Melissa Clarkson
March 2010
sna version 2.0-1
document version 1

A guide to the gplot layout algorithms of the sna library for R

By default, the `gplot` function for drawing node-and-edge representations of networks will use the the `fruchtermanreingold` force-directed layout algorithm. Other layout algorithms can be used, and many allow parameters of the algorithms to be specified.

This guide provides only an overview of the algorithms available. See the `gplot.layout` documentation for descriptions of the parameters of the algorithms.

The form of the function

```
gplot(data_matrix, mode="algorithm")
```
will draw the graph using the specified layout algorithm, with default parameters for the layout algorithm.

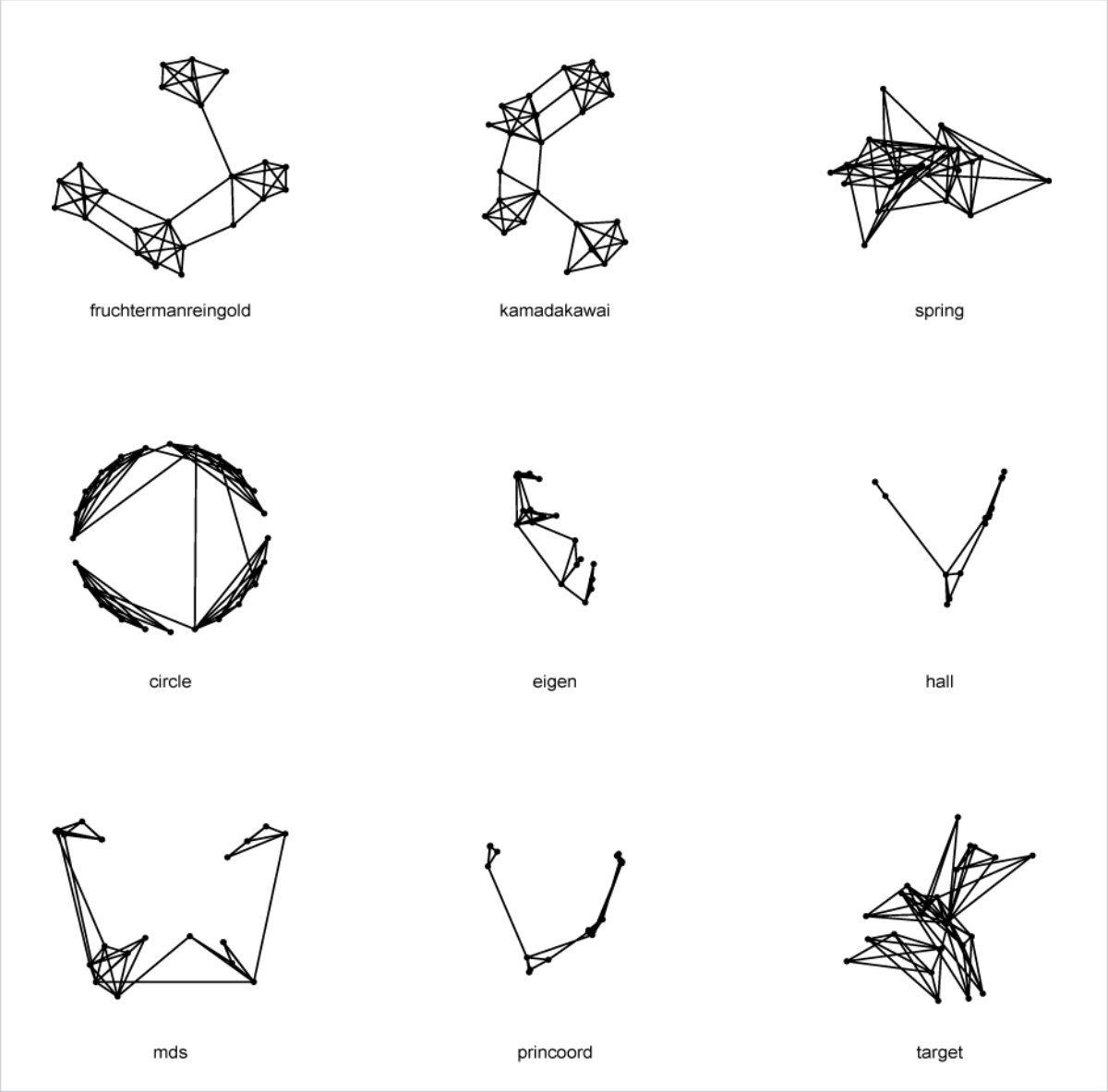
```
gplot(data_matrix, mode="algorithm", layout.par=list(argument=parameter, argument=parameter))
```
will draw the graph using the specified layout algorithm and parameters.

The layout algorithms

`"fruchtermanreingold"`	A variant of Fruchterman and Reingold's force-directed layout algorithm (default). With optional arguments.
`"kamadakawai"`	A version of the Kamada-Kawai force-directed layout algorithm. With optional arguments.
`"spring"`	A spring embedder algorithm. With optional arguments.
`"circle"`	Nodes are placed in a circle, arranged clockwise by their order in the adjacency matrix. No additional arguments.
`"eigen"`	Node placement is based on the eigenstructure of the adjacency matrix. With optional arguments.
`"hall"`	Node placement based on the last two eigenvectors of the Laplacian of the input matrix. No additional arguments.
`"mds"`	Node placement based on multidimensional scaling of a specified distance matrix (matrix of rows and columns used by default). With optional arguments.
`"princoord"`	Node placement based on the eigenstructure of a given correlation/covariance matrix (matrix of rows and columns used by default). With optional arguments.
`"target"`	Arranges nodes on the radii of concentric circles, based on a vector of node values (affine-transformed Freeman degree centrality scores used by default). With optional arguments.
`"random"`	Nodes are placed randomly. Uses a uniform distribution by default, but a Gaussian or "Gaussian donut" distribution may be specified.

