Overview of the spnet package

Emmanuel Rousseaux, Marion Deville and Gilbert Ritschard

2015.01.13

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.

```r
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.

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

Usage

Creating a `spnet` object

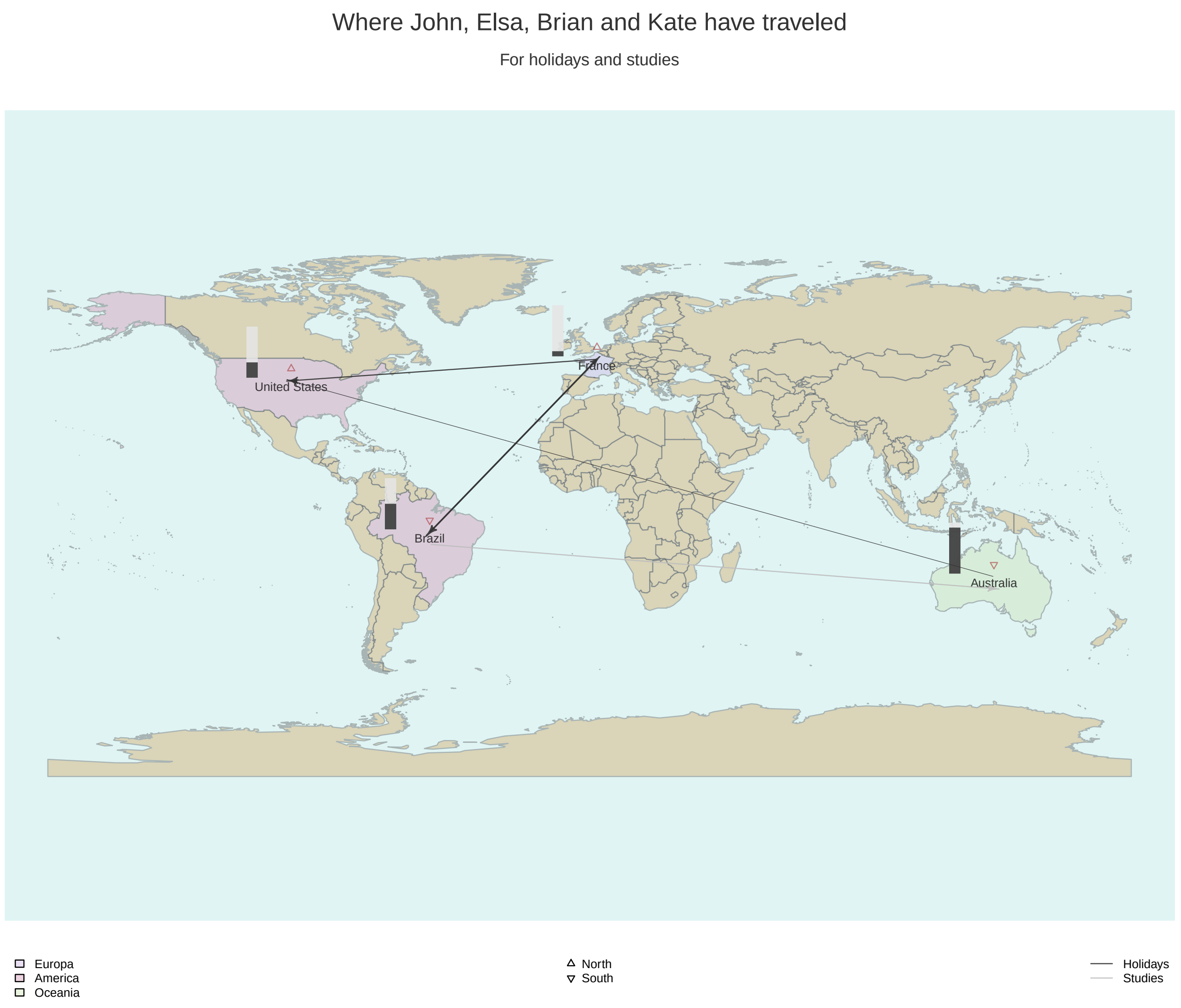
Creating a new `spnet` object (formal class `SpatialNetwork`) is done by the `spnet.create` function. You will have to supply two required parameters: the node IDs, and the corresponding position IDs on the map (which will be set in a second step). Note that in a `spnet` object data are stored in a `data.frame`. In this `data.frame` the two required parameters will respectively supplied as variables `'NODE'` and `'POSITION'`. Here is an example:

