Overview of the spnet package

Emmanuel Rousseaux, Marion Deville and Gilbert Ritschard

2014.12.11

Contents

Introduction 2
Gallery 2
Usage 2
Creating a `spnet` object 2
Setting a map 4
Setting labels 4
Setting colors 5
Dealing with a quantitative covariate: rendering individual barplots 9

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.

```
global.par.mar <- c(0,0,0,0)
```

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.

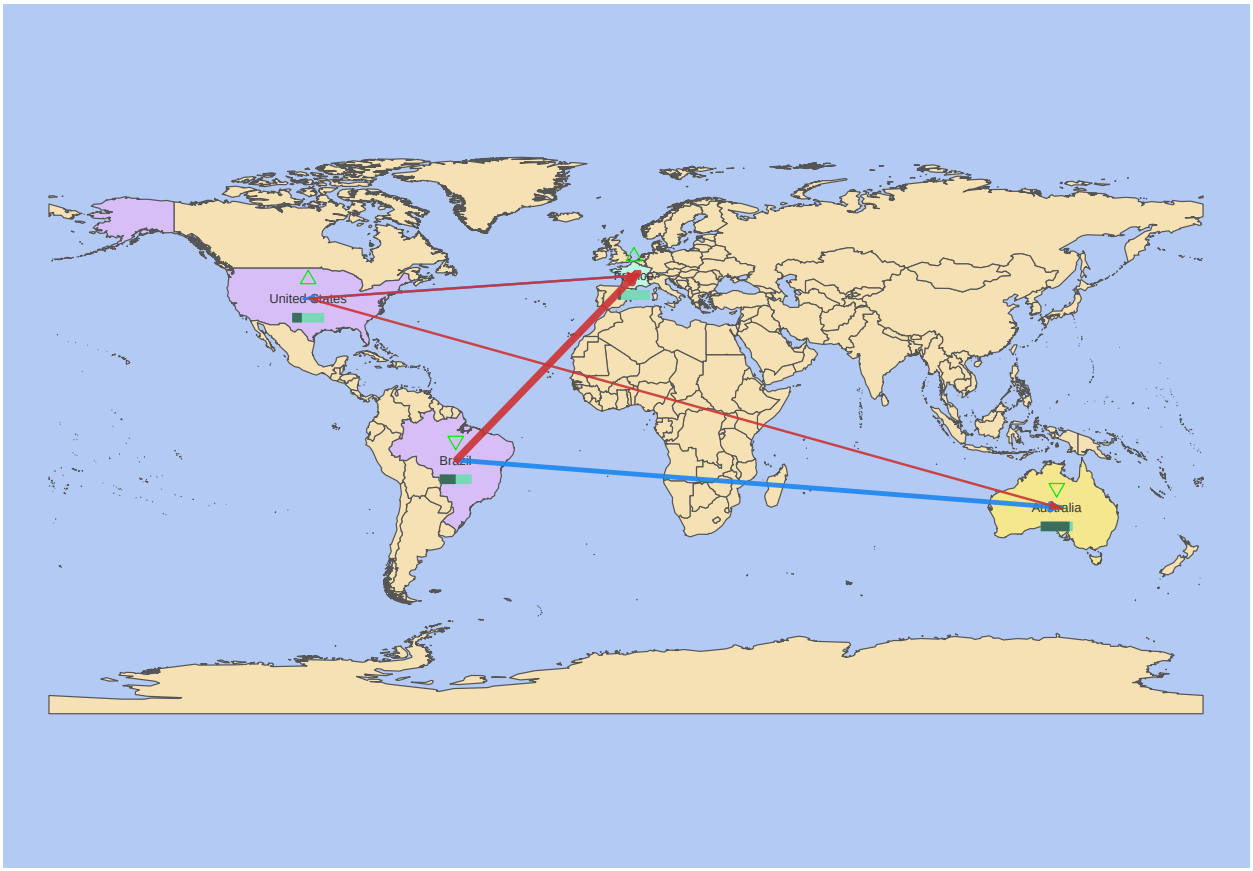
```
net1 <- spnet.example.basic()
plot(net1)
```

