Overview of the spnet package

Emmanuel Rousseaux, Marion Deville and Gilbert Ritschard

2014.07.24

Contents

Introduction

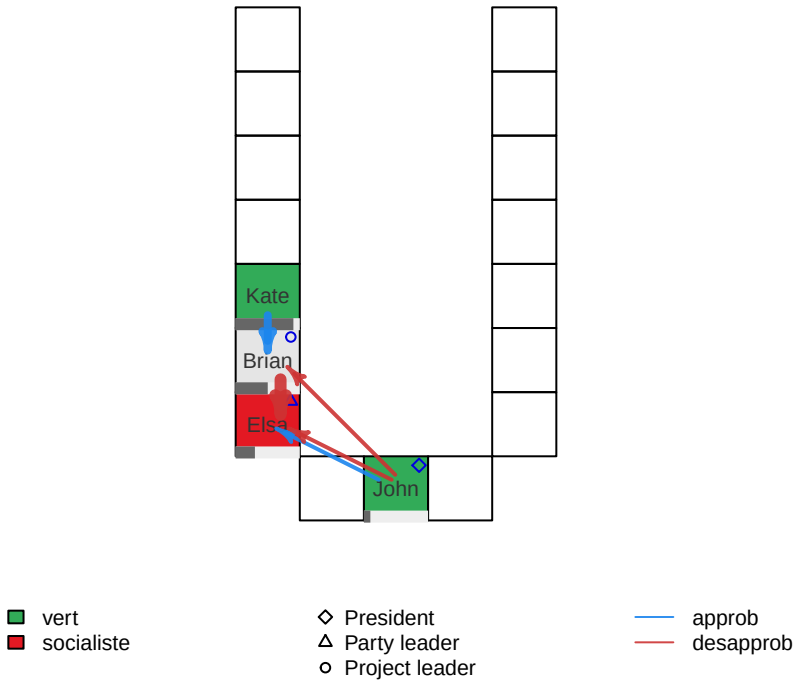
Social networks analysis has received special attention over the past decade, and a lot of tools for manipulating and rendering social networks have emerged. In several situations a social network is associated with a spatial dimension, and behaviors observed within the network cannot be interpreted without taking into account the location of each of its nodes regarding to the other nodes. This is the case, for example, when studying inflows/outflows between cities or companies, or when studying people debating in a room. Based on the `sp` package, which provides efficient classes for storing spatial data and methods for handling and rendering them, the `spnet` package aims at facilitating the rendering of (social) networks on maps. Furthermore, fixing network node positions allows to more easily monitor time-varying networks and observe how connections and flows evolve over time.

The `spnet` package offers methods for dealing with spacial social networks. It allows to plot networks for which actors have a specific location on a map (participants in a political debate, cities, etc.). `SpatialPolygons` objects from the `sp` package are supported.

Gallery

The `spnet.example.basic` function provides a working example involving basic functionnalities of the `spnet` package. Don't hesitate to look at its code and take what you need.

```
net1 <- spnet.example.basic()
plot(net1)
```



Usage

Creating a `spnet` object

Creating a new `spnet` object (formal class `SpatialNetwork`) is done by the `spnet.create` function. You will have to supply two required parameters: the nodes ID, and the corresponding positions on the map (which

will be set thereafter). Note that in a `spnet` object data are stored in a `data.frame`. In this `data.frame` the two required parameters will respectively supplied as variables `'NODE'` and `'POSITION'`. Here is an example:

```
node <- c("John", "Elsa", "Brian", "Kate")
position <- c(2,4,6,8)

net1 <- spnet.create(
  data.frame(
    'NODE' =  node,
    'POSITION' = position
  )
)

class(net1)
```

```
## [1] "SpatialNetwork"
## attr(,"package")
## [1] "spnet"
```

Setting a map

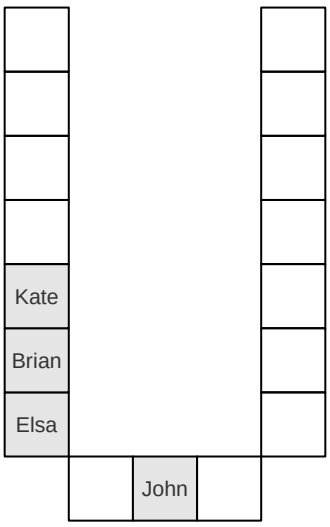
Setting labels

Nodes IDs have to be valid variable names. You should avoid special character or spaces in Nodes IDs. Indeed, currently network data have to be provide as a `matrix`, with node IDs in row names an column names. Thus, the `spnet` object will automatically do the match between your network data and the node positions on the map. To be valid row and column names, node IDs must not contain special character or space.

