Overview of the spnet package

Emmanuel Rousseaux, Marion Deville and Gilbert Ritschard

2015.01.23

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.

```
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)
```

Tips

- By default, arrows start and end in the centre of gravity of areas. If you use labels, this behaviour can make labels unreadable. To fix this you can (1) shorten arrows with `spnet.network.arrow.shorten` or/and (2) shift on x and y axis with respectively `spnet.network.arrow.shift.x` and `spnet.network.arrow.shift.y`.

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 be supplied as variables `'NODE'` and `'POSITION'`. Here is an example:

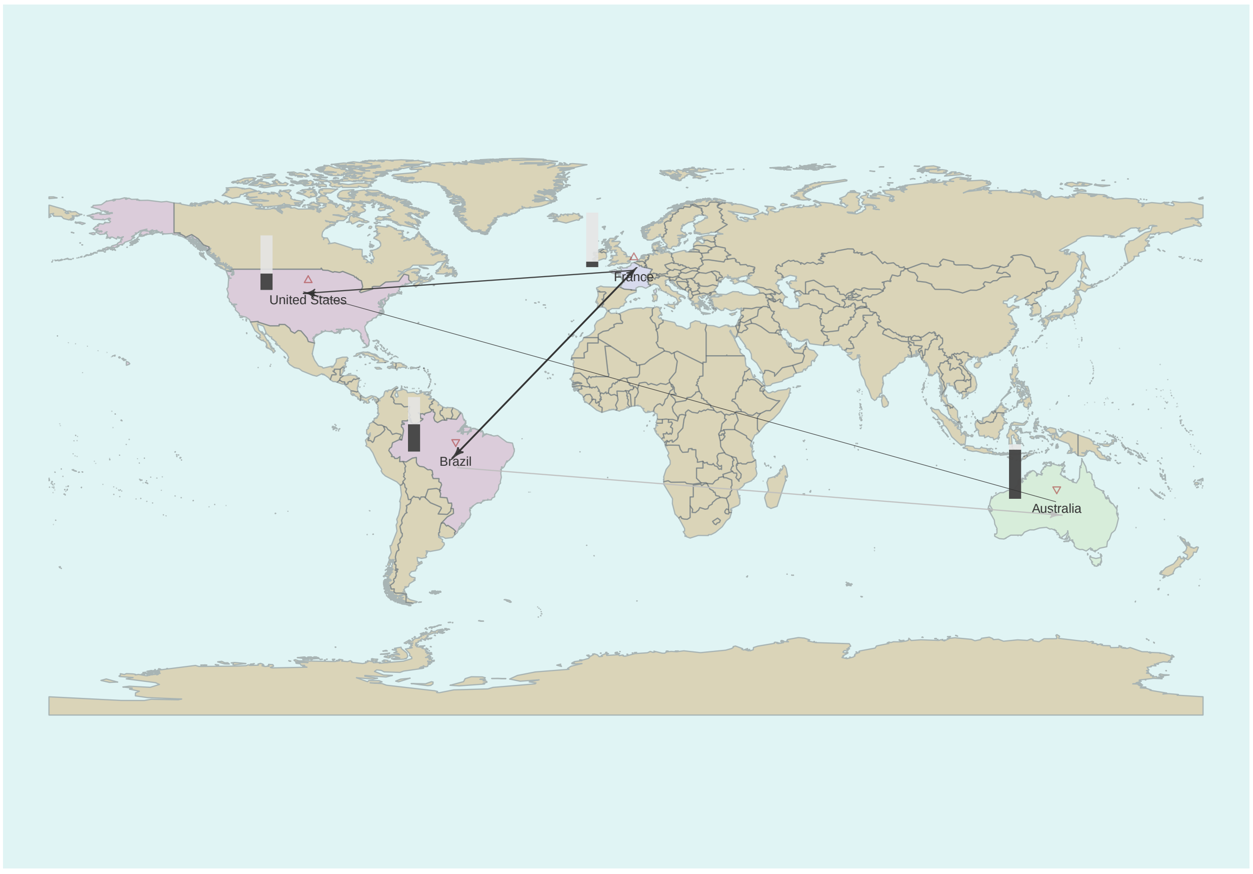
```
node <- c("France", "United States", "Brazil", "Australia")

# 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
```

Places visited by John, Elsa, Brian and Kate

For holidays and studies



□ Europa
□ America
□ Oceania

△ North
▽ South

— Holidays
— Studies

```r
position <- match(node, world.map.simplified@data[,'NAME']) - 1

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

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 (e. g. a shape file .shp) you can use instead the facilities of the `maptools` package to convert your map data into a `SpatialPolygons` object.

Setting labels

The IDs for nodes have to be valid variable names. You should avoid special character or spaces. 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 match your network data with the node positions on the map. To be valid row and column names, node IDs must not contain special characters or spaces.

In case you need to have more suitable label for your nodes when plotting the `spnet` object (e.g. containing spaces or accented characters), here is the protocol to follow:

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

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

Then we introduce the data. In this case we use the following frame/matrix as an example:

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

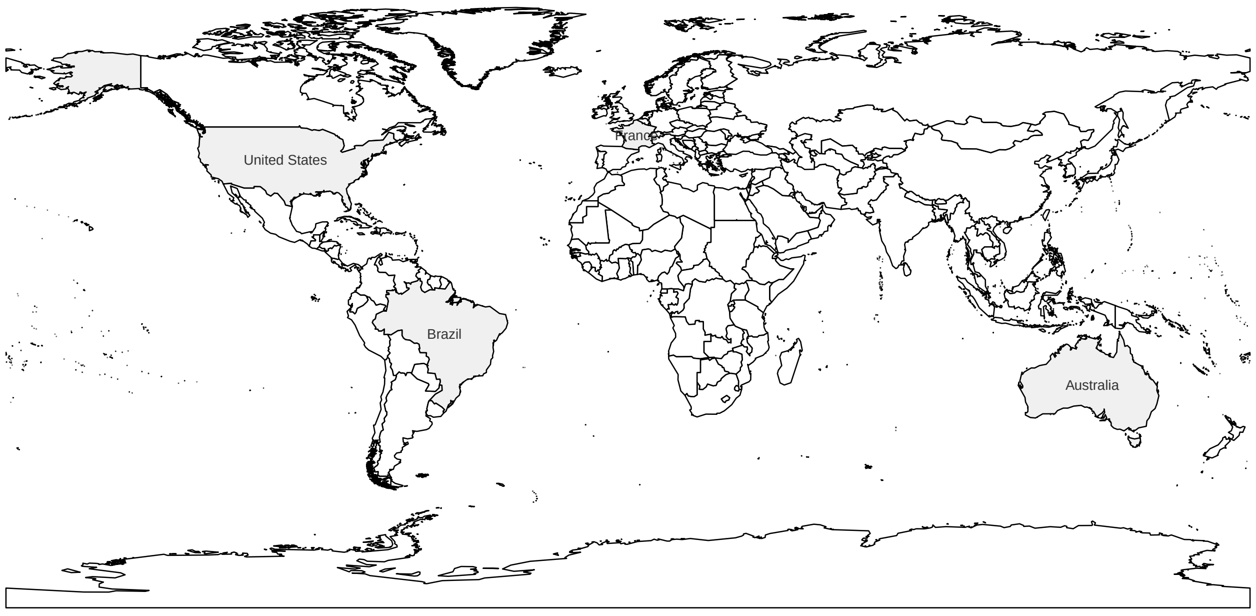
```
##            NODE POSITION
## 1        France       64
## 2 United States      208
## 3        Brazil       20
## 4     Australia        8
```

```r
net1$label <- c("France", "United States", "Brazil", "Australia")
spnet.label.variable(net1) <- 'label'
```

Now you can change the size and the color of labels:

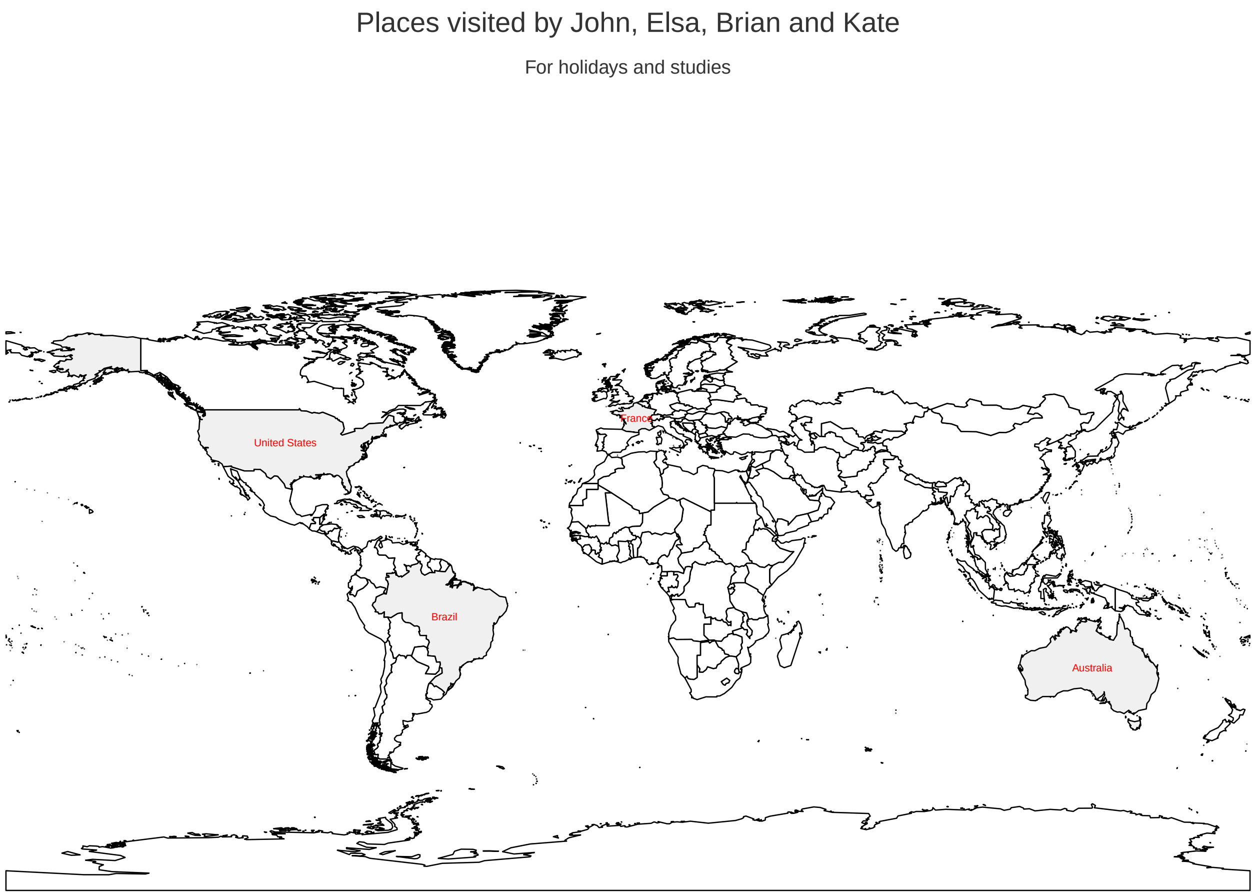
Places visited by John, Elsa, Brian and Kate

For holidays and studies