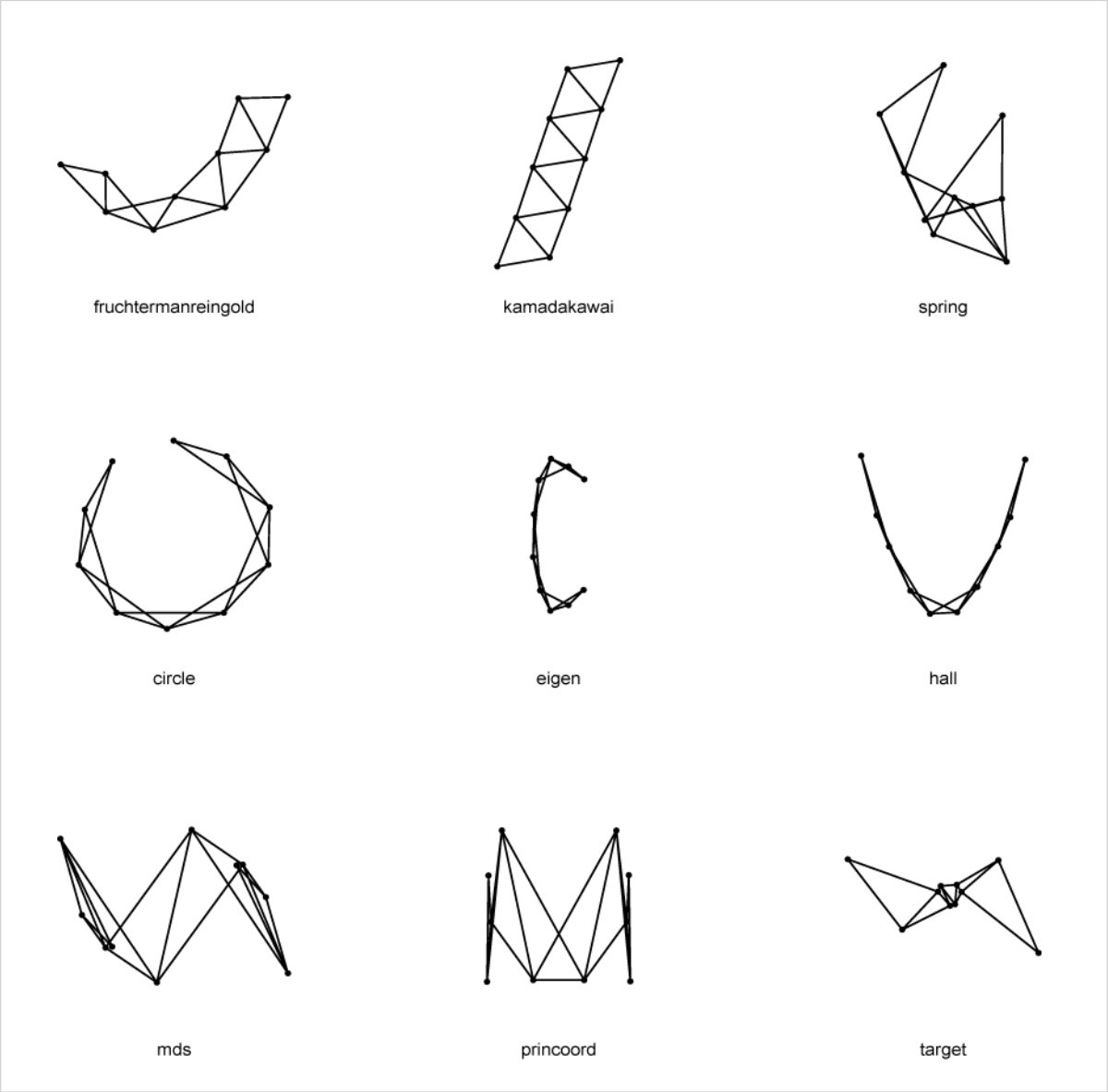
Examples (with default parameters)

