

diagonals: Fat Diagonals in R

Bastiaan Quast

June 18, 2015

1 Introduction

We present the `diagonals` R package, which implements functions for dealing with **fat diagonals**. Fat diagonals are block matrix-diagonals that occur when two or more dimensions are mapped along a single edge of a matrix. For an asymmetric network graph (e.g. a dyadic social network) to be mapped to a matrix, we would need each node along each edge of matrix, however we would also need to map the direction of the tie, which is an additional dimension. Typically these would be represented as higher-order arrays (i.e. ≥ 3). In order to effectively visualise such arrays, it can be helpful to do so in a matrix. This could for instance be represented as in the following matrix (where the `i` and `o` suffices represent incoming and outgoing respectively).

```
##   Ai Ao Bi Bo Ci Co Di Do
## A  1  1  0  0  1  0  1  0
## B  0  1  1  1  0  1  0  1
## C  1  0  0  0  1  1  0  0
## D  1  0  1  0  1  1  1  1
```

Sometimes the ties of a node to itself are not particularly meaningful (e.g. feeling of amiability towards oneself) and can be removed. For a symmetric network this can simply be done using the function `diag()` in R's `base` package, e.g.

```
sm <- matrix(1, nrow=4, ncol=4)
diag(sm) <- NA
sm
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    NA     1     1     1
## [2,]     1    NA     1     1
## [3,]     1     1    NA     1
## [4,]     1     1     1    NA
```

However, for higher-order matrices this does not work well.

```
diag(m) <- NA
m
```

```
##   Ai Ao Bi Bo Ci Co Di Do
## A NA  1  0  0  1  0  1  0
## B  0 NA  1  1  0  1  0  1
## C  1  0 NA  0  1  1  0  0
## D  1  0  1 NA  1  1  1  1
```

It comes the `diagonals` package and its workhorse `fatdiag()` function. The function is designed to mimic the behaviour of the `diag()` as closely as possible, but with then for **fat diagonals**.

```
library(diagonals)
```

```
##
## D I
## A G
##   O N
##   A L
##   S
```

```
fatdiag(m, steps=4) <- NA
m
```

```
##   Ai Ao Bi Bo Ci Co Di Do
## A NA NA 0 0 1 0 1 0
## B 0 1 NA NA 0 1 0 1
## C 1 0 0 0 NA NA 0 0
## D 1 0 1 0 1 1 NA NA
```

Note that the `steps` argument defines the number of steps on the diagonal ladder. Alternatively we could set the `size` of the step.

```
fatdiag(m, size=c(1,2) ) <- 881:888
m
```

```
##   Ai  Ao  Bi  Bo  Ci  Co  Di  Do
## A 881 882  0  0  1  0  1  0
## B  0  1 883 884  0  1  0  1
## C  1  0  0  0 885 886  0  0
## D  1  0  1  0  1  1 887 888
```

So far we have been using the set `fatdiag()`, i.e. `fatdiag()<-`. However, we can also use the `fatdiag()` function either for diagonal extraction, or diagonal matrix creation.

```
fatdiag(m, steps = 4)
```

```
## [1] 881 882 883 884 885 886 887 888
```

Fat diagonal matrices can be created using a scalar:

```
fatdiag(9, steps=3)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
## [1,] 1    1    1    0    0    0    0    0    0
## [2,] 1    1    1    0    0    0    0    0    0
## [3,] 1    1    1    0    0    0    0    0    0
## [4,] 0    0    0    1    1    1    0    0    0
## [5,] 0    0    0    1    1    1    0    0    0
## [6,] 0    0    0    1    1    1    0    0    0
## [7,] 0    0    0    0    0    0    1    1    1
## [8,] 0    0    0    0    0    0    1    1    1
## [9,] 0    0    0    0    0    0    1    1    1
```

or using a vector:

```
fatdiag(1:27, steps=3)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
## [1,]    1    4    7    0    0    0    0    0    0
## [2,]    2    5    8    0    0    0    0    0    0
## [3,]    3    6    9    0    0    0    0    0    0
## [4,]    0    0    0   10   13   16    0    0    0
## [5,]    0    0    0   11   14   17    0    0    0
## [6,]    0    0    0   12   15   18    0    0    0
## [7,]    0    0    0    0    0    0   19   22   25
## [8,]    0    0    0    0    0    0   20   23   26
## [9,]    0    0    0    0    0    0   21   24   27
```

We can extract a fat diagonal and diagonalise it again.

```
m <- matrix(801:881, nrow=9, ncol=9)
fatdiag(fatdiag(m, steps=3), steps=3)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
## [1,]  801  810  819    0    0    0    0    0    0
## [2,]  802  811  820    0    0    0    0    0    0
## [3,]  803  812  821    0    0    0    0    0    0
## [4,]    0    0    0  831  840  849    0    0    0
## [5,]    0    0    0  832  841  850    0    0    0
## [6,]    0    0    0  833  842  851    0    0    0
## [7,]    0    0    0    0    0    0  861  870  879
## [8,]    0    0    0    0    0    0  862  871  880
## [9,]    0    0    0    0    0    0  863  872  881
```