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 node IDs, and the corresponding position IDs on the map (which will be set in a second step). 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("France", "United States", "Brazil", "Australia")

# We load the world map
data(world.map.simplified)

# We retrieve position IDs for our nodes
# As countries are numbered from 0 to 245 on the map, we subtract 1
position <- match(node, world.map.simplified@data[,'NAME']) - 1

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


United States
Brazil
Australia
Europa
America
Oceania
North
South
network1
network2

```
class(net1)
```

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

Setting a map

The next step is to define the map to use for plotting. This is done with the `spnet.map` function. Currently only `SpatialPolygons` objects are supported. If your map comes on another format (for example a shape file .shp) you can use the facilities of the `maptools` package to convert your map data into a `SpatialPolygons` object.

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:

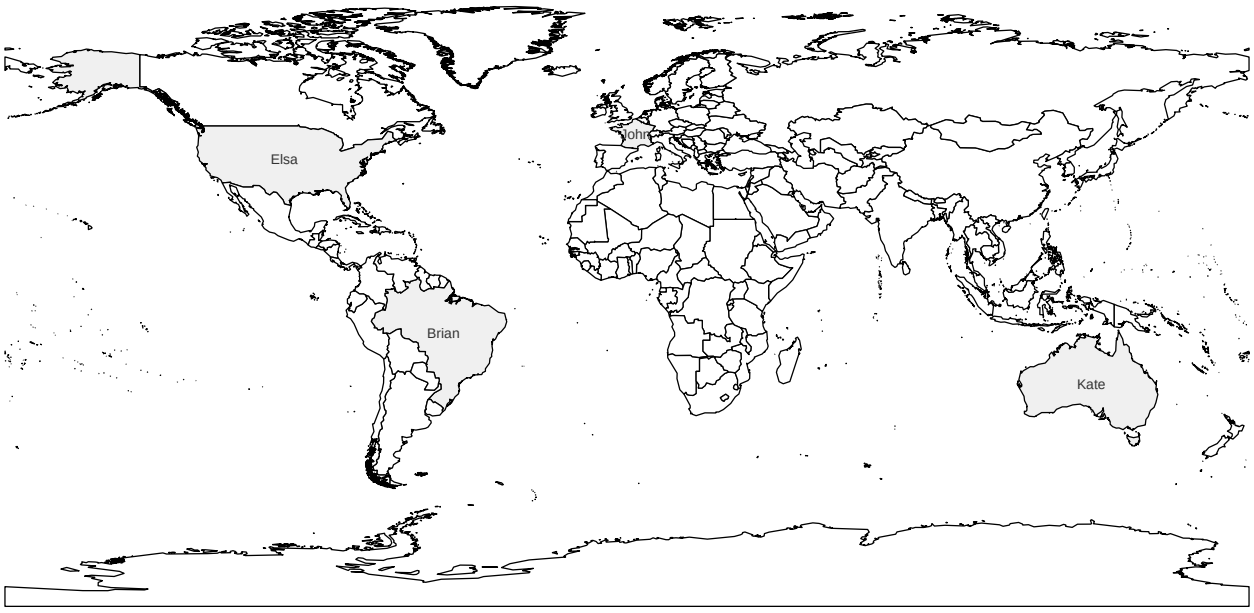
```
net1 <- spnet.example.basic.map()
plot(net1)
```

This example contains the following data:

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

```
##            NODE POSITION
## 1        France       64
## 2 United States      208
## 3        Brazil       20
## 4     Australia        8
```

```
net1$label <- c("John", "Elsa", "Brian", "Kate")
spnet.label.variable(net1) <- 'label'

plot(net1)
```



You can change the size and the color of labels:

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

plot(net1)
```

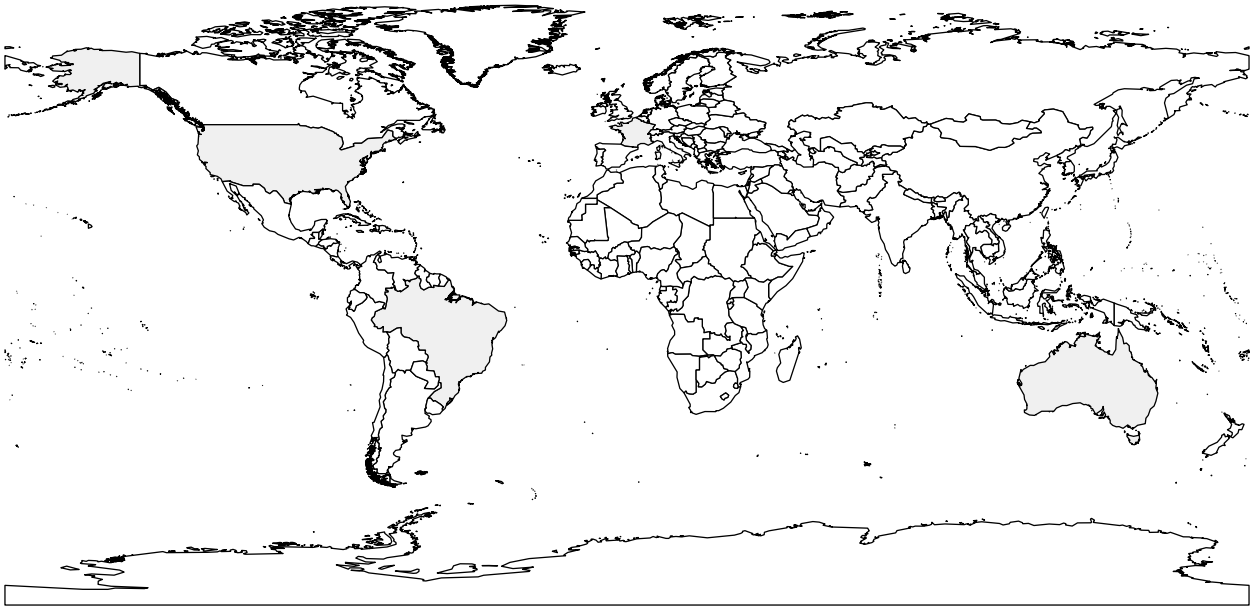
If you want to disable label printing, you can add an empty variable and use it as the labelling variable:

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

plot(net1)
```

Setting colors

To set colors you basically need:

John
Elsa
Brian
Kate



- 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)
```

To make the graphic nicer, we can set set the backround color, border color, and a default color for colorizing the regions:

```
spnet.color.background(net1) <- "#B3CAF5" # light blue
spnet.color.border(net1) <- "#555555" # grey
spnet.color.region(net1) <- "#F5E1B3" # light kaki
plot(net1)
```

Now, we add a categorical variable affecting each node to a class:

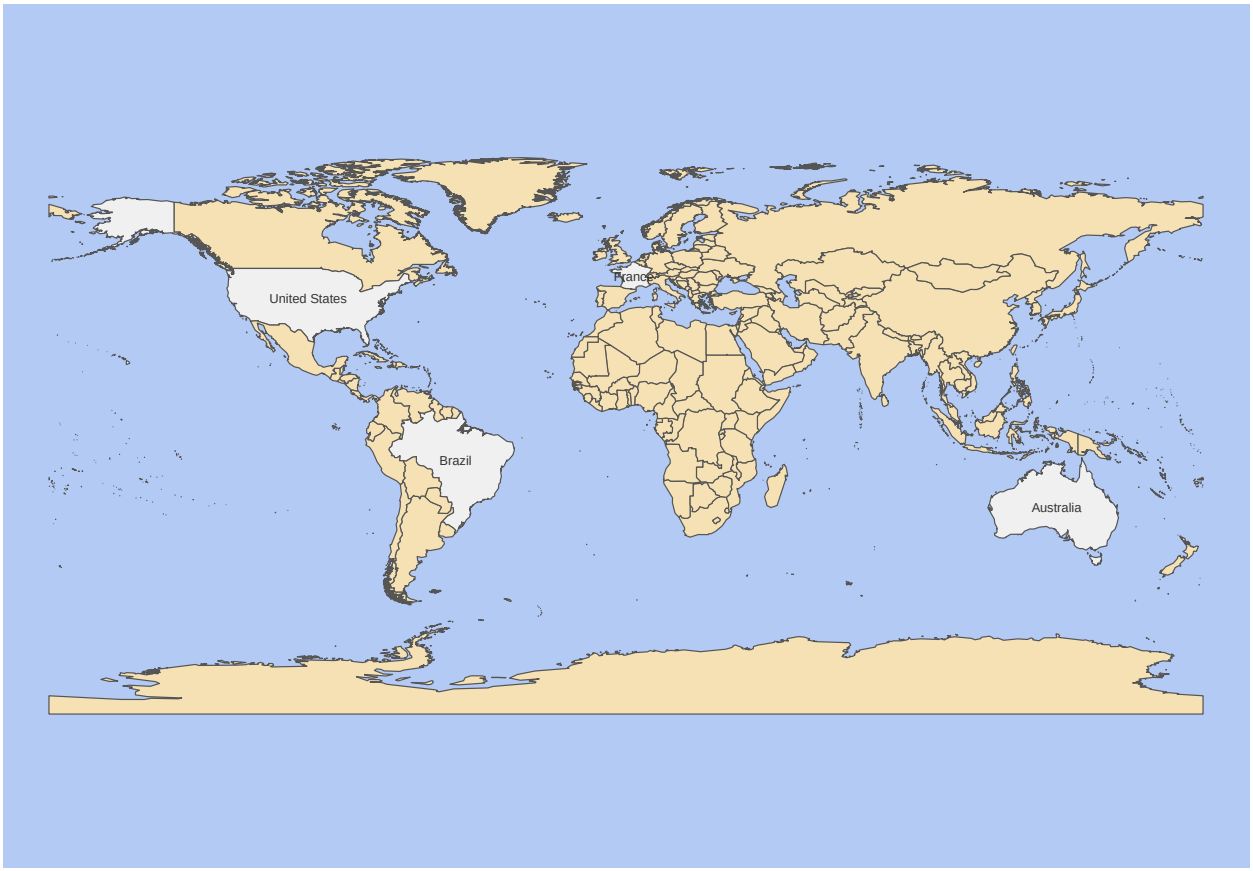
```
net1$continent <- c("Europa", "America", "America", "Oceania")
```

Data are now:

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