```r
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
  )
)
```


Where John, Elsa, Brian and Kate have traveled
For holidays and studies
United States
France
Brazil
Australia
Europa
America
Oceania
North
South
Holidays
Studies

```r
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:

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

This example contains the following data:

```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'
```

You can change the size and the color of labels:

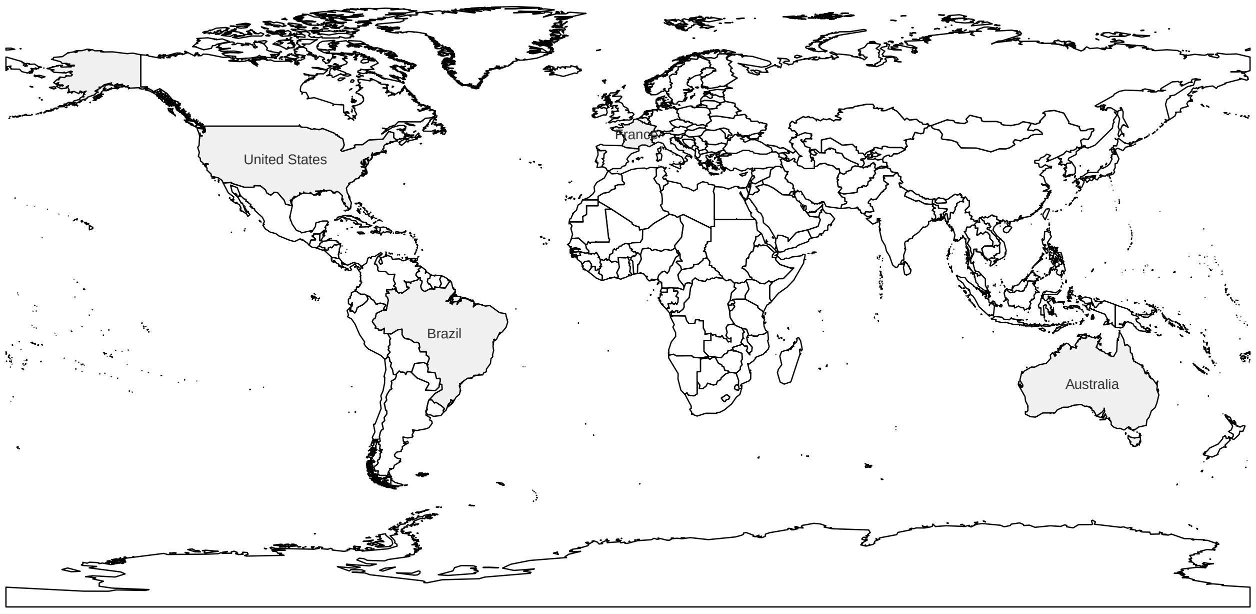
```r
spnet.label.cex(net1) <- 0.7
spnet.label.color(net1) <- '#FF0000'

plot(net1)
```

