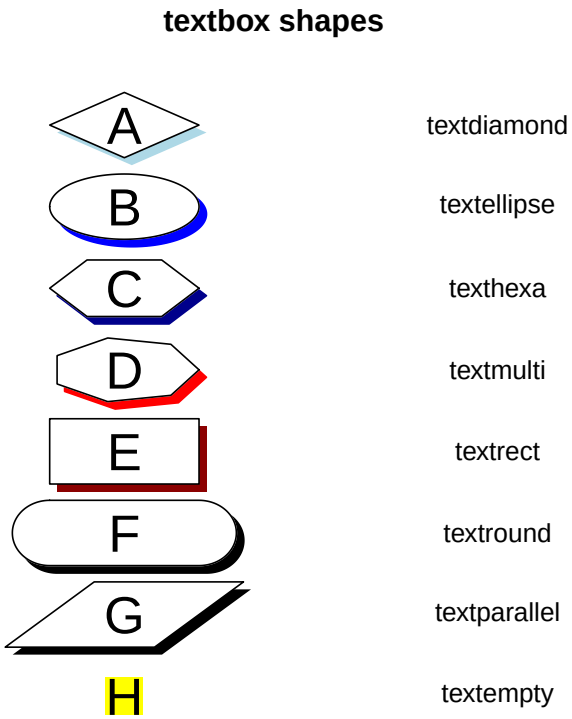

Figure 8: The text boxes

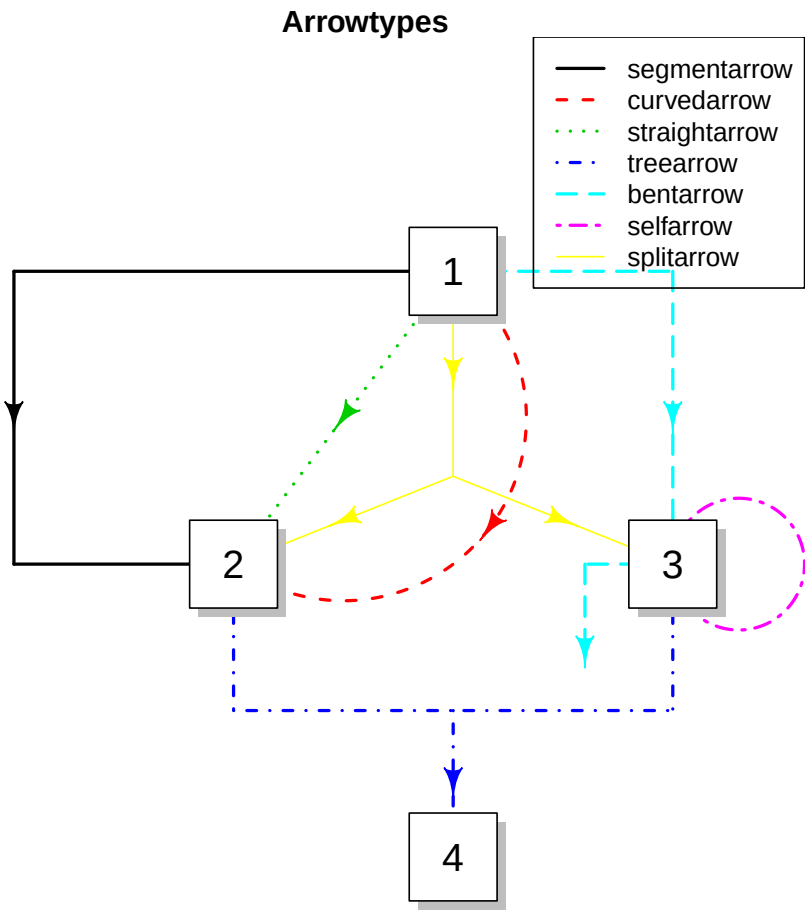

Figure 9: The arrow types

```
> layoutmat <- matrix(data = c(rep(1, 12), 2, 3, 4, 5),
+                     nrow = 4, ncol = 4, byrow = TRUE)
> nf <- layout(layoutmat, respect = FALSE)
> par(lwd = 1.5)
> par(mar = c(0, 0, 2, 0))
> emptyplot(main = "transistor Amplifier", asp = FALSE,
+           ylim = c(-0.1, 1), xlim = c(-0.1, 1.1))
> x1 <- 0; x2 <- 0.2; x3 <- 0.4; x4 <- 0.6; x5 <- 0.8; x6 <- 1
> y1 <- 0.05; y2 <- 0.4; y3 <- 0.5; y4 <- 0.6; y5 <- 0.95
> x23 <- (x2 + x3)/2
> x56 <- (x5 + x6)/2
> lines(c(x2, x6, x6, x2, x2, x1, x1, x23, x3, x3),
+       c(y1, y1, y5, y5, y1, y1, y3, y3,  y4, y5))
> lines(c(x23, x3, x3),            c(y3,  y2, y1))
> lines(c(x3,  x4,  x4),           c(y2,  y2,  y1))
> lines(c(x3,  x5, x5),            c(y4,  y4, y1))
> en.Amplifier(c(x23, y3), r = 0.035)
> en.Signal(c(x1, 0.2), lab = expression("U"["in"]))
> en.Signal(c(x6, y2), lab = expression("U"["b"]))
> straightarrow(c(x1 - 0.05, 0.23), c(x1 - 0.05, 0.17),
+               arr.pos =1, arr.type = "triangle", lwd = 1)
> straightarrow(c(x6 + 0.05, y2 + 0.03), c(x6 + 0.05, y2 - 0.03),
+               arr.pos = 1, arr.type = "triangle", lwd = 1)
> en.Node(c(x1, y3), lab = "u1")
> en.Node(c(x2, y3), lab = "u2")
> en.Node(c(x3, y2), lab = "u3", pos = 1.5)
> en.Node(c(x3, y4), lab = "u4", pos = 2.5)
> en.Node(c(x5, y4), lab = "u5")
> en.Capacitator(c(0.5*(x1 + x2),y3), lab = "C1", vert = FALSE)
> en.Capacitator(c(x4, y4), lab = "C3", vert = FALSE)
> en.Capacitator(c(x4, 0.5*(y1+y2)), lab = "C2", vert = TRUE)
> en.Resistor(c(x1, y2), lab = "R0")
> en.Resistor(c(x2, 0.5*(y1+y2)), lab = "R1")
> en.Resistor(c(x2, 0.5*(y4+y5)), lab = "R2")
> en.Resistor(c(x3, 0.5*(y4+y5)), lab = "R4")
> en.Resistor(c(x3, 0.5*(y1+y2)), lab = "R3")
> en.Resistor(c(x5, 0.5*(y1+y2)), lab = "R5")
> en.Ground(c(1.0, 0.05))
> par(mar=c(2, 2, 2, 2))
> emptyplot(main = "transistor")