```
##            NODE POSITION continent
## 1        France       64    Europa
## 2 United States      208   America
## 3        Brazil       20   America
## 4     Australia        8   Oceania
```

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


United States
France
Brazil
Australia

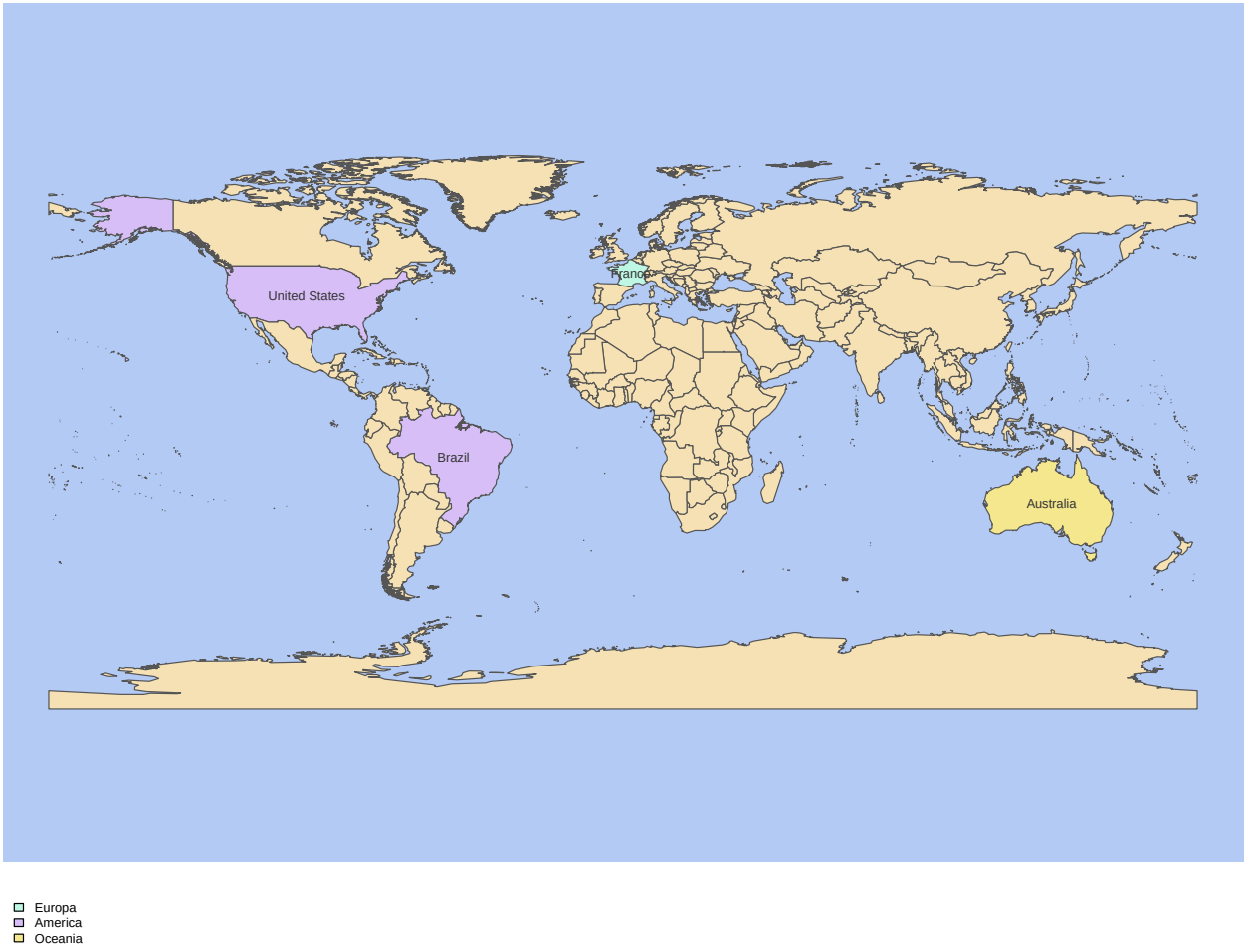
```
spnet.color.variable(net1) <- "continent"
```

Finally we specify the colors to use:

```
spnet.color.legend(net1) <- c('Europa' = "#BEF7E3", 'America' = "#D7BEF7", 'Oceania' = "#F5E78E")
```

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 define a variable to fill in the individual barplot. This variable can be defined manually in a vector, or imported. The value has to be normalized between 0 and 1.

```
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        France       64     0.1
## 2 United States      208     0.3
## 3        Brazil       20     0.5
## 4     Australia        8     0.9
##
## - Map:
##     Length: 246
##
## - Plotting options:
```

The 'content' variable is added into the database.

The second step is to define the barplot variable. This operation is confirmed on the first line of the spnet.barplot.list().