Mixing matrix with low probability of between-class edges



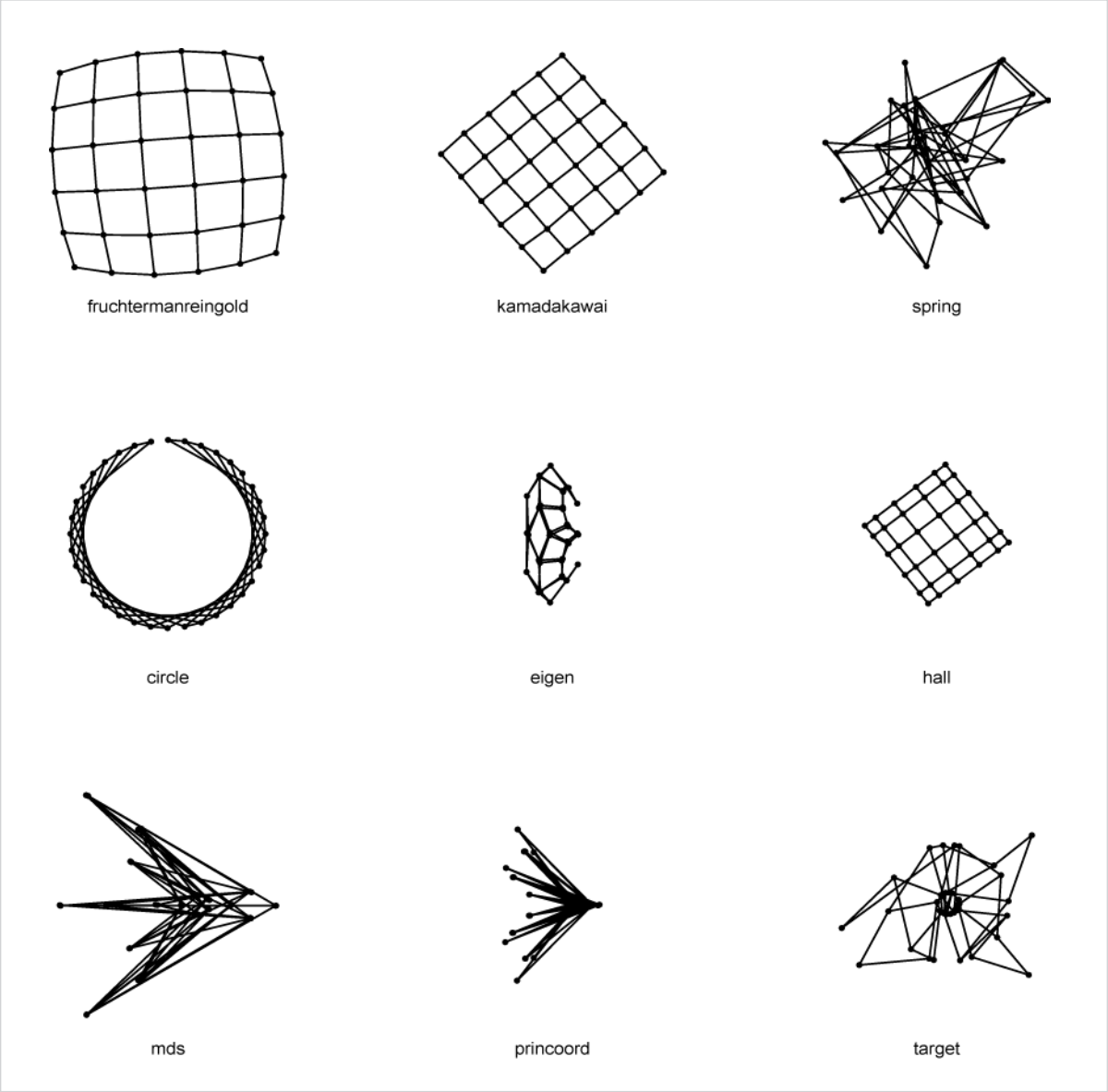

```
old_par <- par(no.readonly=TRUE)
par(mfrow=c(3,3))
par(mai=c(0.4, 0.4, 0.4, 0.4))

mix_vector <- c( 0.9, 0.05, 0, 0.05,  0.05, 0.9, 0.05, 0,  0, 0.05, 0.9, 0,  0.05, 0, 0.05, 0.9)
mix_matrix <- matrix(mix_vector, nrow=4)

layout_vector <- c("fruchtermanreingold", "kamadakawai", "spring", "circle", "eigen", "hall", "mds", "princoord", "target")

g <- rgnmix(1, rep(1:4, each=6), mix_matrix, mode="graph", method="probability")

for (i in 1:9) {
        gplot(g, gmode="graph", vertex.col="black", mode=layout_vector[i])
        title(sub=layout_vector[i], line=0.5)
}

par(old_par)
```