```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

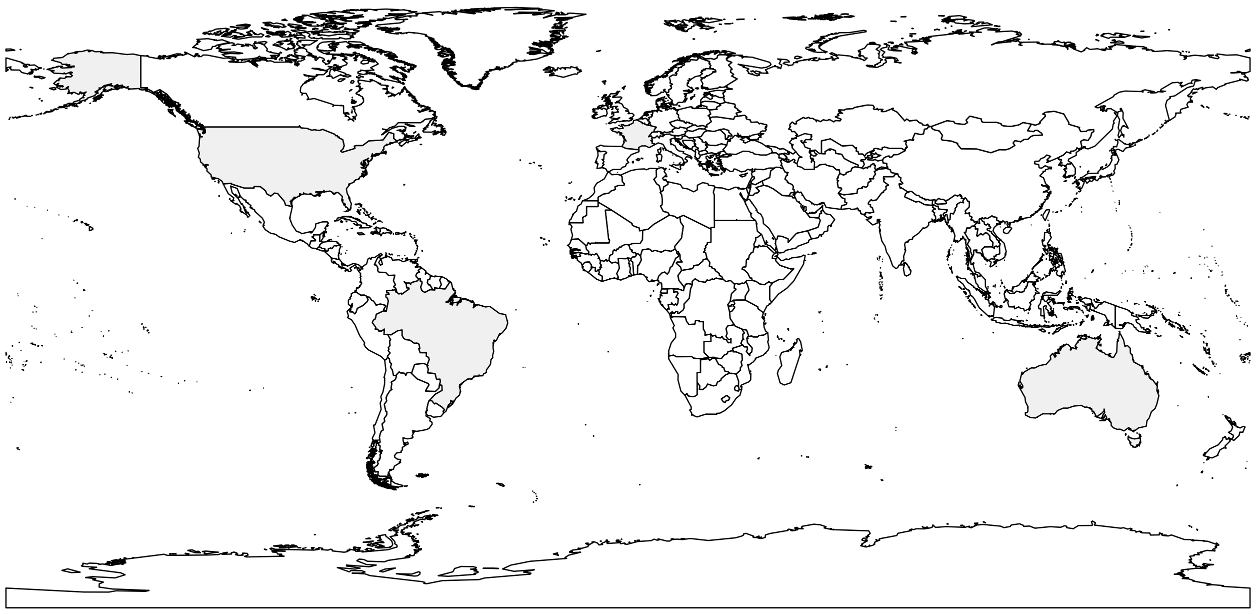
To set colors you need:

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

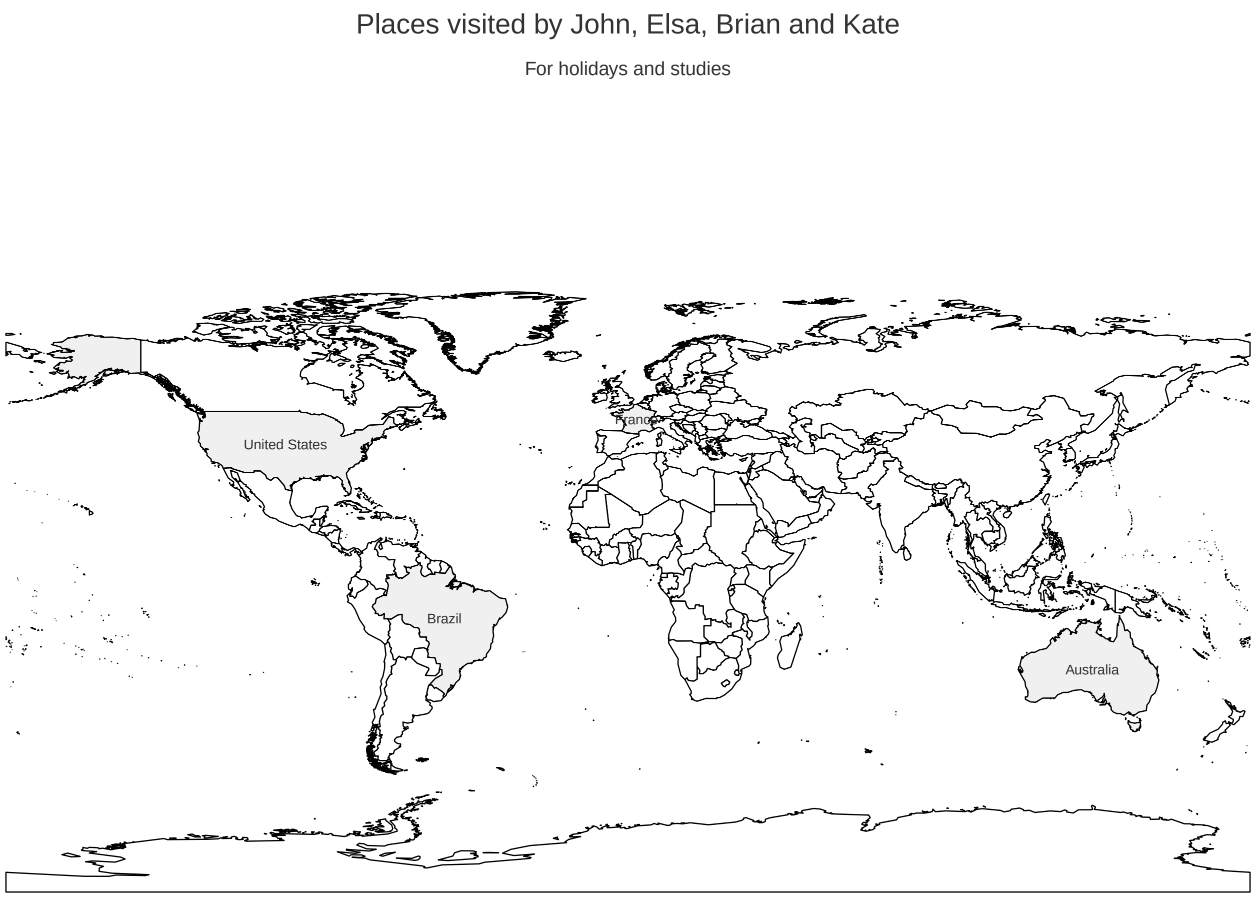
Here is a practical example. First, we start with our basic `spnet` object containing a map.

Places visited by John, Elsa, Brian and Kate

For holidays and studies



```r
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:

```r
spnet.color.background(net1) <- "#B3E4E466" # light blue
spnet.color.border(net1) <- "#55555566" # grey
spnet.color.region(net1) <- "#D2A65F66" # light orange
plot(net1)
```

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

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

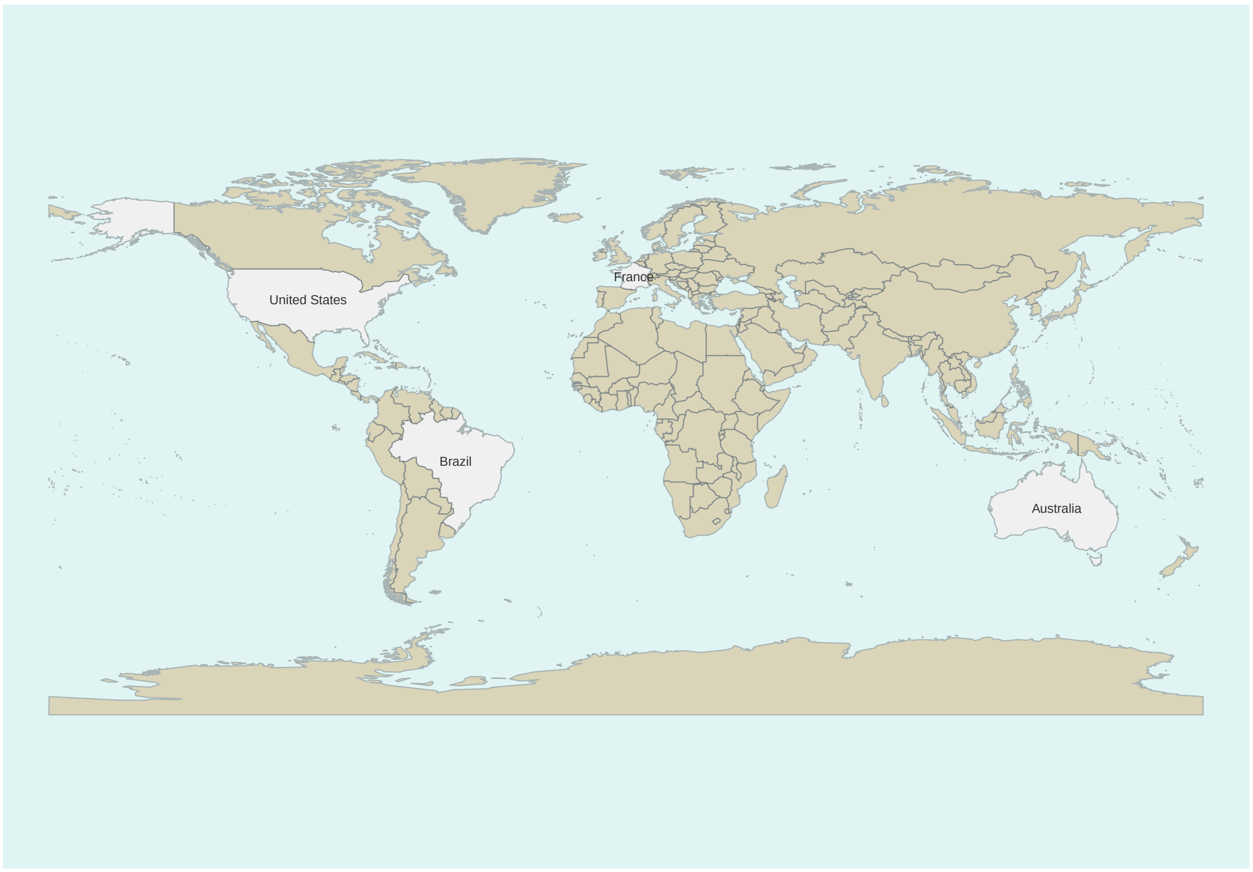
Therefore, the data is changed to:

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

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

Places visited by John, Elsa, Brian and Kate

For holidays and studies



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

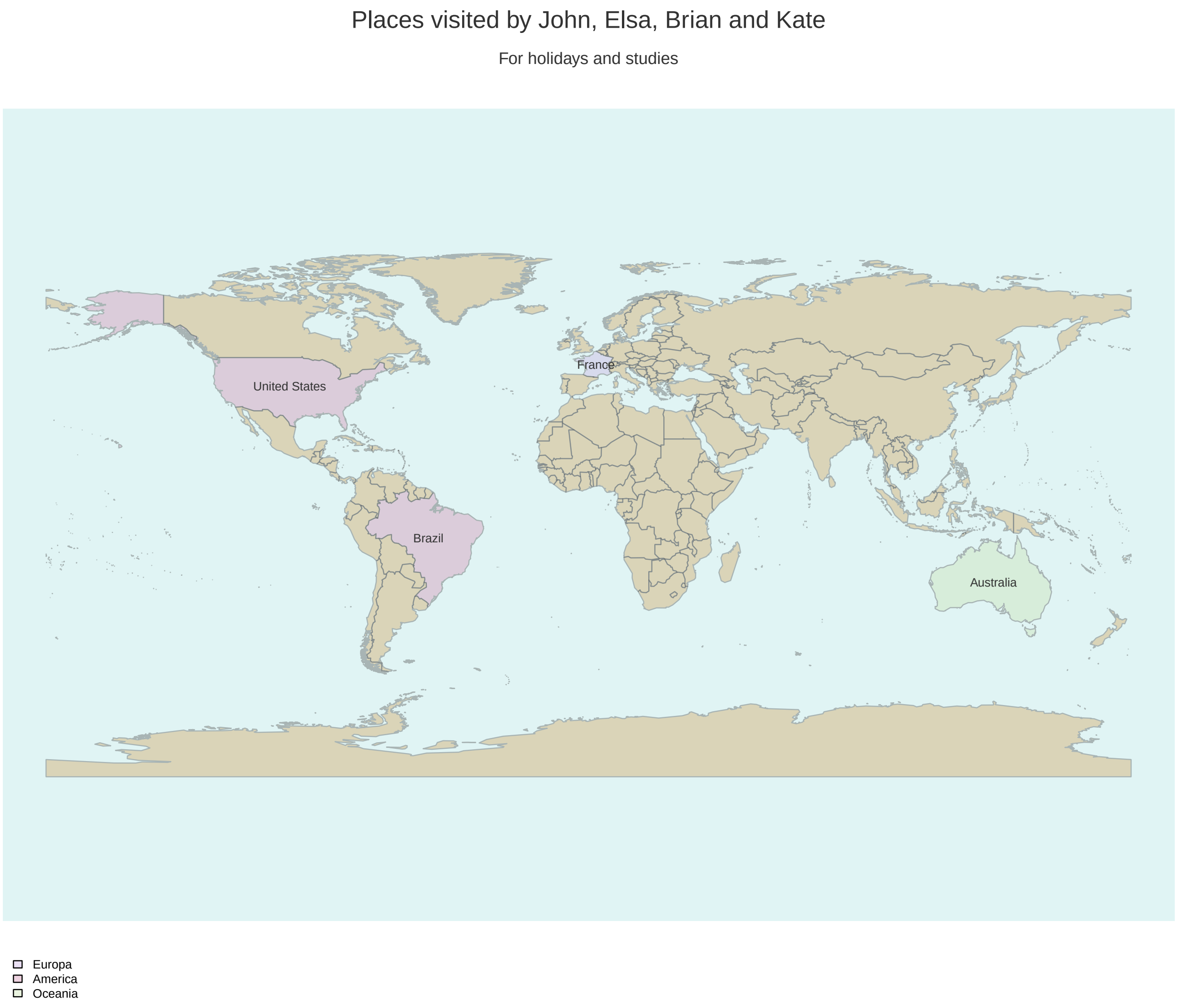
```r
spnet.color.variable(net1) <- "continent"
```

Finally we specify the colors to use:

```r
spnet.color.legend(net1) <- c('Europa' = "#CBB3E466", 'America' = "#D490B366", 'Oceania' = "#CBE4B366")
```

Now the `plot` function is able to colorize the graphics just by running:

```r
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 <- net1
```

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 either 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 continent content
## 1        France       64    Europa     0.1
## 2 United States      208   America     0.3
## 3        Brazil       20   America     0.5
## 4     Australia        8   Oceania     0.9
##
## - Map:
##     Length: 246
##
## - Plotting options:
##     Variable used to colorize: 'continent'
```

Now 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 function spnet.barplot.list().

```r
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
```

```r
ex.bp
```

```
## This is a valid 'SpatialNetwork' object.
##
## - Data: (first rows)
##
##            NODE POSITION continent content
## 1        France       64    Europa     0.1
## 2 United States      208   America     0.3
## 3        Brazil       20   America     0.5
## 4     Australia        8   Oceania     0.9
##
## - Map:
##     Length: 246
##
## - Plotting options:
##     Variable used to colorize: 'continent'
##     Variable used to draw barplots: 'content'
```

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

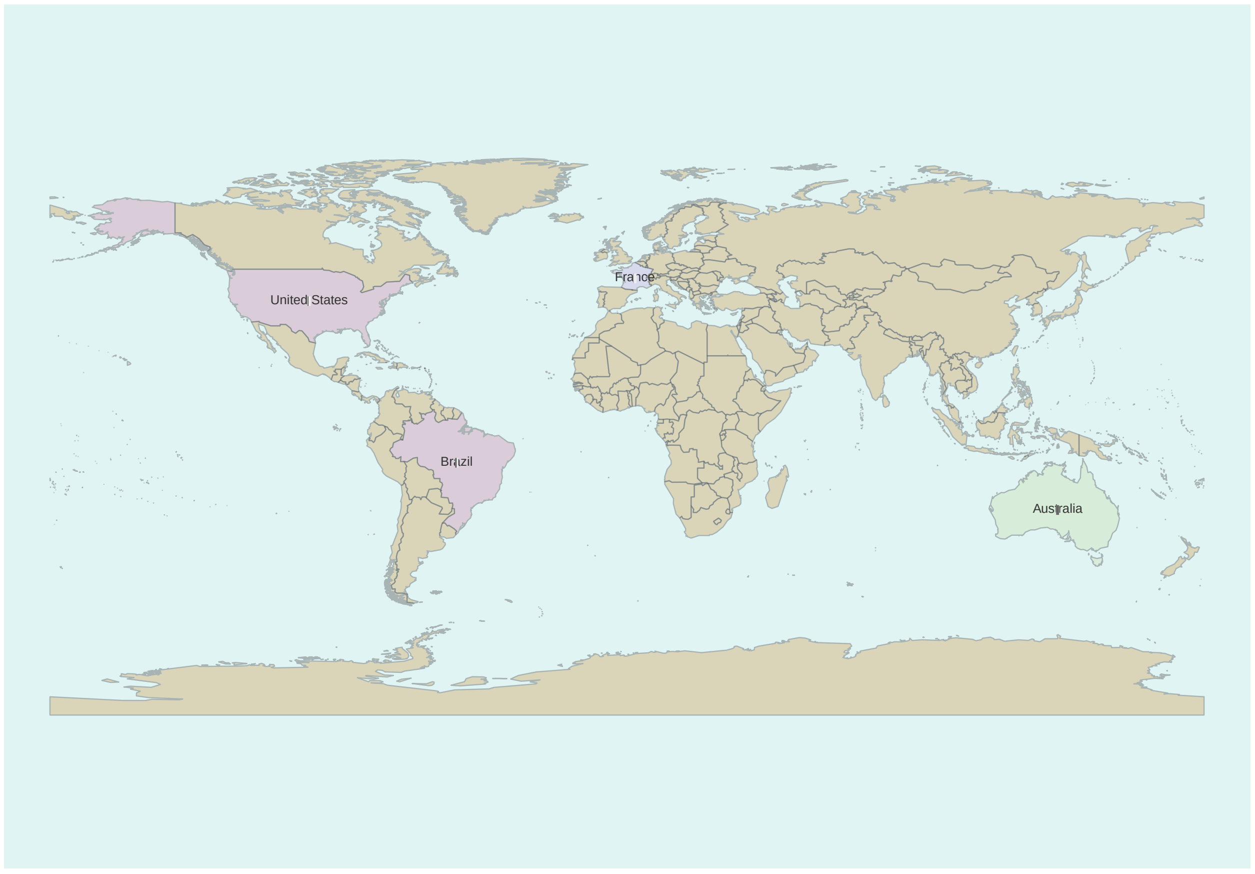
The plot is by default in black and white. On the graphs, individual barplots appear at the bottom of the boxes. To pull them up a little, use the functions `spnet.barplot.bound.lower` and `spnet.barplot.bound.uper` to adjust. The coordinate system is computed from the center of the grid (as for the other fonctions to adjust positions).

```r
spnet.barplot.bound.upper(ex.bp) <- c(-13,20)
spnet.barplot.bound.lower(ex.bp) <- c(-13,3)
plot(ex.bp)
```

To change the colors of the barplot, you can define a fill color `spnet.barplot.fgcolor` and a background color `spnet.barplot.bgcolor`.