You may need to have more suitable label for your nodes when plotting the `spnet` object, as for instance containing spaces or accented characters. Here is an example :

First, we start with our basic `spnet` object containing a simple room map:

```
net1 <- spnet.example.basic.map()
plot(net1)
```



This example contains the following data:

```r
data.frame(net1)
```

```
##    NODE POSITION
## 1  John        2
## 2  Elsa        4
## 3 Brian        6
## 4  Kate        8
```

```r
net1$label <- c("John A.", "Elsa B.", "Brian C.", "Kate D.")
spnet.label.variable(net1) <- 'label'

plot(net1)
```

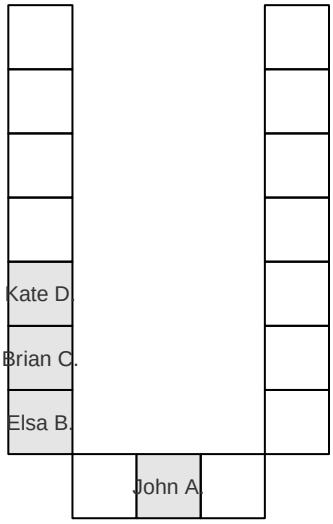


Figure 1: plot of chunk unnamed-chunk-6

You can change the size and the color of labels:

```r
spnet.label.cex(net1) <- 0.7
spnet.label.color(net1) <- '#FF0000'

plot(net1)
```

If you want to disable label printing, you can add an empty variable and use it as the labelling variable:

```r
net1$label.empty <- rep("", nrow(net1))
spnet.label.variable(net1) = 'label.empty'

plot(net1)
```

Setting colors

To set colors you basically need:

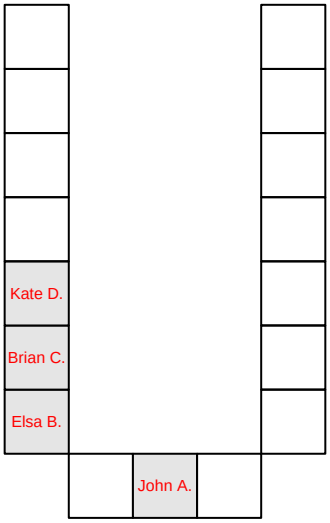


Figure 2: plot of chunk unnamed-chunk-7

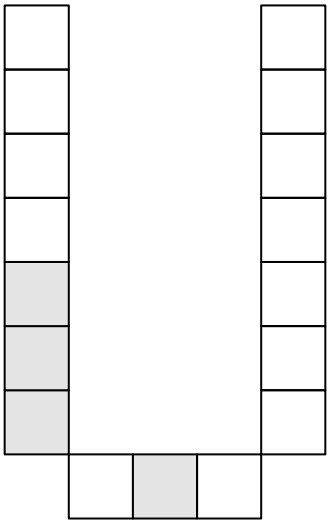
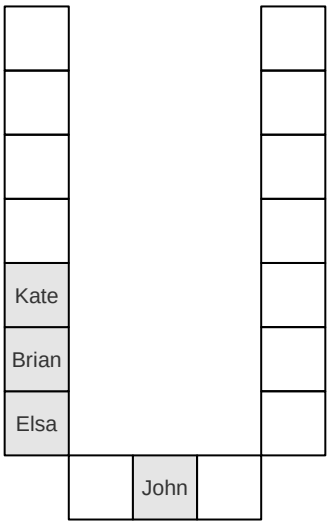


Figure 3: plot of chunk unnamed-chunk-8

- A categorical variable affecting each node to a class
- a legend of color specifying the color to use for each class

Here is a practical example. First, we start with our basic `spnet` object containing a map.

```
net1 <- spnet.example.basic.map()
plot(net1)
```