1-D lattice with neighborhood of 2



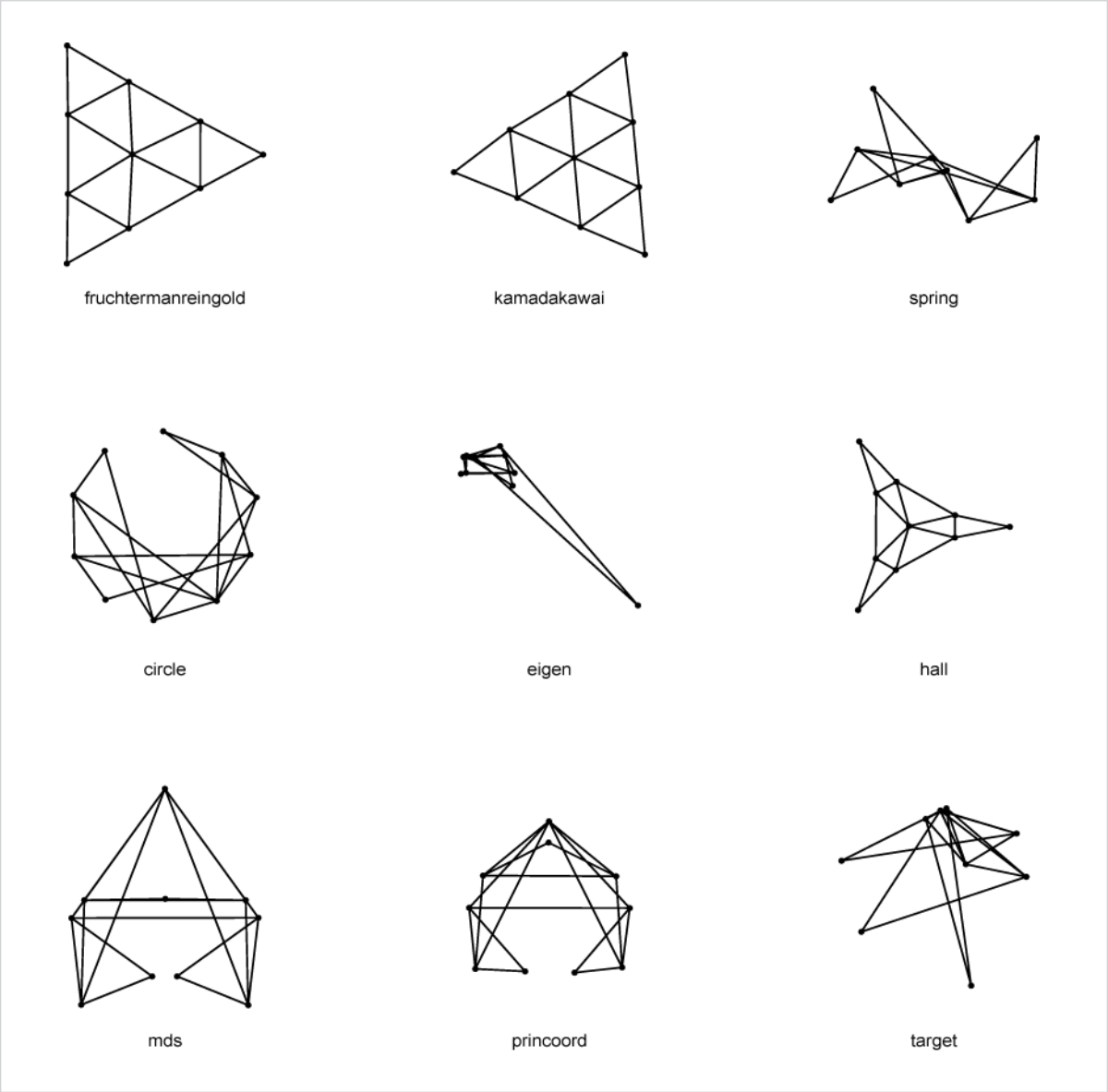

```
old_par <- par(no.readonly=TRUE)
par(mfrow=c(3,3))
par(mai=c(0.4, 0.4, 0.4, 0.4))

layout_vector <- c("fruchtermanreingold", "kamadakawai", "spring", "circle", "eigen", "hall", "mds", "princoord", "target")

g <- rgws(1,10,1,2,0)

for (i in 1:9) {
        gplot(g, gmode="graph", vertex.col="black", mode=layout_vector[i])
        title(sub=layout_vector[i], line=0.5)
}

par(old_par)
```