Places visited by John, Elsa, Brian and Kate

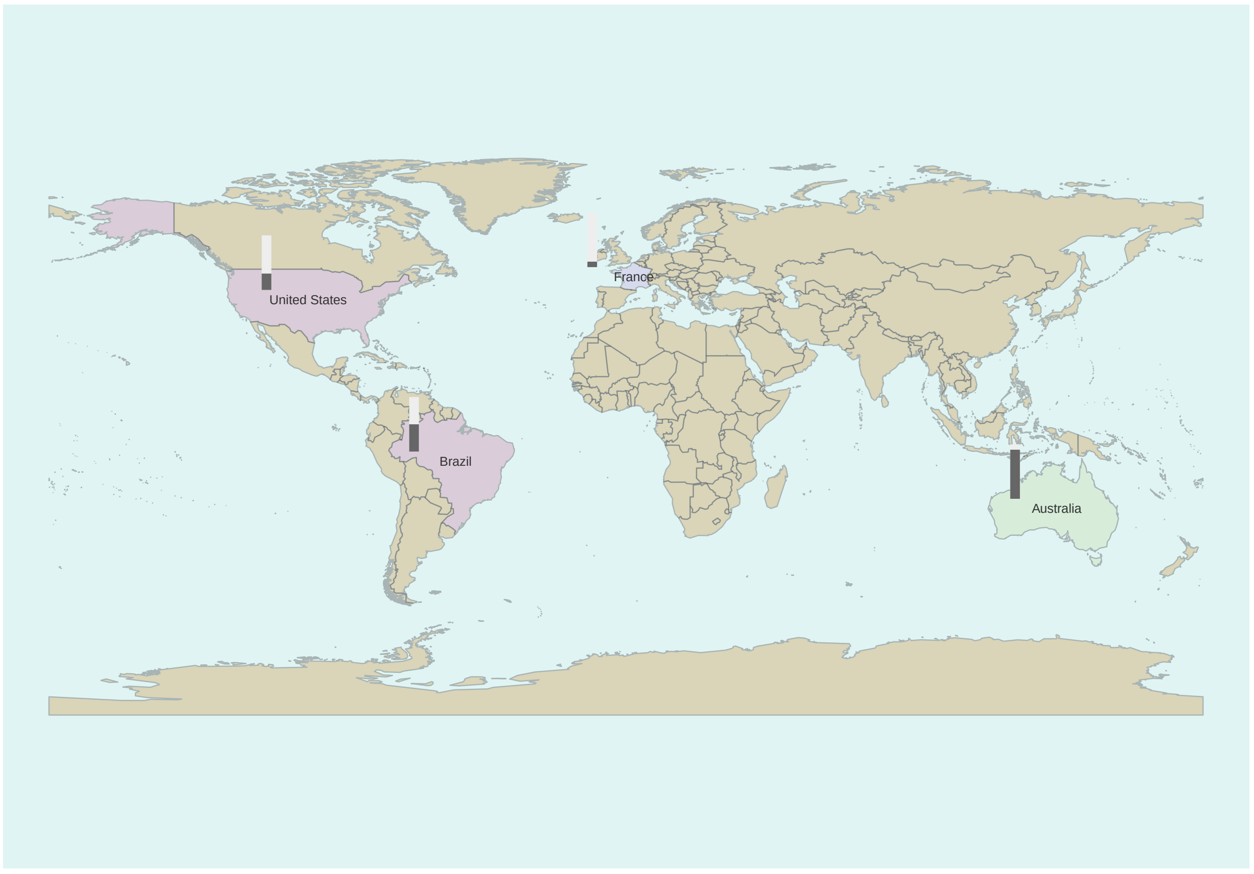
For holidays and studies



□ Europa
□ America
□ Oceania

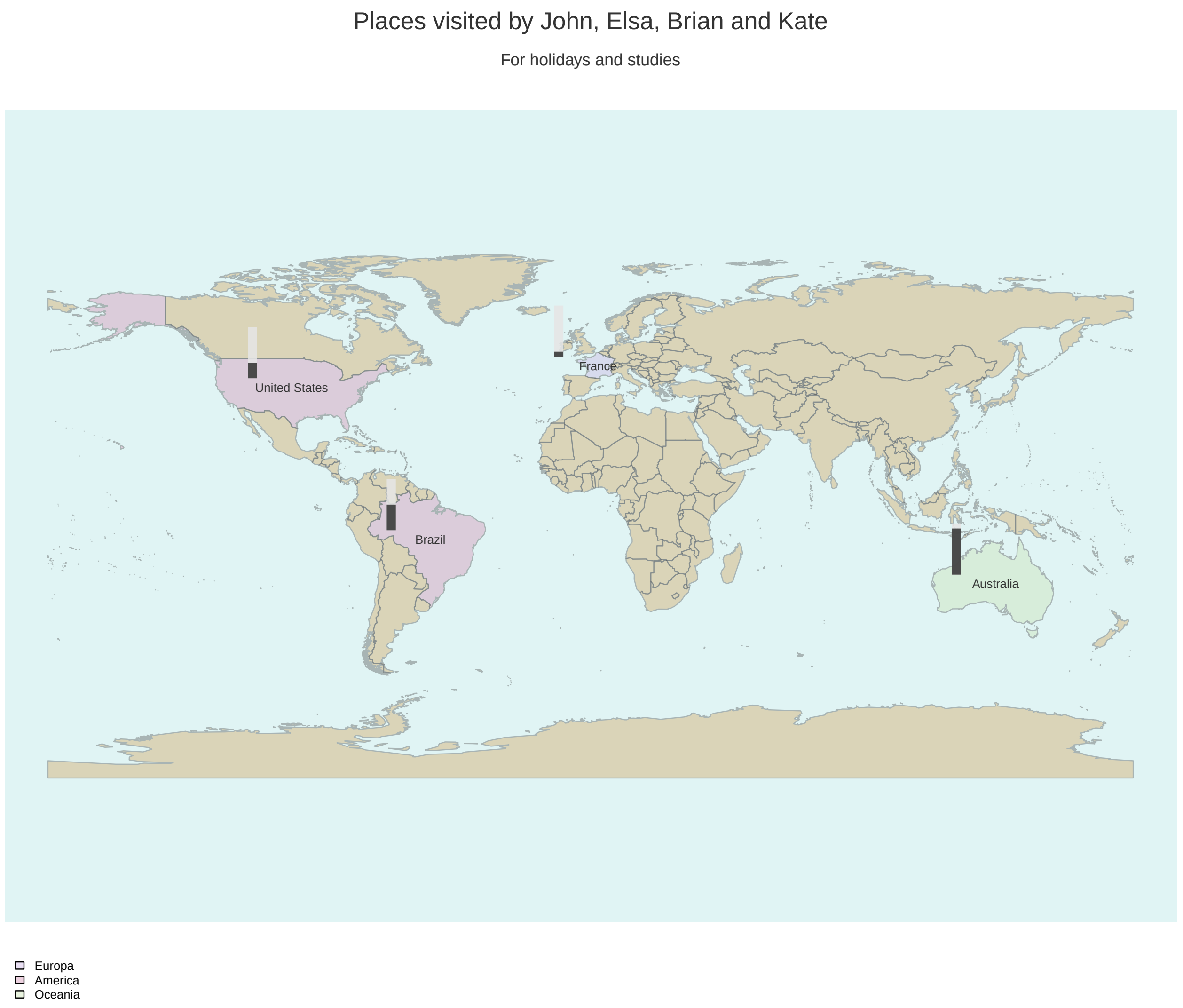
Places visited by John, Elsa, Brian and Kate

For holidays and studies



- ▢ Europa
- ▢ America
- ▢ Oceania

```r
spnet.barplot.fgcolor(ex.bp) <- "#333333DD"
spnet.barplot.bgcolor(ex.bp) <- "#E6E6E6DD"
plot(ex.bp)
```



To find colors, you can go to a web repository such as ColorPicker.com. To determine oppacity, specify a number from 1 (transparent) to 100 (full opacity) at the end of the string.

To change size of the barplot is possible with the `spnet.barplot.width` function:

```r
spnet.barplot.width(ex.bp) <- 10
plot(ex.bp)
```

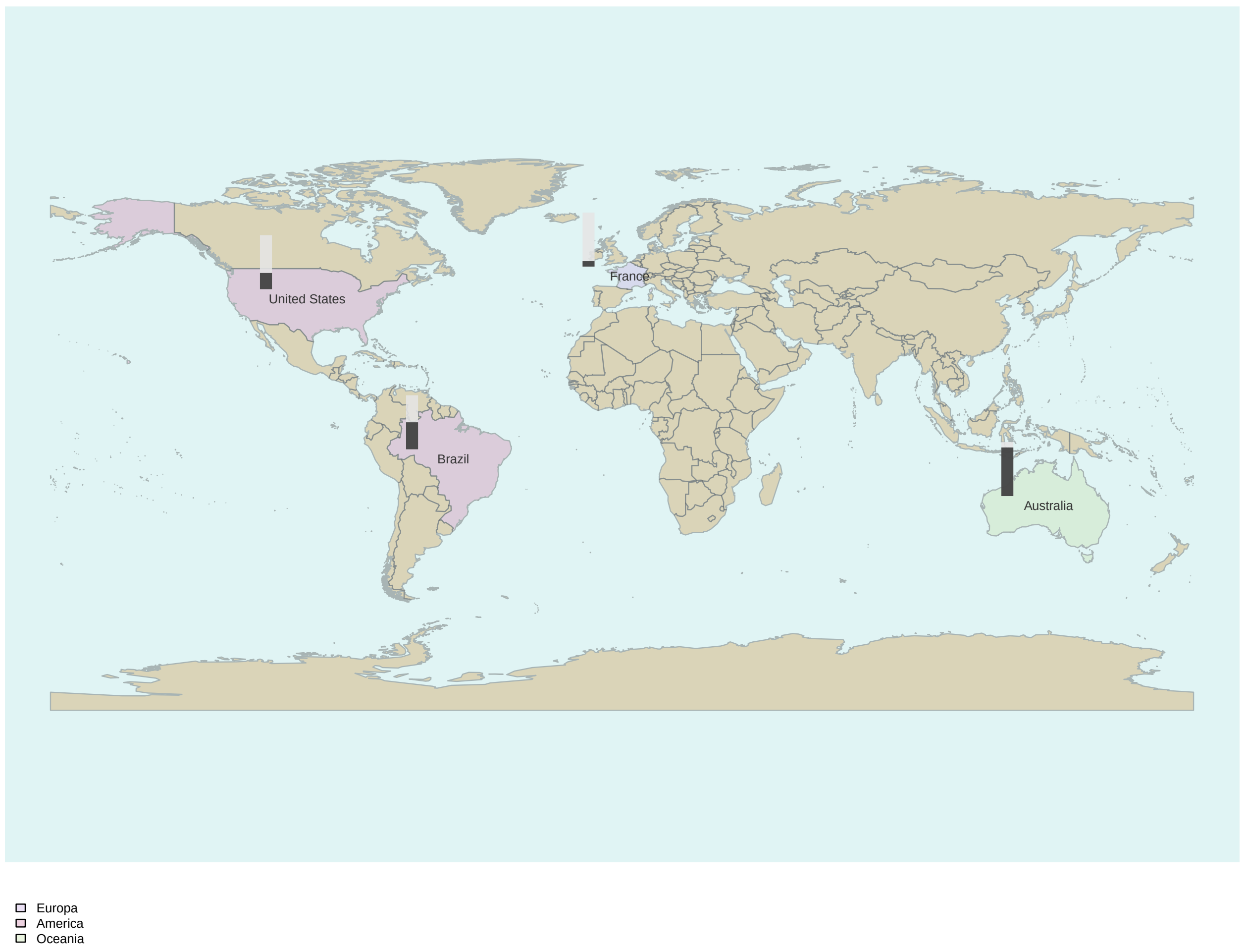
Put symbols on the map

You can add a symbol for any additional covariate:

```r
ex.bp$hemisphere <- c('North', 'North', 'South','South')
spnet.symbol.variable(ex.bp) <- 'hemisphere'
spnet.symbol.legend(ex.bp) <- c('North' = 'triangle.up', 'South' = 'triangle.down')
plot(ex.bp)
```

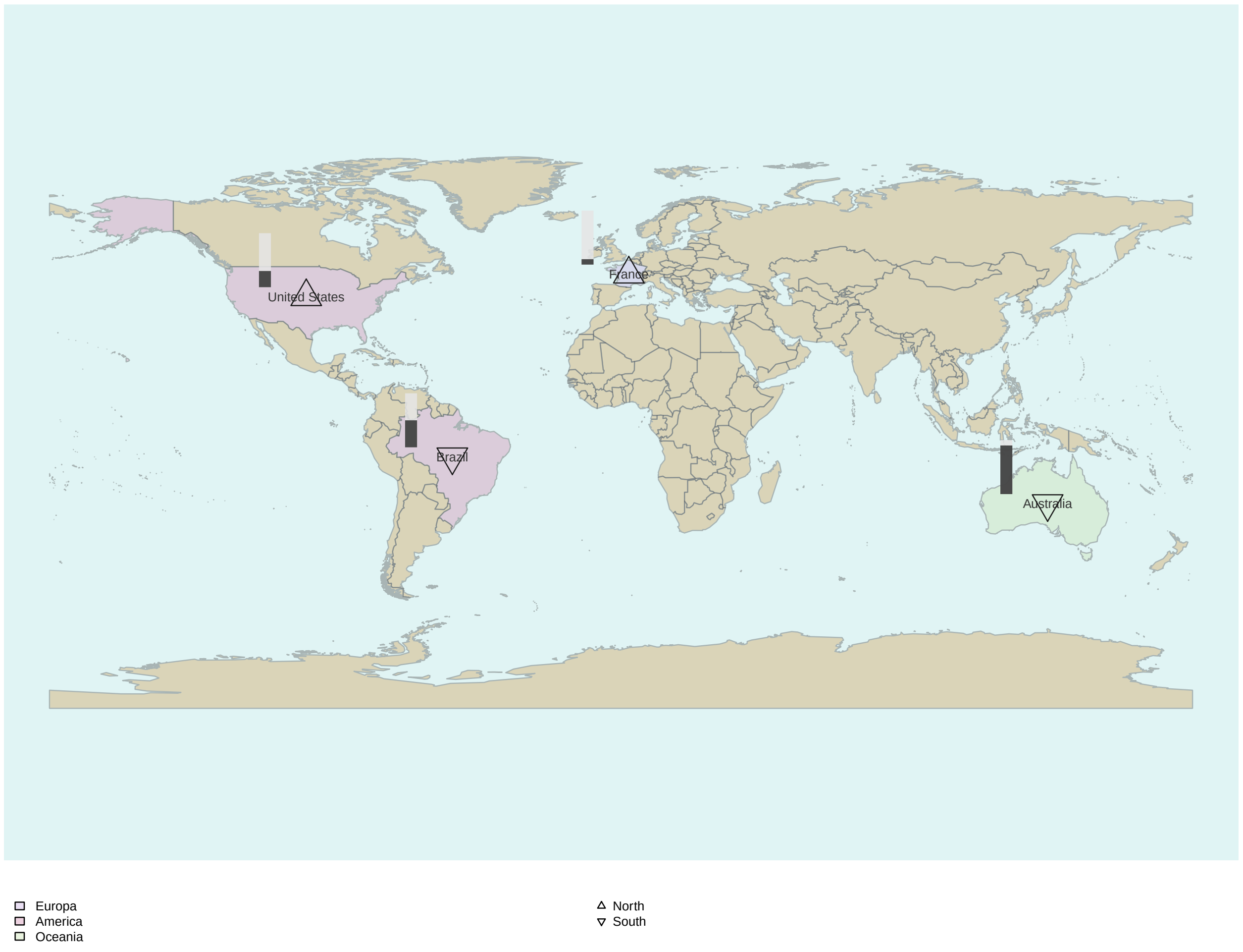
Places visited by John, Elsa, Brian and Kate