We add a categorical variable affecting each node to a class:

```
net1$parti <- c('vert', 'socialiste', 'autre', 'vert')
```

Data are now:

```
data.frame(net1)
```

```
##    NODE POSITION      parti
## 1  John        2       vert
## 2  Elsa        4 socialiste
## 3 Brian        6      autre
## 4  Kate        8       vert
```

Then we specify we want to use the variable `parti` to colorize the map:

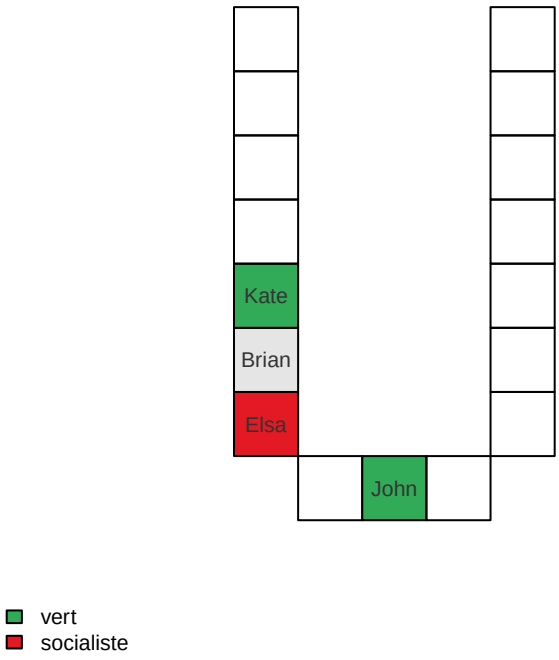
```
spnet.color.variable(net1) <- "parti"
```

Finally we specify the colors to use:

```
spnet.color.legend(net1) <- c('vert' = "#32AB58", 'socialiste' = "#E31923")
```

Now the `plot` function is able to colorize the graphic:

```
plot(net1)
```



Dealing with a quantitative covariate: rendering individual barplots

We may need to render a quantitative attribute related to each node of the network. To that purpose we provide a simple barplot tool. This section details how to use it. We start with a fresh `spnet` object and equipe it with a map.

```
ex.bp <- spnet.example.basic.map()
```

A fresh `spnet` object contains the following default barplot settings:

```
spnet.barplot.list(ex.bp)
```

```
## $variable
## [1] ""
##
## $bound.lower
## [1] -0.5 -0.5
##
## $bound.upper
## [1]  0.5 -0.5
##
## $fgcolor
## [1] "#666666"
##
## $bgcolor
## [1] "#eeeeee"
##
## $width
## [1] 8
```

The first point is to

```
ex.bp$content <- c(0.1,0.3,0.5,0.9)
ex.bp
```

```
## This is a valid 'SpatialNetwork' object.
##
## - Data: (first rows)
##
##    NODE POSITION content
## 1  John        2     0.1
## 2  Elsa        4     0.3
## 3 Brian        6     0.5
## 4  Kate        8     0.9
##
## - Map:
##     Length: 17
##
## - Plotting options:
```

```
spnet.barplot.variable(ex.bp) <- "content"
spnet.barplot.list(ex.bp)
```

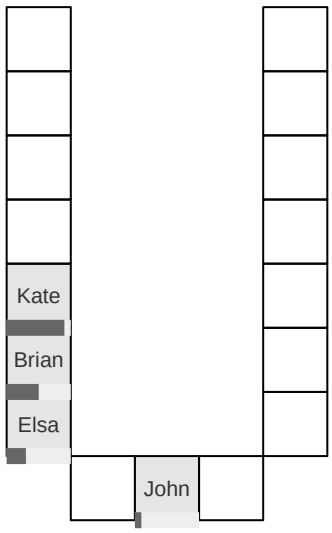
```
## $variable
## [1] "content"
##
## $bound.lower
## [1] -0.5 -0.5
##
## $bound.upper
## [1]  0.5 -0.5
##
## $fgcolor
## [1] "#666666"
##
## $bgcolor
## [1] "#eeeeee"
##
## $width
## [1] 8
```

```
ex.bp
```