```
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        France       64     0.1
## 2 United States      208     0.3
## 3        Brazil       20     0.5
## 4     Australia        8     0.9
##
## - Map:
##     Length: 246
##
## - Plotting options:
##     Variable used to draw barplots: 'content'
```

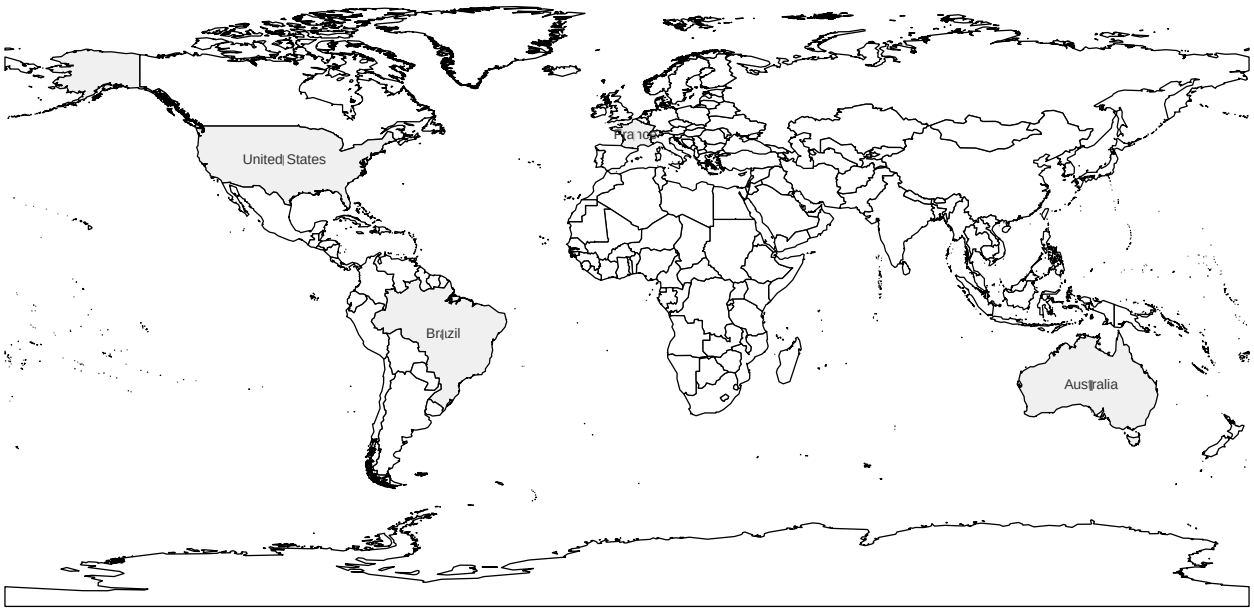
```
plot(ex.bp)
```

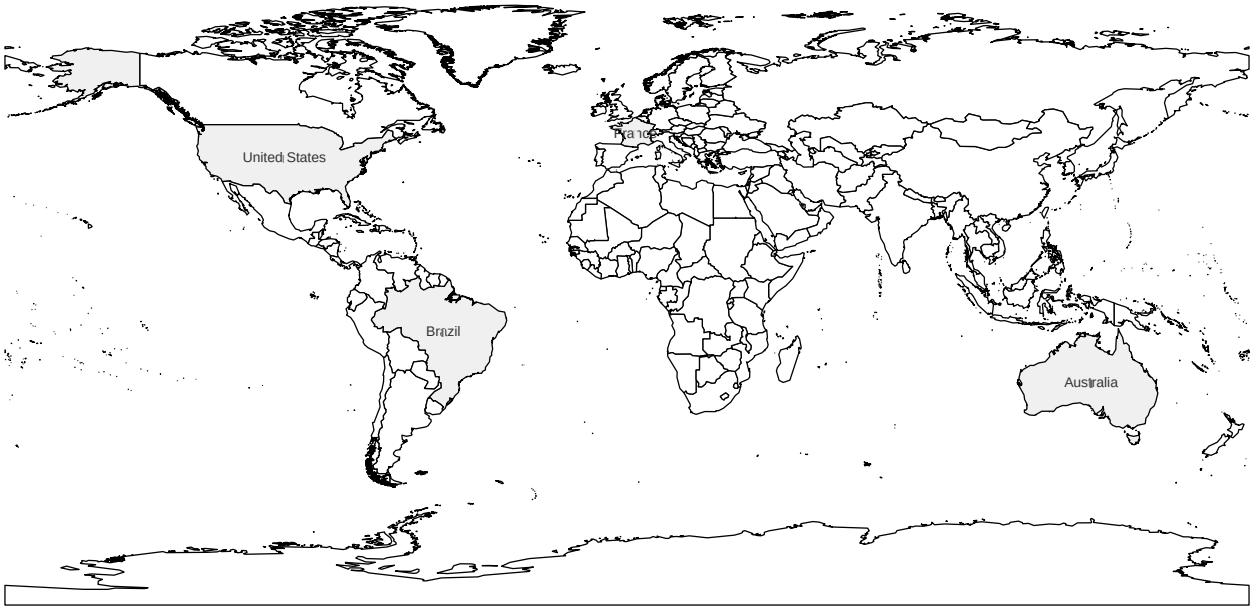
The plot is by default in black and white. On the graphs individual barplot appear at the bottom of the boxes. To pull them up a little, use the functions `spnet.barplot.bound.lower` and `spnet.barplot.bound.uper` to adjust.