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

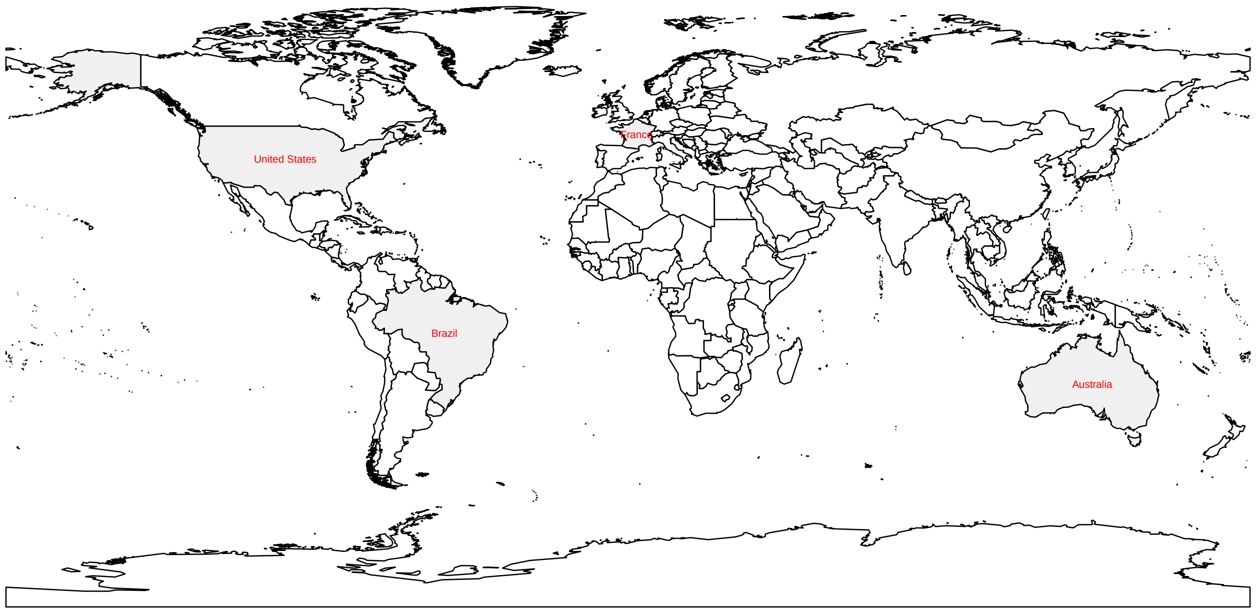
Where John, Elsa, Brian and Kate have traveled

For holidays and studies



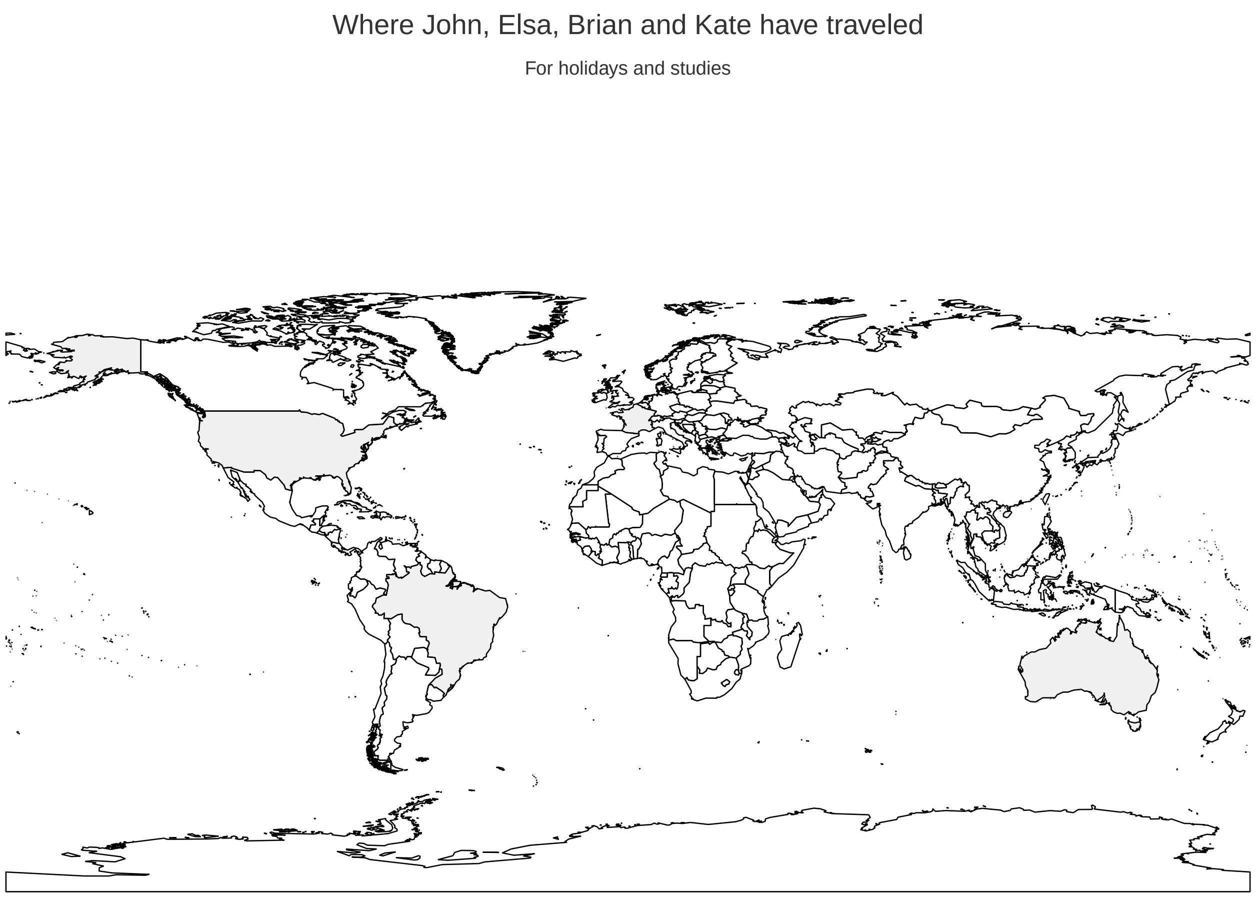
Where John, Elsa, Brian and Kate have traveled