2-D lattice with neighborhood of 1



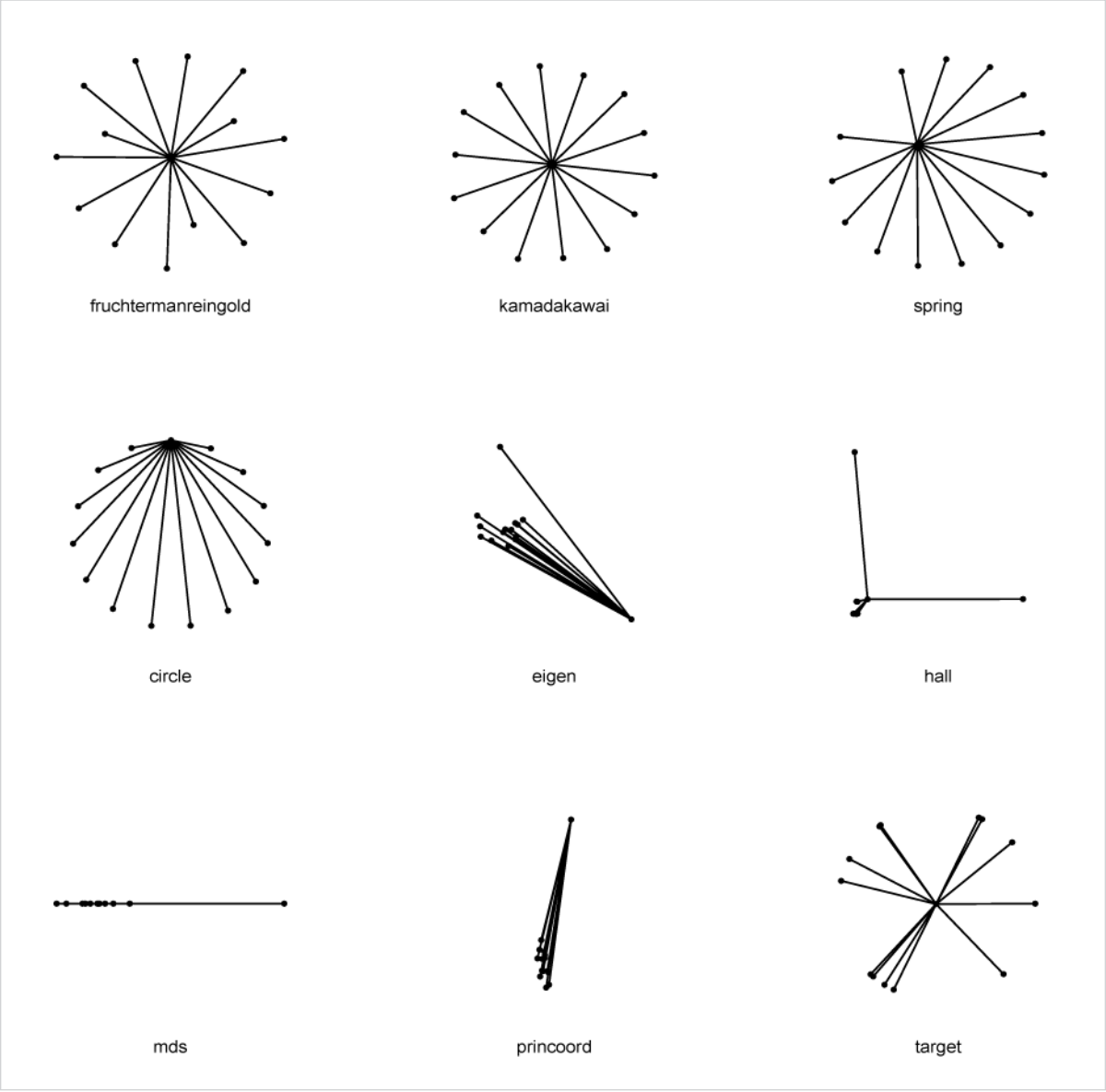

```
old_par <- par(no.readonly=TRUE)
par(mfrow=c(3,3))
par(mai=c(0.4, 0.4, 0.4, 0.4))

layout_vector <- c("fruchtermanreingold", "kamadakawai", "spring", "circle", "eigen", "hall", "mds", "princoord", "target")

g <- rgws(1,6,2,1,0)

for (i in 1:9) {
        gplot(g, gmode="graph", vertex.col="black", mode=layout_vector[i])
        title(sub=layout_vector[i], line=0.5)
}

par(old_par)
```

Triangle-shaped lattice



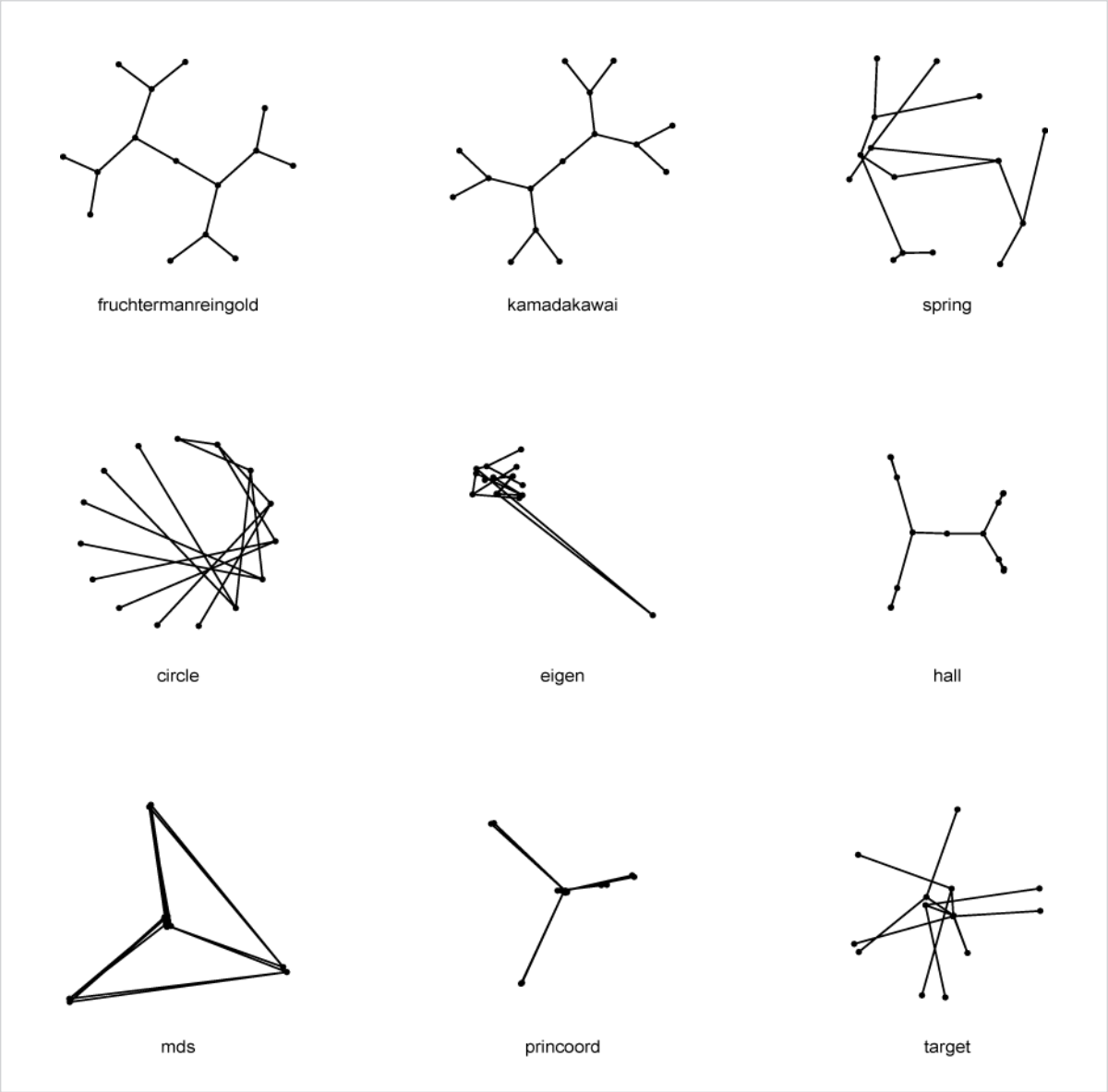

```
old_par <- par(no.readonly=TRUE)
par(mfrow=c(3,3))
par(mai=c(0.4, 0.4, 0.4, 0.4))
layout_vector <- c("fruchtermanreingold", "kamadakawai", "spring", "circle", "eigen", "hall", "mds", "princoord", "target")

g <- matrix(0, 10, 10)
g[1,2] <- 1;     g[1,3] <- 1
g[2,3] <- 1;     g[2,4] <- 1
g[2,5] <- 1;     g[3,5] <- 1
g[3,6] <- 1;     g[4,5] <- 1
g[4,7] <- 1;     g[4,8] <- 1
g[5,6] <- 1;     g[5,8] <- 1
g[5,9] <- 1;     g[6,9] <- 1
g[6,10] <- 1;    g[7,8] <- 1
g[8,9] <- 1;     g[9,10] <- 1

for (i in 1:9) {
         gplot(g, gmode="graph", vertex.col="black", mode=layout_vector[i])
         title(sub=layout_vector[i], line=0.5)
}

par(old_par)
```