```
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)
```

To look at the wide range of function `spnet` package offers, let get into a very simple example:


United States
Brazil
Australia

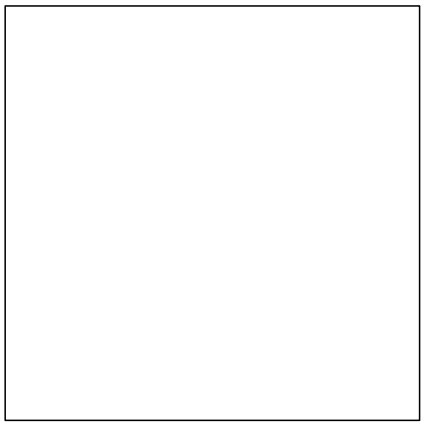

United States
Brazil
Australia

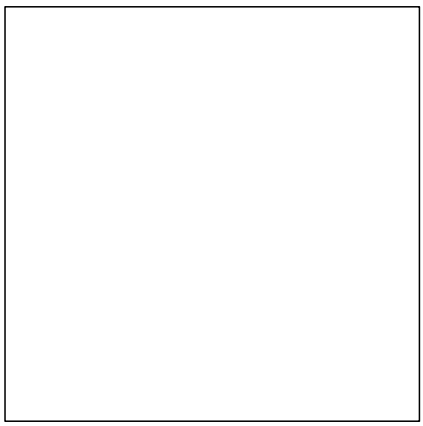
```
col <- 3
row <- 3

m3 <- matrix(rep(-1, col*row), nrow=row)

m3[1,1] <- 1
m3[3,3] <- 2

m3
```

```
##      [,1] [,2] [,3]
## [1,]    1   -1   -1
## [2,]   -1   -1   -1
## [3,]   -1   -1    2
```

Then we convert the `matrix` object into a `SpatialPolygon` object

```
example <- room.create.grid(m3, seat.width=0.5, seat.height=0.5)
plot(example)
```



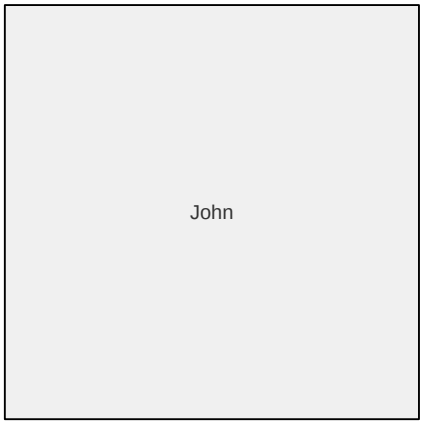


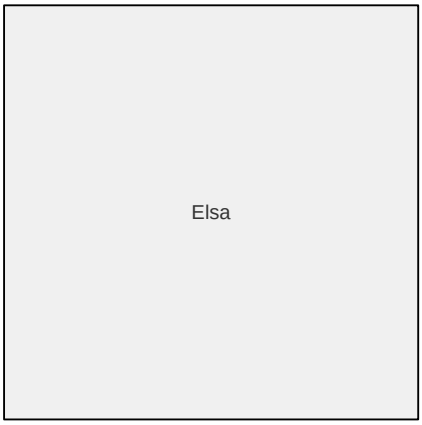
We have drawn a room of two boxes.