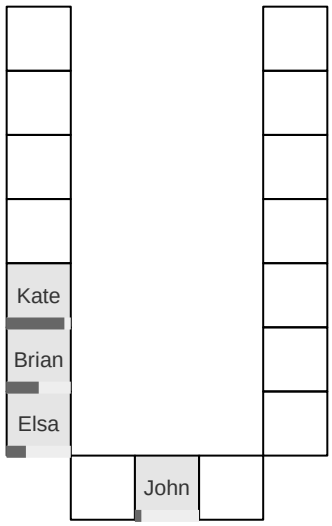
```
## This is a valid 'SpatialNetwork' object.
##
## - Data: (first rows)
##
##    NODE POSITION content
## 1  John        2     0.1
## 2  Elsa        4     0.3
## 3 Brian        6     0.5
## 4  Kate        8     0.9
```

```
##
## - Map:
##     Length: 17
##
## - Plotting options:
##     Variable used to draw barplots: 'content'
```

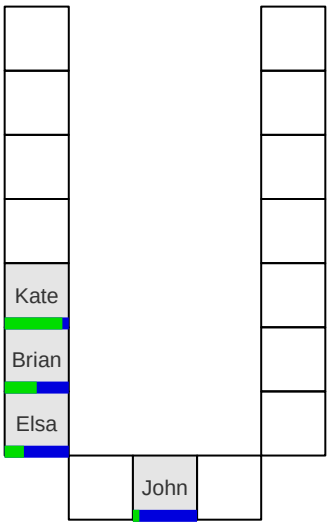
```
plot(ex.bp)
```




```
spnet.barplot.bound.lower(ex.bp) <- c(-0.5,-0.44)
spnet.barplot.bound.upper(ex.bp) <- c(0.5,-0.44)
spnet.barplot.width(ex.bp) <- 6
plot(ex.bp)
```




```
spnet.barplot.fgcolor(ex.bp) <- "#00dd00"
spnet.barplot.bgcolor(ex.bp) <- "#0000dd"
plot(ex.bp)
```



Maps

SpatialPolygons maps

Rooms

The easiest way to create a room to represent a debate is with the `room.create.grid` function. Here is an example of use:

```
col <- 5
row <- 6
m <- matrix(rep(-1, col*row), nrow = row)
m[1,2:4] <- 0
m[3,c(1,5)] <- 0
m[4,c(1,5)] <- 0
m[5,c(1,5)] <- 0
m[6,c(1,5)] <- 0
m
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   -1    0    0    0   -1
## [2,]   -1   -1   -1   -1   -1
## [3,]    0   -1   -1   -1    0
## [4,]    0   -1   -1   -1    0
## [5,]    0   -1   -1   -1    0
## [6,]    0   -1   -1   -1    0
```

```
room1 <- room.create.grid(m, seat.width=2, seat.height=1)
spnet.map.plot.position(room1)
```

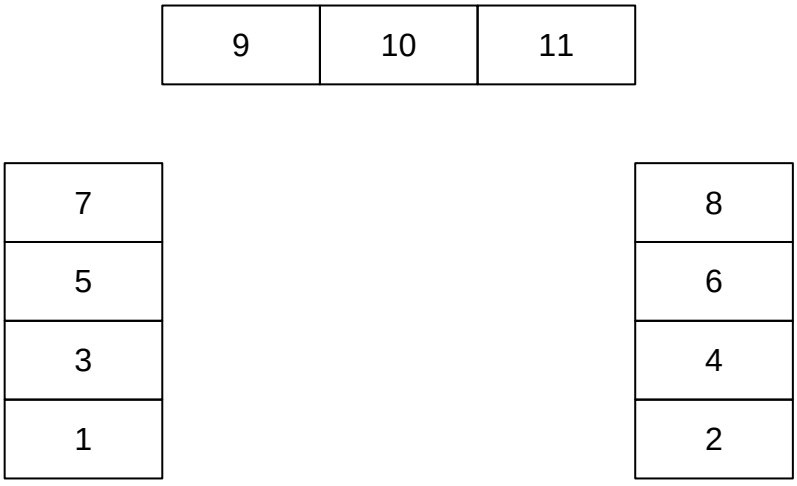


Figure 4: A simple room with table in invered ‘U’ form