For holidays and studies



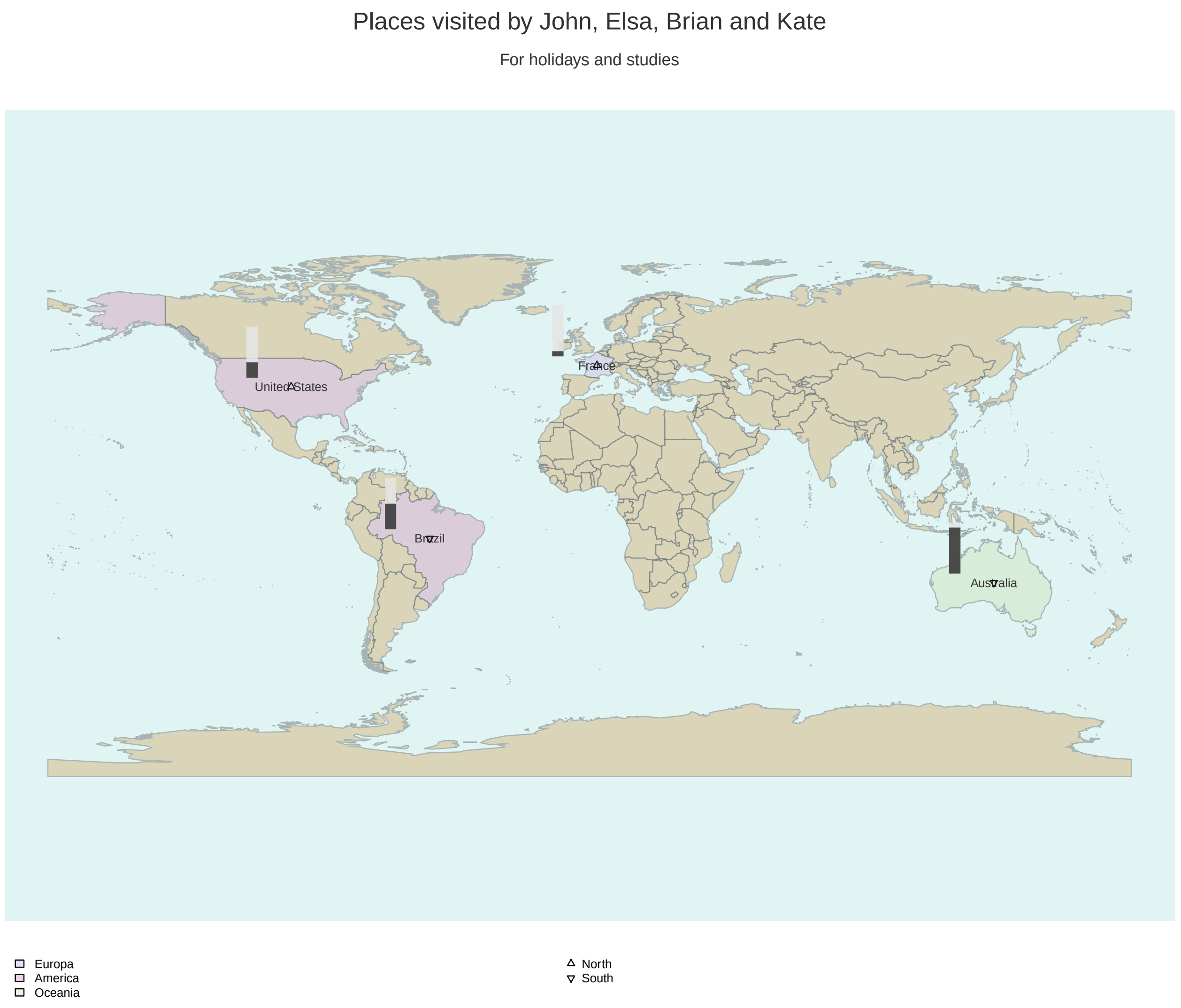
Places visited by John, Elsa, Brian and Kate

For holidays and studies



To fix size of the symbol:

```r
spnet.symbol.cex(ex.bp) <- 1
plot(ex.bp)
```



To fix position of the symbol:

```r
spnet.symbol.shift.y(ex.bp) <- 4
plot(ex.bp)
```

To fix color of the symbol:

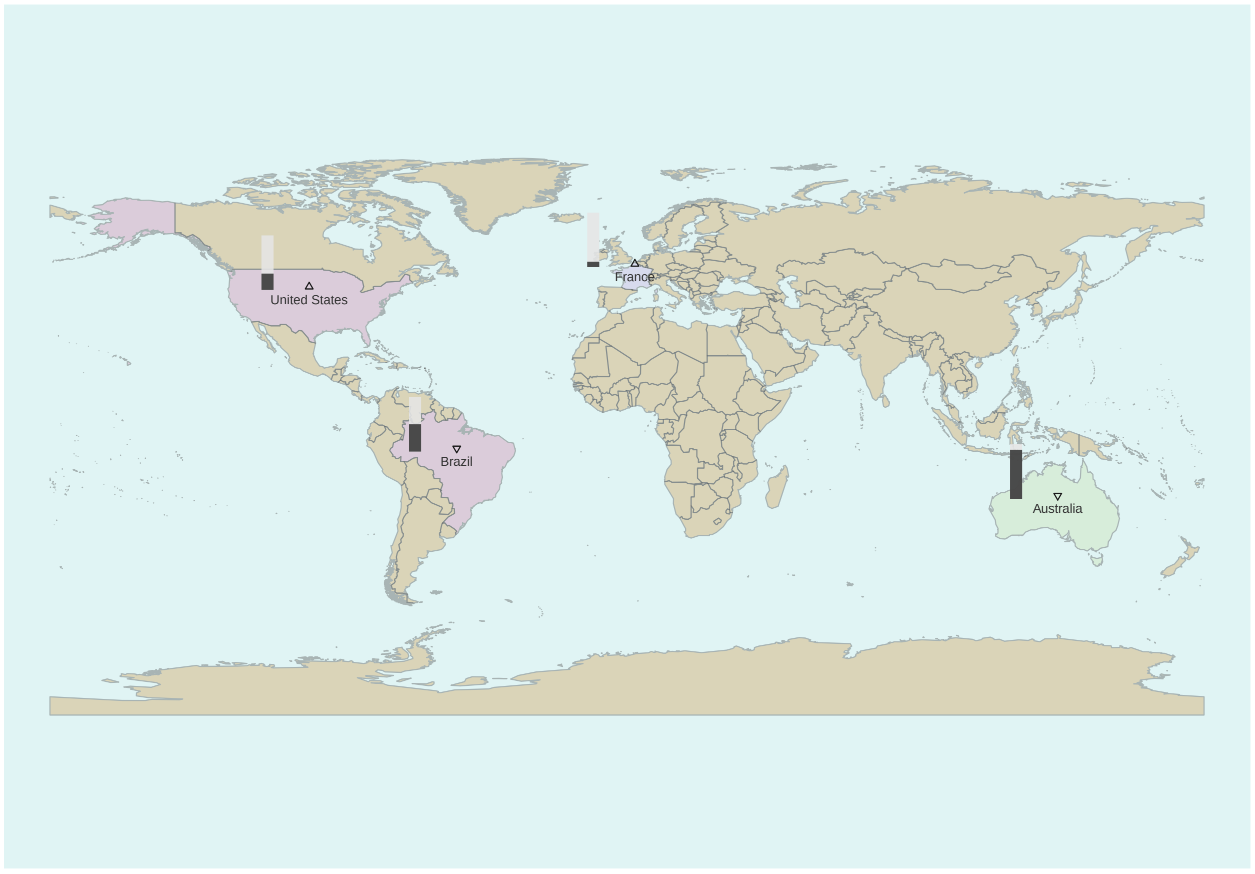
```r
spnet.symbol.color(ex.bp) <- '#A52A2A88'
plot(ex.bp)
```

Define and draft networks on the graph

To add a network on the map:

Places visited by John, Elsa, Brian and Kate

For holidays and studies

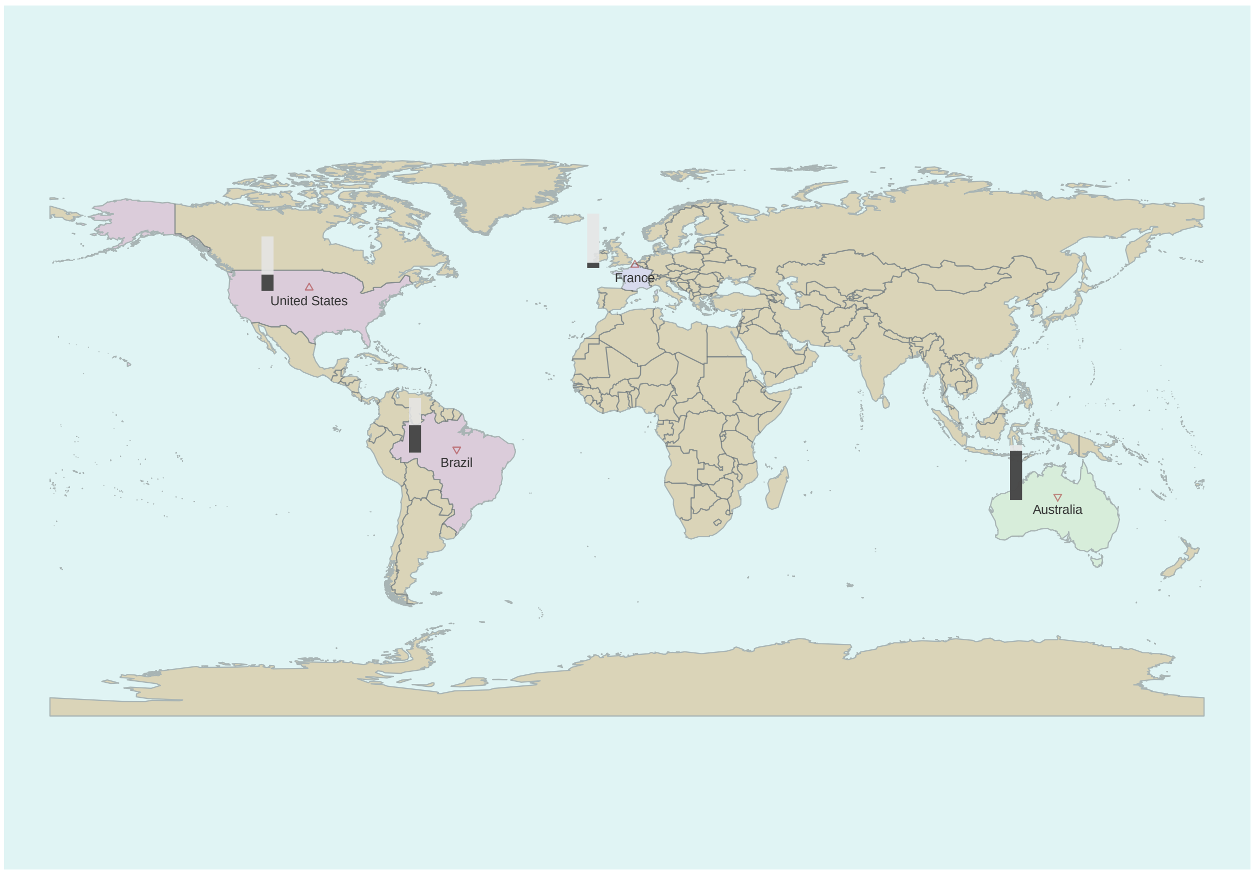


□ Europa
□ America
□ Oceania

△ North
▽ South

Places visited by John, Elsa, Brian and Kate

For holidays and studies



□ Europa
□ America
□ Oceania

△ North
▽ South

```
spnet.networks.add(ex.bp) <-'n1'
spnet.network.label(ex.bp, 'n1') <- 'Holidays'
ex.bp
```

```
## This is a valid 'SpatialNetwork' object.
##
## - Data: (first rows)
##
##            NODE POSITION continent content hemisphere
## 1        France       64    Europa     0.1      North
## 2 United States      208   America     0.3      North
## 3        Brazil       20   America     0.5      South
## 4     Australia        8   Oceania     0.9      South
##
## - Map:
##     Length: 246
##
## - Network data:
##     Number of network(s): 1
##
## - Plotting options:
##     Variable used to colorize: 'continent'
##     Variable used to draw symbols: 'hemisphere'
##     Variable used to draw barplots: 'content'
```

In the "Network data" sectionDetail of ex.bp object mentions "Number of network(s): 1".