```
node <- c('John', 'Elsa')
position <- c(1,2)
netexample <- spnet.create(data.frame(
  'NODE' = node,
  'POSITION' = position
  ))
spnet.map(netexample) <- example
plot(netexample)
```

Creating the spnet object

Untitled SPNET object





```
spnet.title.main(netexample) <- "Congress debate"
spnet.title.sub(netexample) <- "Climate change"
plot(netexample)
```

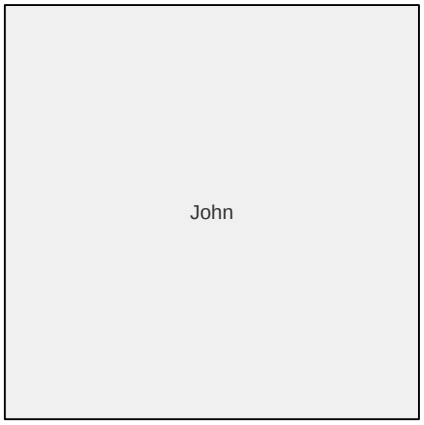
Putting title and subtitle on the graph No title:

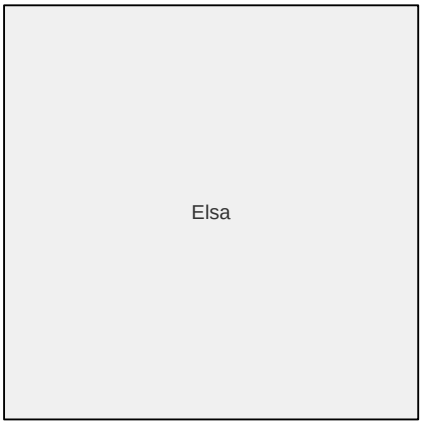
```
spnet.title.main(netexample) <- ""
spnet.title.sub(netexample) <- ""
plot(netexample)
```

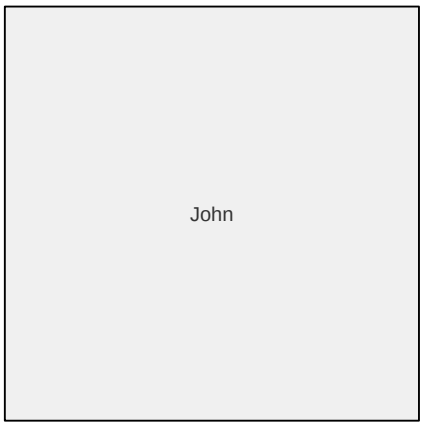
```
netexample$parti <- c('Democrate', 'Republican')
spnet.color.variable(netexample) <- 'parti'
spnet.color.legend(netexample) <- c(
  'Democrate' = "#DE869C",
  'Republican' = "#868DDE")
plot(netexample)
```

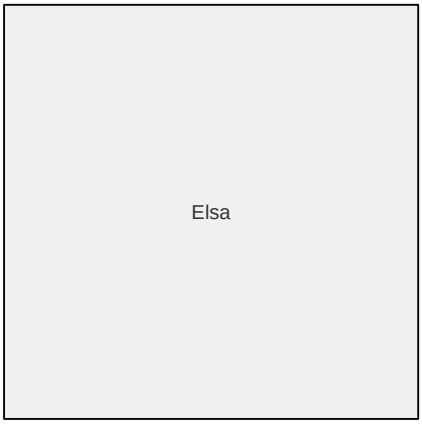
Congress debate

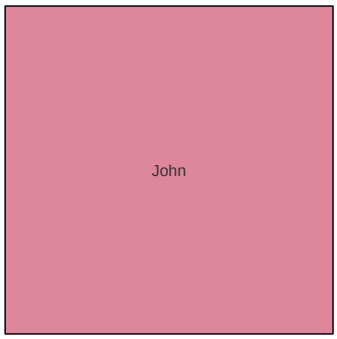
Climate change

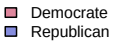





John


Elsa





Setting colors according to affiliations
ColorPicker.com gives a large choice of colors in Hex Color.