Star



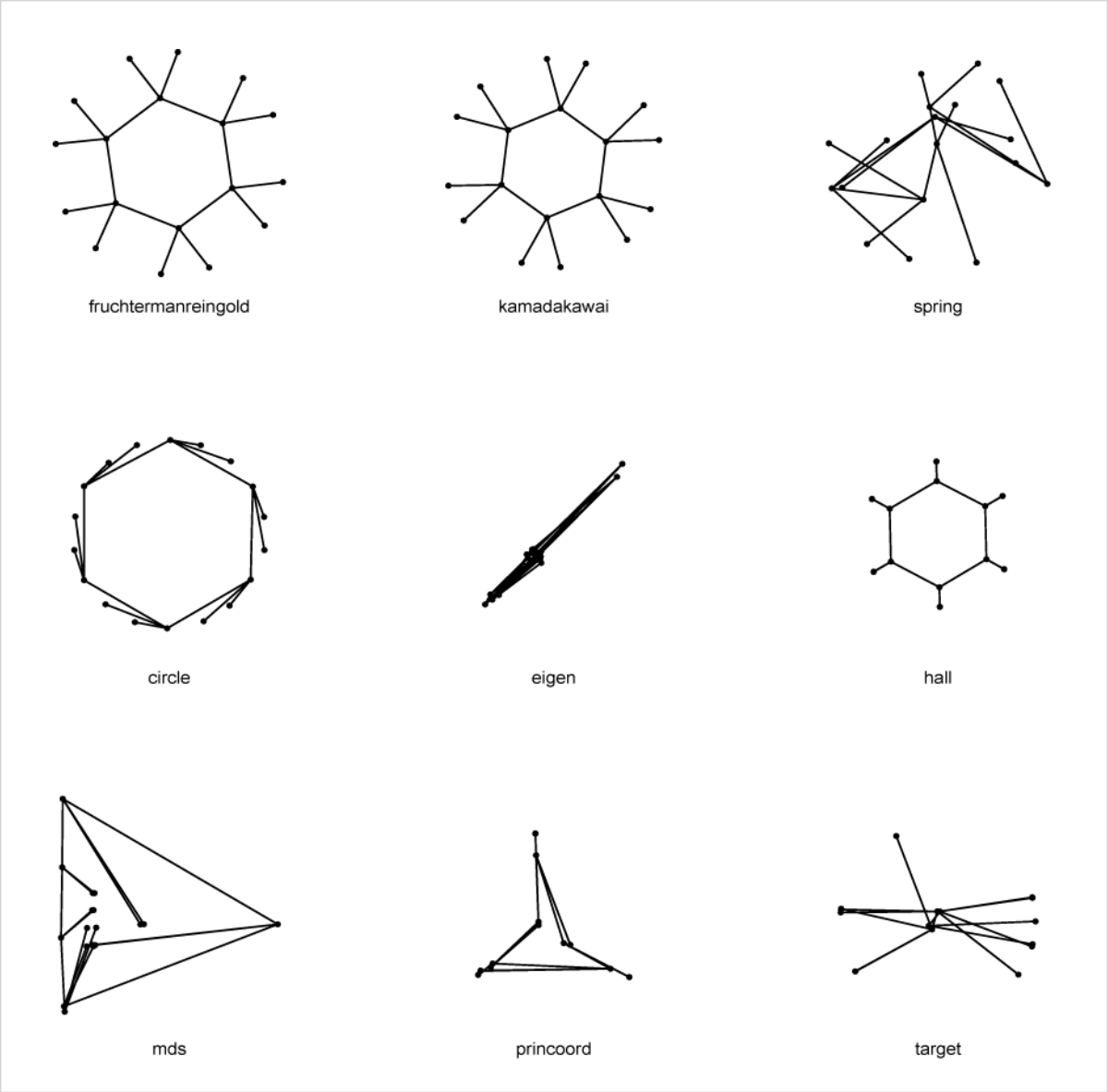

```
old_par <- par(no.readonly=TRUE)
par(mfrow=c(3,3))
par(mai=c(0.4, 0.4, 0.4, 0.4))

layout_vector <- c("fruchtermanreingold", "kamadakawai", "spring", "circle", "eigen", "hall", "mds", "princoord", "target")

g <- matrix(0, 15, 15)
g[1,] <- 1

for (i in 1:9) {
        gplot(g, gmode="graph", vertex.col="black", mode=layout_vector[i])
        title(sub=layout_vector[i], line=0.5)
}

par(old_par)
```