Now, to add a second network:

```
spnet.networks.add(ex.bp) <-'n2'
spnet.network.label(ex.bp, 'n2') <- 'Studies'
```

We define a matrix manually; for this example we will use:

```
matrix_int <- matrix(
  rep(0, length(node)^2),
  nrow = length(node),
  dimnames = list(node, node)
)
```

And now we attach it to our `spnet` object:

```
spnet.network.data(ex.bp, 'n1') <- matrix_int
spnet.network.data(ex.bp, 'n2') <- matrix_int
```

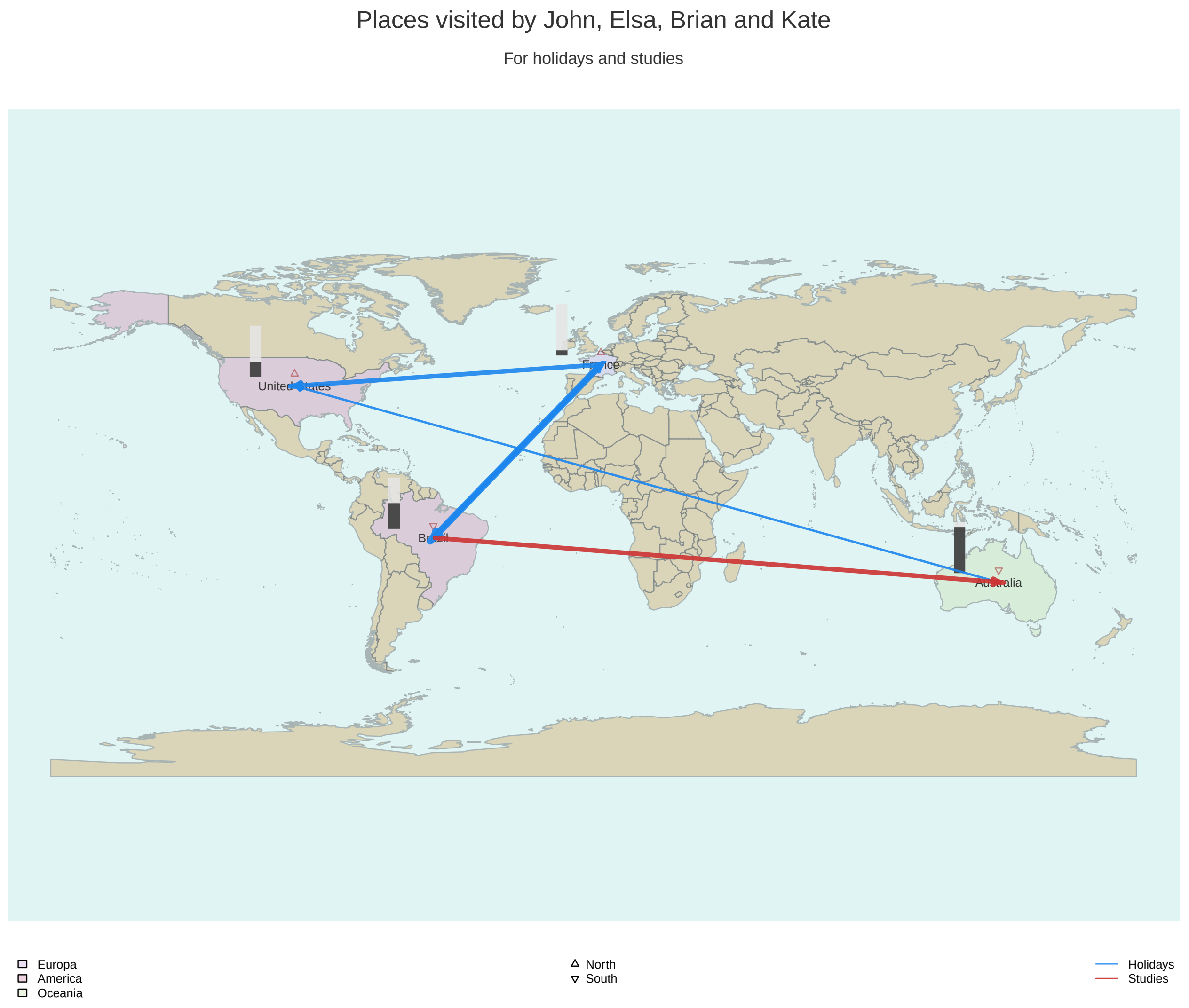
A matrix can also be imported from a data file.

We fill our network manually directly into the `spnet` object:

```
#Network 1
spnet.network.data(ex.bp, 'n1')['France', 'United States'] <- 2
spnet.network.data(ex.bp, 'n1')['Australia', 'United States'] <- 1
```

```r
spnet.network.data(ex.bp, 'n1')['France', 'Brazil'] <- 3
spnet.network.data(ex.bp, 'n1')['Brazil', 'France'] <- 2

#Network 2
spnet.network.data(ex.bp, 'n2')['Brazil', 'Australia'] <- 2
plot(ex.bp)
```



There are ‘by default’ colors, but we can change them by using:

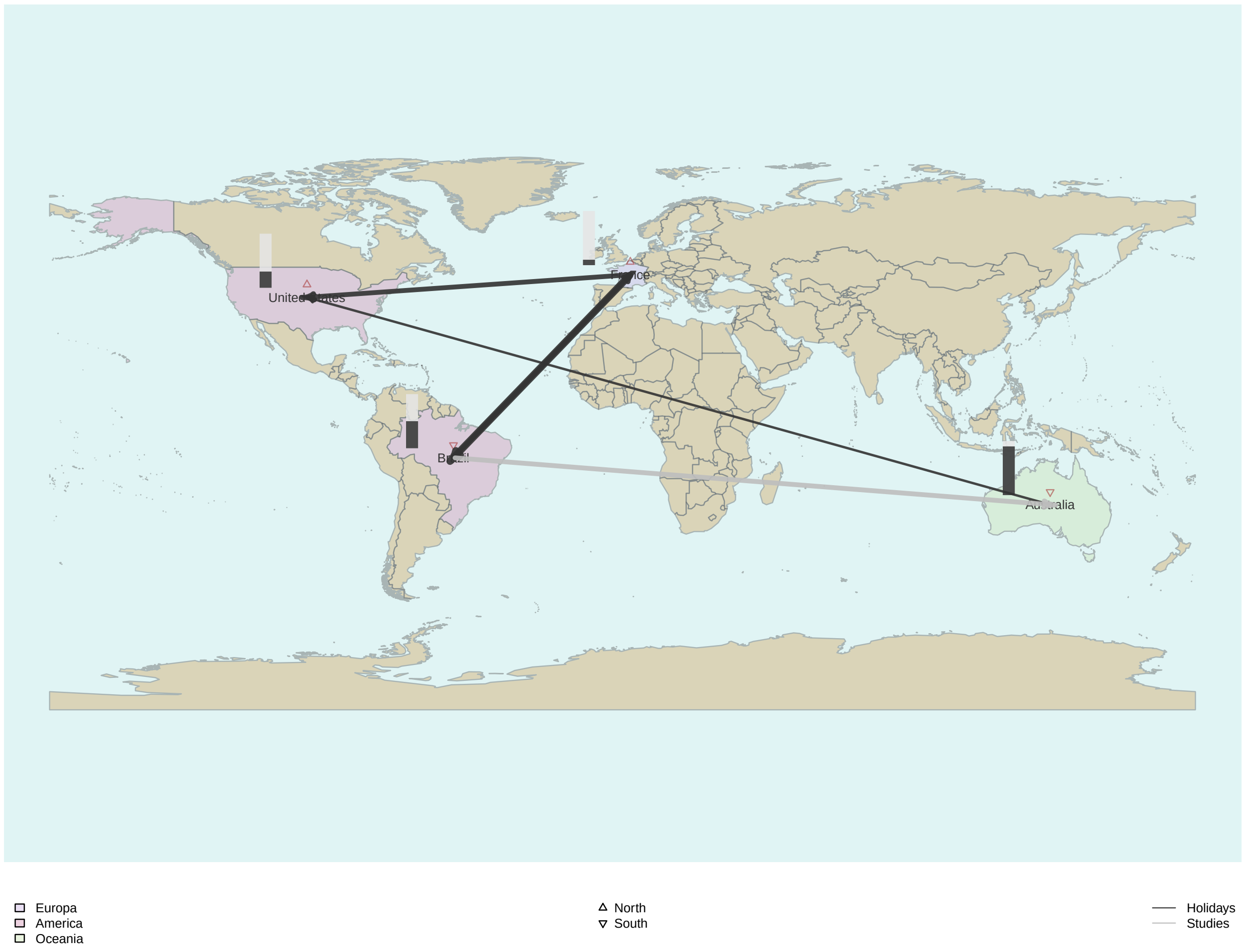
```r
spnet.network.arrow.color(ex.bp, 'n1') <- '#33333366'
spnet.network.arrow.color(ex.bp, 'n2') <- 'grey'
plot(ex.bp)
```

We can reduce as well the thinkness and length of arrows:

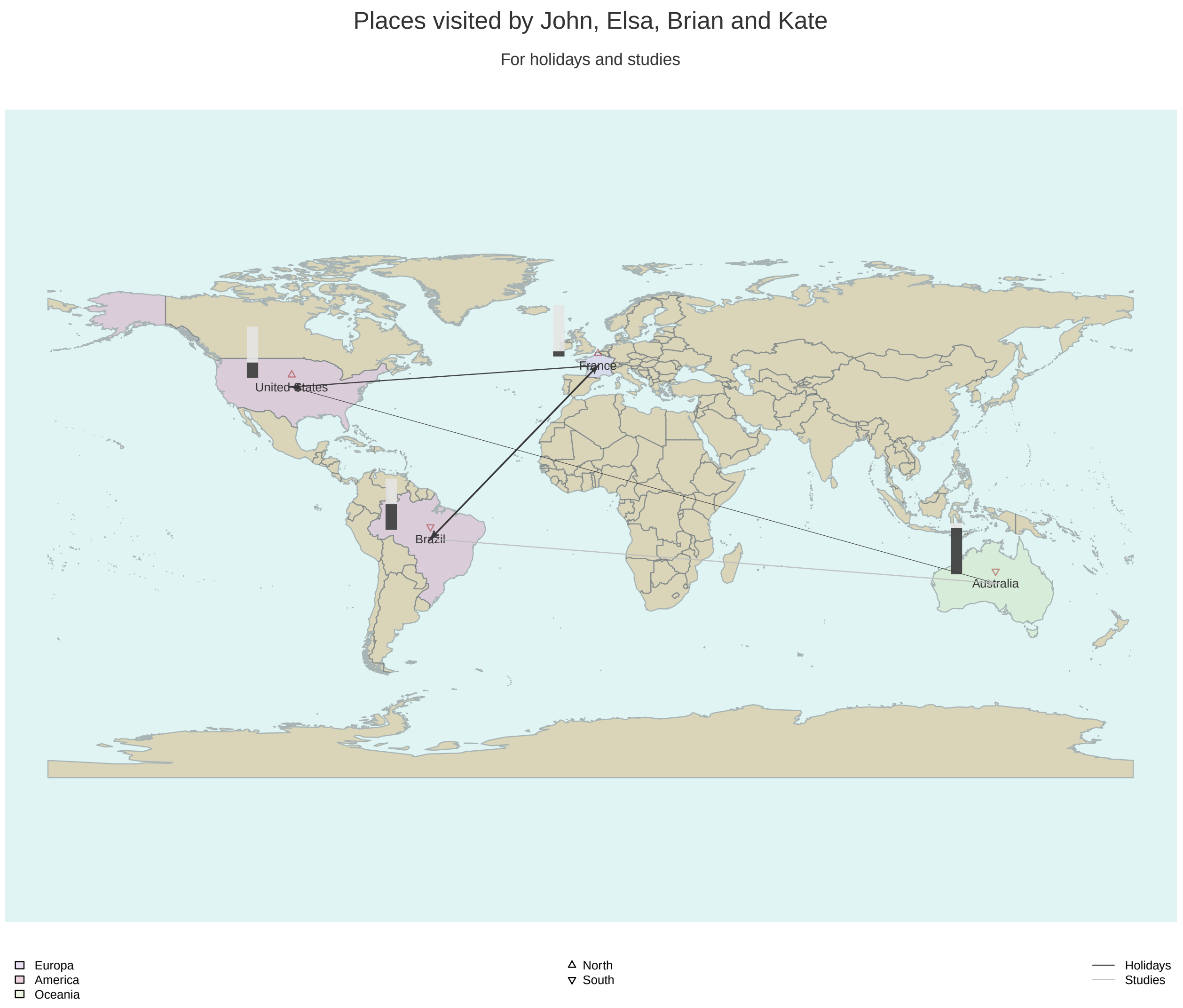
```r
spnet.network.arrow.thickness(ex.bp, 'n1') <- 0.5
spnet.network.arrow.thickness(ex.bp, 'n2') <- 0.5
```

Places visited by John, Elsa, Brian and Kate

For holidays and studies



```
spnet.network.arrow.shorten(ex.bp, 'n1') <- 0.9
spnet.network.arrow.shorten(ex.bp, 'n2') <- 0.9
plot(ex.bp)
```



Some examples to grasp how to edit the graph:

We shift 'Holidays' a little bit upper:

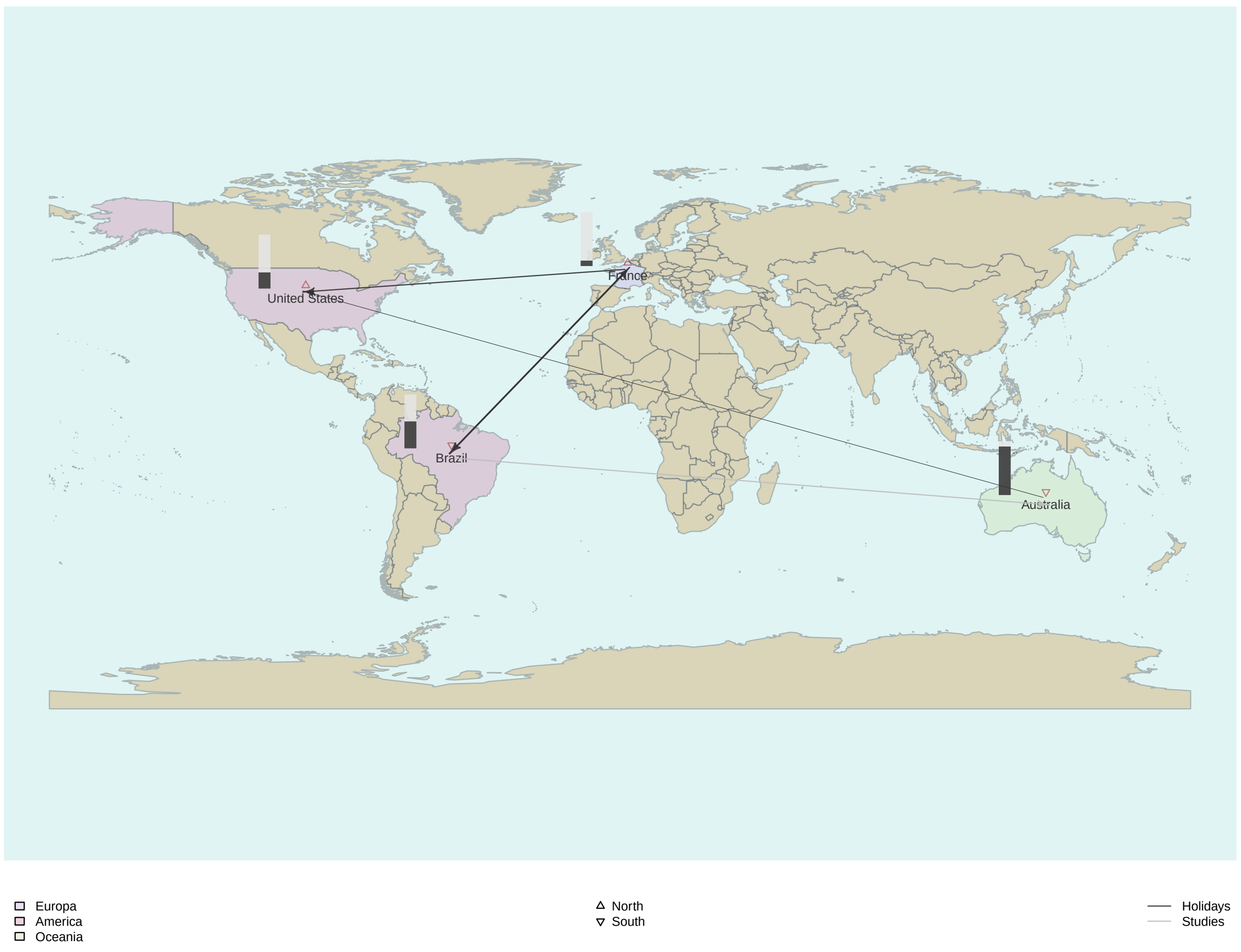
```
spnet.network.arrow.shift.y(ex.bp, 'n1') <- 2
plot(ex.bp)
```

And we shift 'Studies' a little bit lower and give it some transparency:

```
spnet.network.arrow.shift.y(ex.bp, 'n2') <- -2
spnet.network.arrow.opacity(ex.bp, 'n2') <- 0.9
plot(ex.bp)
```

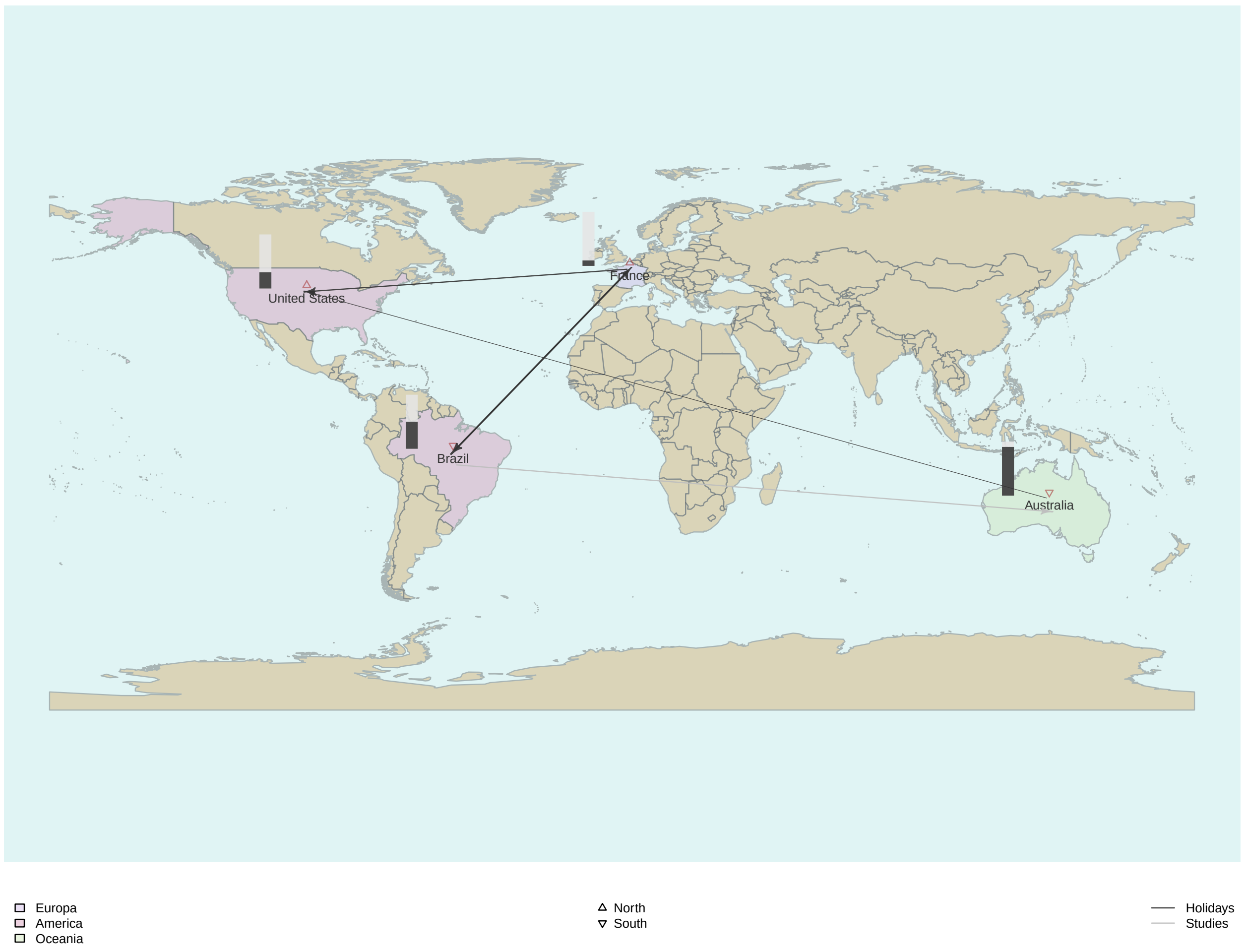
Places visited by John, Elsa, Brian and Kate