> lines(c(0.1, 0.5,0.9), c(0.5, 0.5, 0.9))
> lines(c(0.5, 0.9), c(0.5, 0.1))
> lines(c(0.5, 0.5), c(0.4, 0.6))
> text(0.2, 0.4, "Gate", font = 3)
> text(0.8, 0.9, "Drain", font = 3,adj = 1)
> text(0.8, 0.1, "Source", font = 3,adj = 1)
> en.Amplifier(c(0.5, 0.5), r = 0.15)
```

```
> box(col = "grey")
> emptyplot(main = "capacitator")
> straightarrow(c(0.5, 0.9), c(0.5, 0.1),
+               arr.pos = 0.3, arr.length = 0.25, arr.type = "triangle")
> en.Capacitator(c(0.5, 0.5), width = 0.075, length = 0.5, vert = TRUE)
> text(0.4, 0.65, "i", font = 3, cex = 2)
> straightarrow(c(0.8, 0.3), c(0.8, 0.77), arr.pos = 1,
+               arr.length = 0.25, arr.type = "triangle", lwd = 1)
> text(0.925, 0.65, "v", font = 3, cex = 2)
> text(0.15, 0.5, "C", font = 3, cex = 2)
> box(col = "grey")
> emptyplot(main = "resistor")
> straightarrow(c(0.5, 0.9), c(0.5, 0.1), arr.pos = 0.2,
+               arr.length = 0.25, arr.type = "triangle", lwd = 1)
> text(0.4, 0.85, "i", font = 3, cex = 2)
> en.Resistor(c(0.5, 0.5), width = 0.25, length = 0.35 )
> straightarrow(c(0.8, 0.3), c(0.8, 0.77), arr.pos = 1,
+               arr.length = 0.25, arr.type = "triangle", lwd = 1)
> text(0.925, 0.65, "v", font = 3, cex = 2)
> text(0.5, 0.5, "R", font = 3, cex = 2)
> box(col = "grey")
> emptyplot(main = "voltage source")
> lines(c(0.5, 0.5), c(0.1, 0.9))
> en.Signal(c(0.5, 0.5), r = 0.15)
> straightarrow(c(0.8, 0.3), c(0.8, 0.77), arr.pos = 1,
+               arr.length = 0.25, arr.type = "triangle", lwd = 1)
> text(0.925, 0.65, "v", font = 3, cex = 2)
> box(col = "grey")
```

This vignette was created using Sweave (Leisch 2002).

The package is on CRAN, the R-archive website ((R Development Core Team 2008))

More examples can be found in the demo's of package **ecolMod** (Soetaert and Herman 2009a)

References

Caswell H (2001). *Matrix population models: construction, analysis, and interpretation*. Second edition edition. Sinauer, Sunderland.