Tree



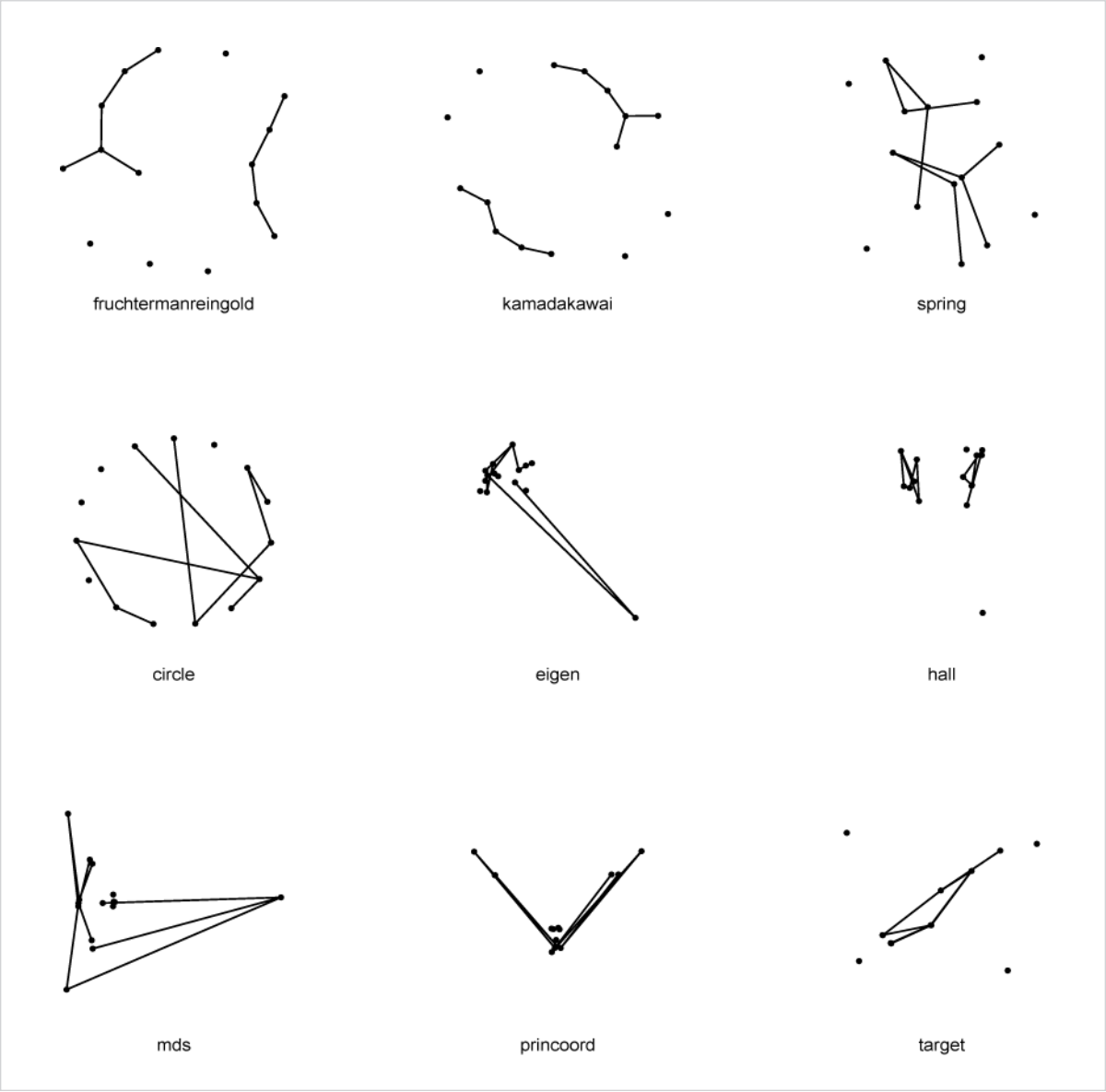

```
old_par <- par(no.readonly=TRUE)
par(mfrow=c(3,3))
par(mai=c(0.4, 0.4, 0.4, 0.4))

layout_vector <- c("fruchtermanreingold", "kamadakawai", "spring", "circle", "eigen", "hall", "mds", "princoord", "target")

g <- matrix(0, 15, 15)
g[1,2] <- 1;     g[1,3] <- 1
g[2,4] <- 1;     g[2,5] <- 1
g[3,6] <- 1;     g[3,7] <- 1
g[4,8] <- 1;     g[4,9] <- 1
g[5,10] <- 1;    g[5,11] <- 1
g[6,12] <- 1;    g[6,13] <- 1
g[7,14] <- 1;    g[7,15] <- 1

for (i in 1:9) {
         gplot(g, gmode="graph", vertex.col="black", mode=layout_vector[i])
         title(sub=layout_vector[i], line=0.5)
}

par(old_par)
```