For holidays and studies



Places visited by John, Elsa, Brian and Kate

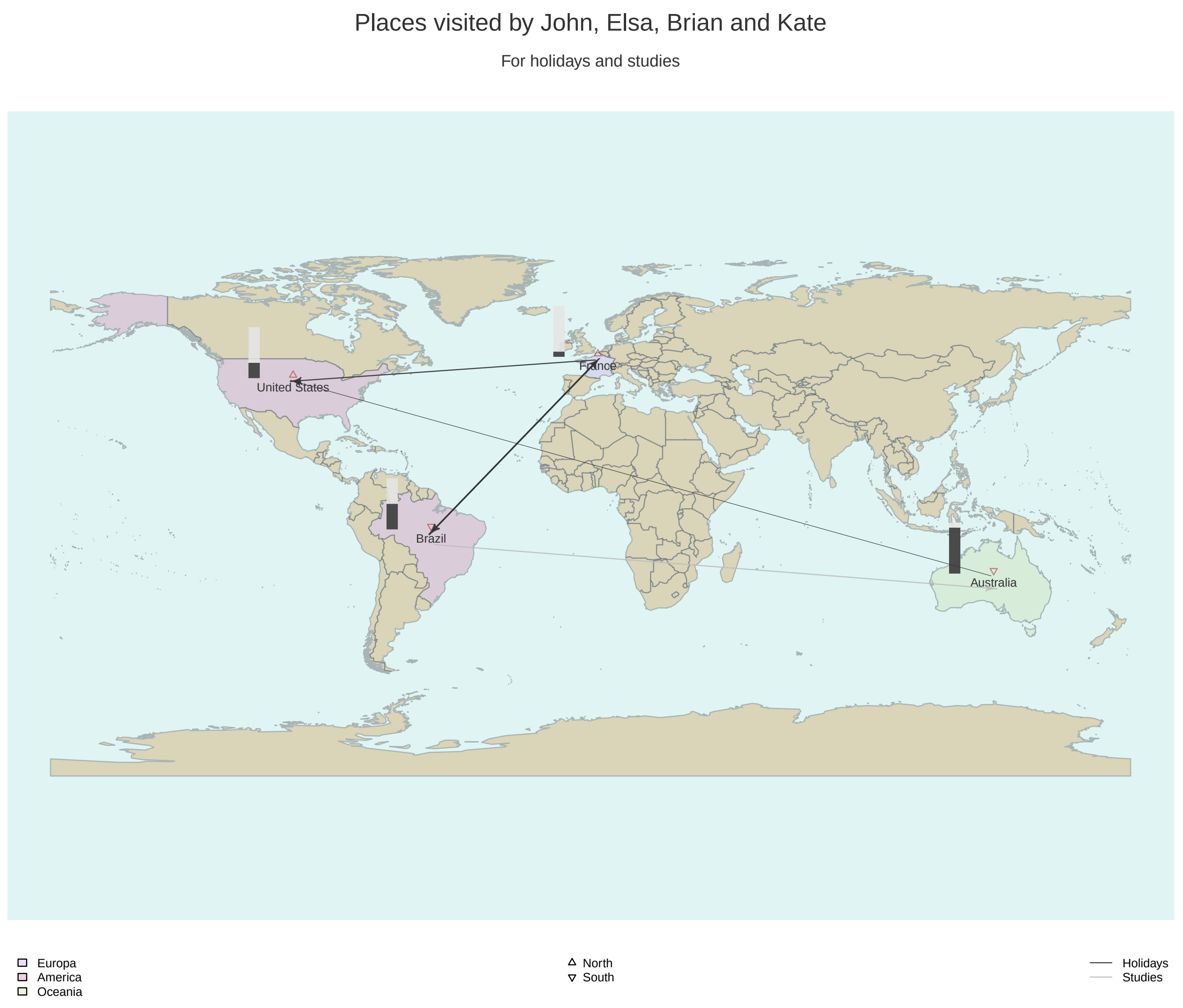
For holidays and studies



Title and legend

To give a title to the graph we use:

```r
spnet.title.main(ex.bp) <- "Places visited by John, Elsa, Brian and Kate"
spnet.title.sub(ex.bp) <- "For holidays and studies"
plot(ex.bp)
```



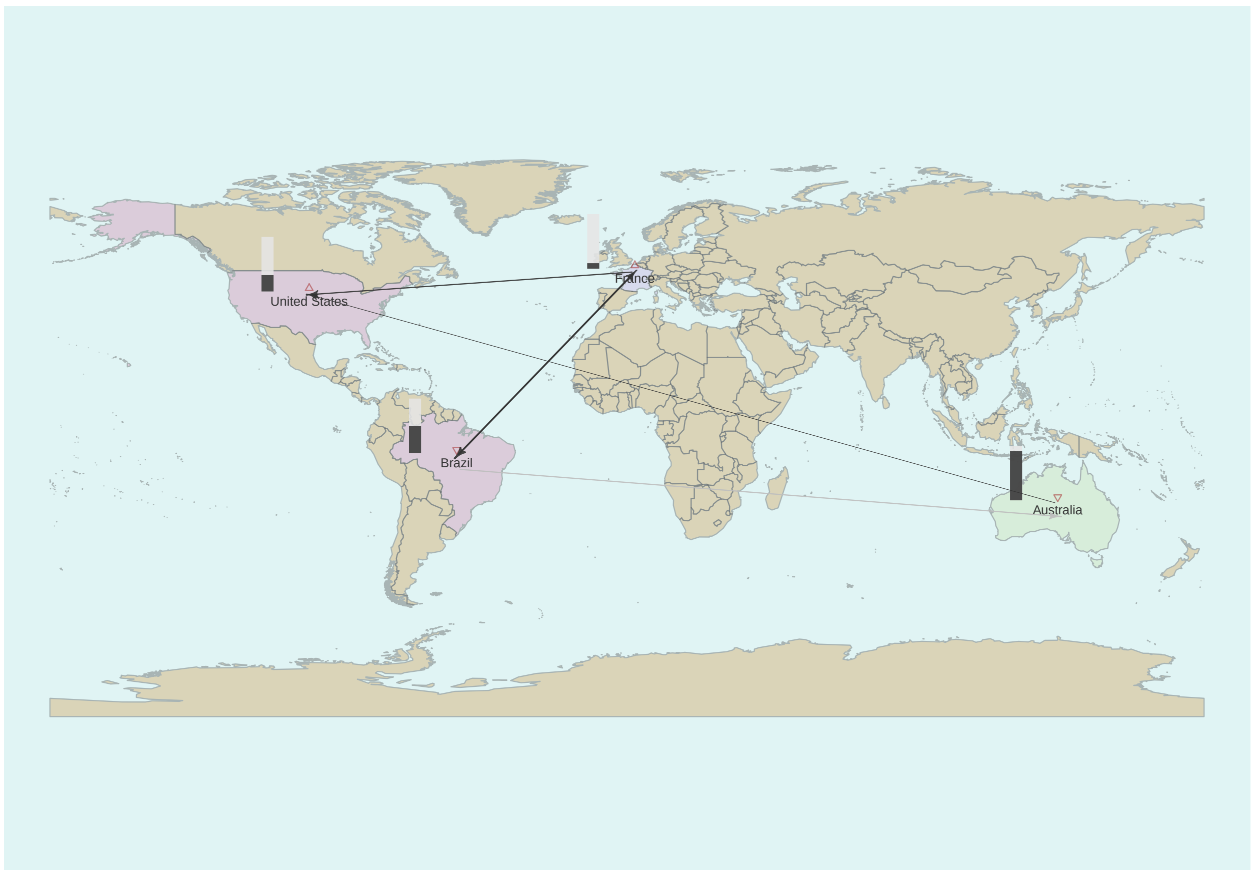
In order to make the legend a little bit bigger, network lines more visible and colored variable legend on 3 columns, for instance, we use:

```r
spnet.legend.cex(ex.bp) <- 1.2
spnet.legend.line.width(ex.bp) <- 2
spnet.legend.ncol(ex.bp) <- 3
plot(ex.bp)
```

To hide the legend, we should run:

Places visited by John, Elsa, Brian and Kate

For holidays and studies

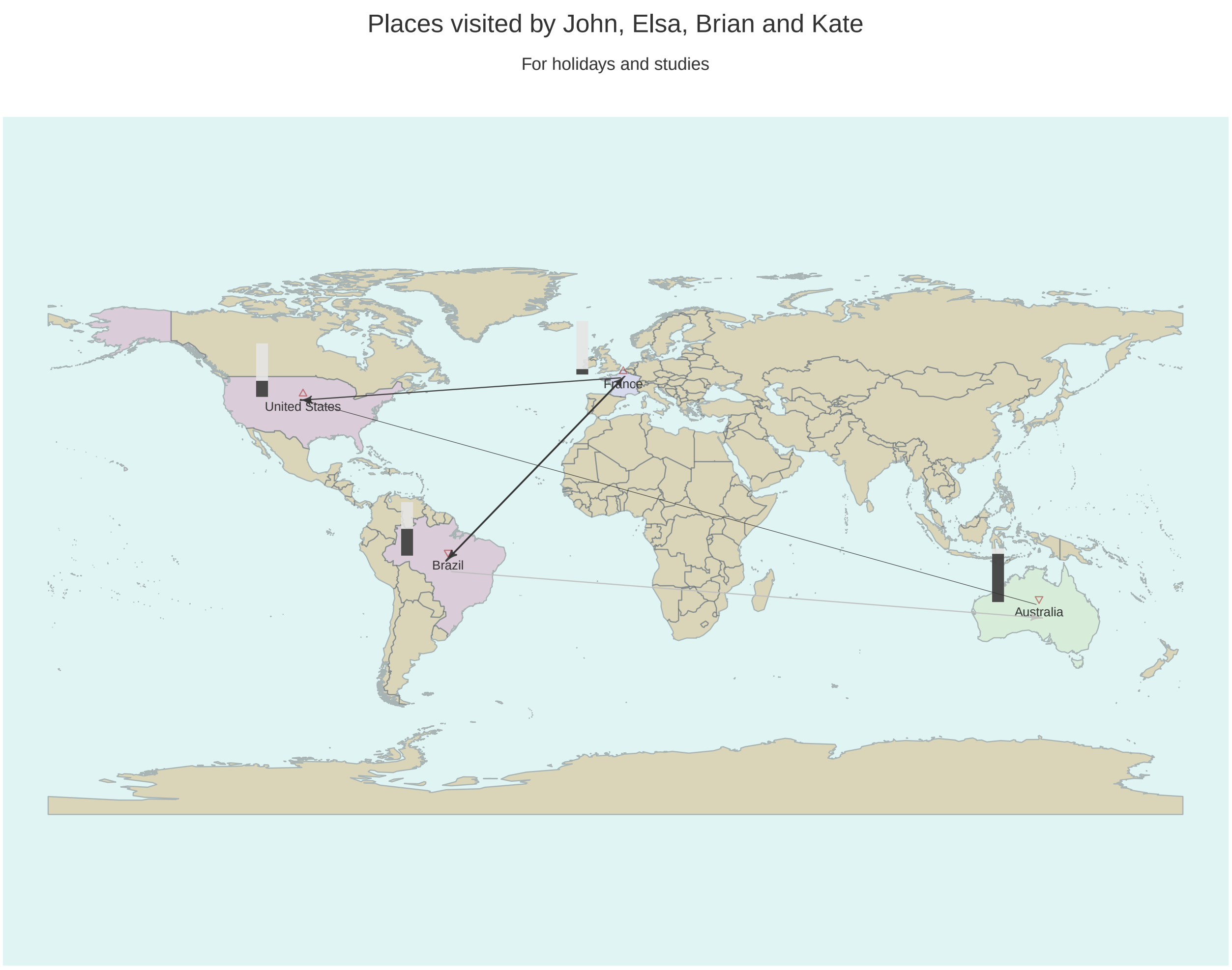


□ Europa □ America □ Oceania

△ North
▽ South

— Holidays
— Studies

```r
spnet.legend.print(ex.bp) <- FALSE
plot(ex.bp)
```


Places visited by John, Elsa, Brian and Kate
For holidays and studies
United States
France
Brazil
Australia