Donali E, Olli K, Heiskanen AS, Andersen T (1999). "Carbon flow patterns in the planktonic food web of the Gulf of Riga, the Baltic Sea: a reconstruction by the inverse method." *Journal of Marine Systems*, **23**, 251–268.

Leisch F (2002). "Sweave: Dynamic Generation of Statistical Reports Using Literate Data Analysis." In W Härdle, B Rönz (eds.), *Compstat 2002 — Proceedings in Computational Statistics*, pp. 575–580. Physica Verlag, Heidelberg. ISBN 3-7908-1517-9, URL http://www.stat.uni-muenchen.de/~leisch/Sweave.

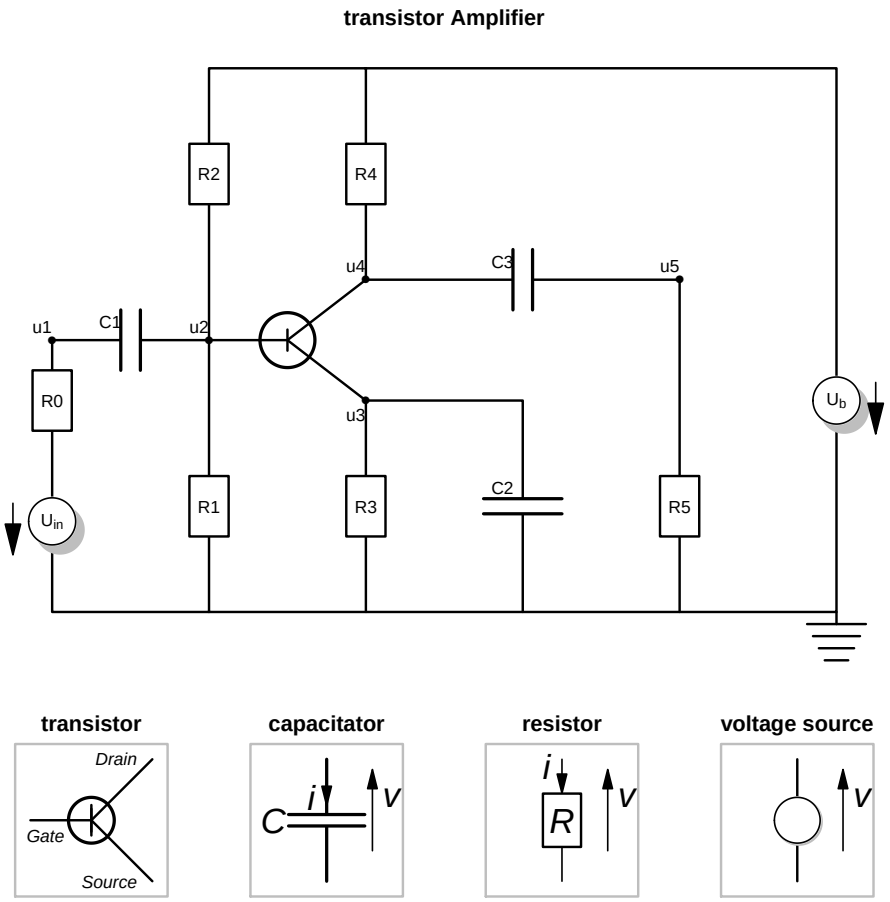

Figure 10: Drawing an electrical network with package diagram

Niquil N, Jackson G, Legendre L, Delesalle B (1998). "Inverse model analysis of the planktonic food web of Takapoto Atoll (French Polynesia)." *Marine Ecology Progress Series*, **165**, 17–29.

R Development Core Team (2008). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

Soetaert K (2009a). *diagram: Functions for visualising simple graphs (networks), plotting flow diagrams*. R package version 1.5.

Soetaert K (2009b). *shape: Functions for plotting graphical shapes, colors*. R package version 1.2.2.

Soetaert K, Cash JR, Mazzia F (2012). *Solving Differential Equations in R*. Springer. In press.

Soetaert K, Herman PM (2009a). *ecolMod: "A practical guide to ecological modelling - using R as a simulation platform"*. R package version 1.3.

Soetaert K, Herman PMJ (2009b). *A Practical Guide to Ecological Modelling. Using R as a Simulation Platform*. Springer. ISBN 978-1-4020-8623-6.

Soetaert K, Van den Meersche K, van Oevelen D (2009). *limSolve: Solving linear inverse models*. R package version 1.5.

Soetaert K, van Oevelen D (2009). *LIM: Linear Inverse Model examples and solution methods*. R package version 1.4.

Affiliation:

Karline Soetaert
Royal Netherlands Institute of Sea Research (NIOZ)
4401 NT Yerseke, Netherlands E-mail: karline.soetaert@nioz.nl
URL: http://www.nioz.nl