Graph with a cycle




```
old_par <- par(no.readonly=TRUE)
par(mfrow=c(3,3))
par(mai=c(0.4, 0.4, 0.4, 0.4))

layout_vector <- c("fruchtermanreingold", "kamadakawai", "spring", "circle", "eigen", "hall", "mds", "princoord", "target")

g <- matrix(0, 18, 18)
g[1,2] <- 1;     g[1,3] <- 1
g[1,4] <- 1;     g[4,5] <- 1
g[4,6] <- 1;     g[4,7] <- 1
g[7,8] <- 1;     g[7,9] <- 1
g[7,10] <- 1;    g[10,11] <- 1
g[10,12] <- 1;   g[10,13] <- 1
g[13,14] <- 1;   g[13,15] <- 1
g[13,16] <- 1;   g[16,17] <- 1
g[16,18] <- 1;   g[16,1] <- 1

for (i in 1:9) {
        gplot(g, gmode="graph", vertex.col="black", mode=layout_vector[i])
        title(sub=layout_vector[i], line=0.5)
}

par(old_par)
```

Sparse random graph



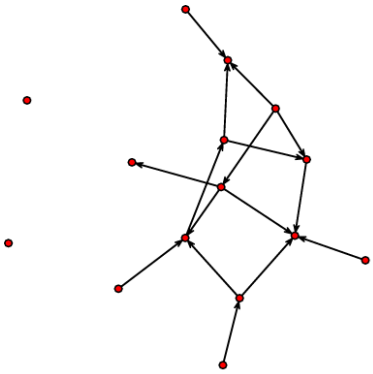

```
old_par <- par(no.readonly=TRUE)
par(mfrow=c(3,3))
par(mai=c(0.4, 0.4, 0.4, 0.4))

layout_vector <- c("fruchtermanreingold", "kamadakawai", "spring", "circle", "eigen", "hall", "mds", "princoord", "target")

g <- rgraph(15, tprob=0.05)

for (i in 1:9) {
        gplot(g, gmode="graph", vertex.col="black", mode=layout_vector[i])
        title(sub=layout_vector[i], line=0.5)
}

par(old_par)
```

Displaying a target diagram
gplot.target

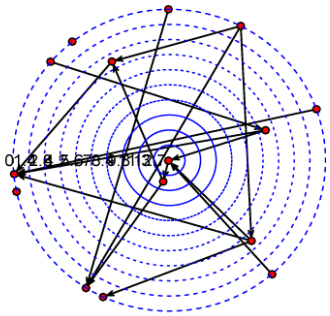
The "target" layout algorithm is used by gplot.target. The layout algorithm requires both a graph and a vector specifying the value of each node (such as the betweenness value).The layout algorithm places the node of highest value in the center and positions the remaining nodes at radii corresponding to their values. The gplot.target function draws concentric circles around the origin and displays the values of the radii.

The layout is optimized for nodes within mutual dyads. The remaining nodes are then added to the layout. See the R documentation for more details and graphical parameters to control circle radii, color, line style, and labeling.

Note: The circles that mark radii are drawn as a series of many short line segments. Therefore, circles made of dotted or dashed lines will not have a uniform appearance.



default layout with gplot()

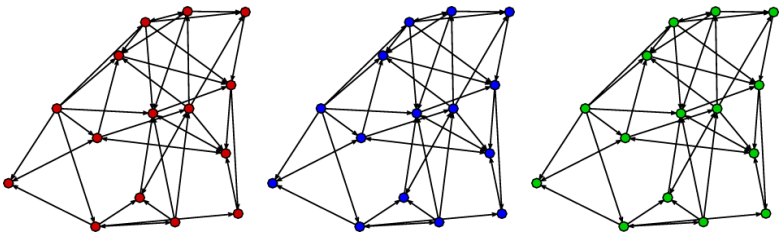


betweenness with gplot.target()

```
old_par <- par(no.readonly=TRUE)
par(mfrow=c(1,2))
par(mar=c(10, 1, 10, 1))
par(xpd=NA)

g <- rgraph(15, tprob=0.1)

gplot(g)
title(sub="default layout with gplot()", line=0)

gplot.target(g, betweenness(g))
title(sub="betweenness with gplot.target()", line=0)

par(old_par)
```

Saving node coordinates

gplot() returns a two-column matrix containing the x- and y-coordinates of the nodes. This matrix can be used to reproduce the positions of the nodes when the graph is drawn again.



```
old_par <- par(no.readonly=TRUE)
par(mfrow=c(1, 3))
par(mar=c(0, 0, 0, 0))

g <- rgraph(15, tprob=0.2)
positions <- gplot(g, vertex.col="red3", vertex.cex=2)
gplot(g, coord=positions, vertex.col="blue", vertex.cex=2)
gplot(g, coord=positions, vertex.col="green3", vertex.cex=2)

par(old_par)
```