For holidays and studies



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

plot(net1)
```



Setting colors

To set colors you basically need:

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

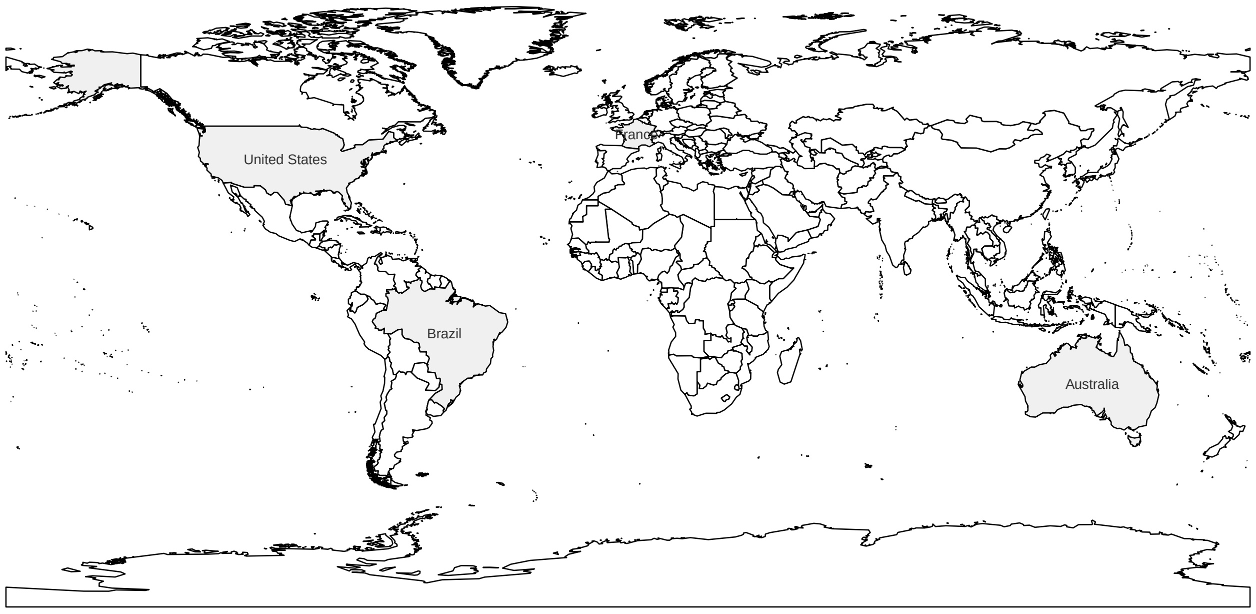
Here is a practical example. First, we start with our basic `spnet` object containing a map.

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

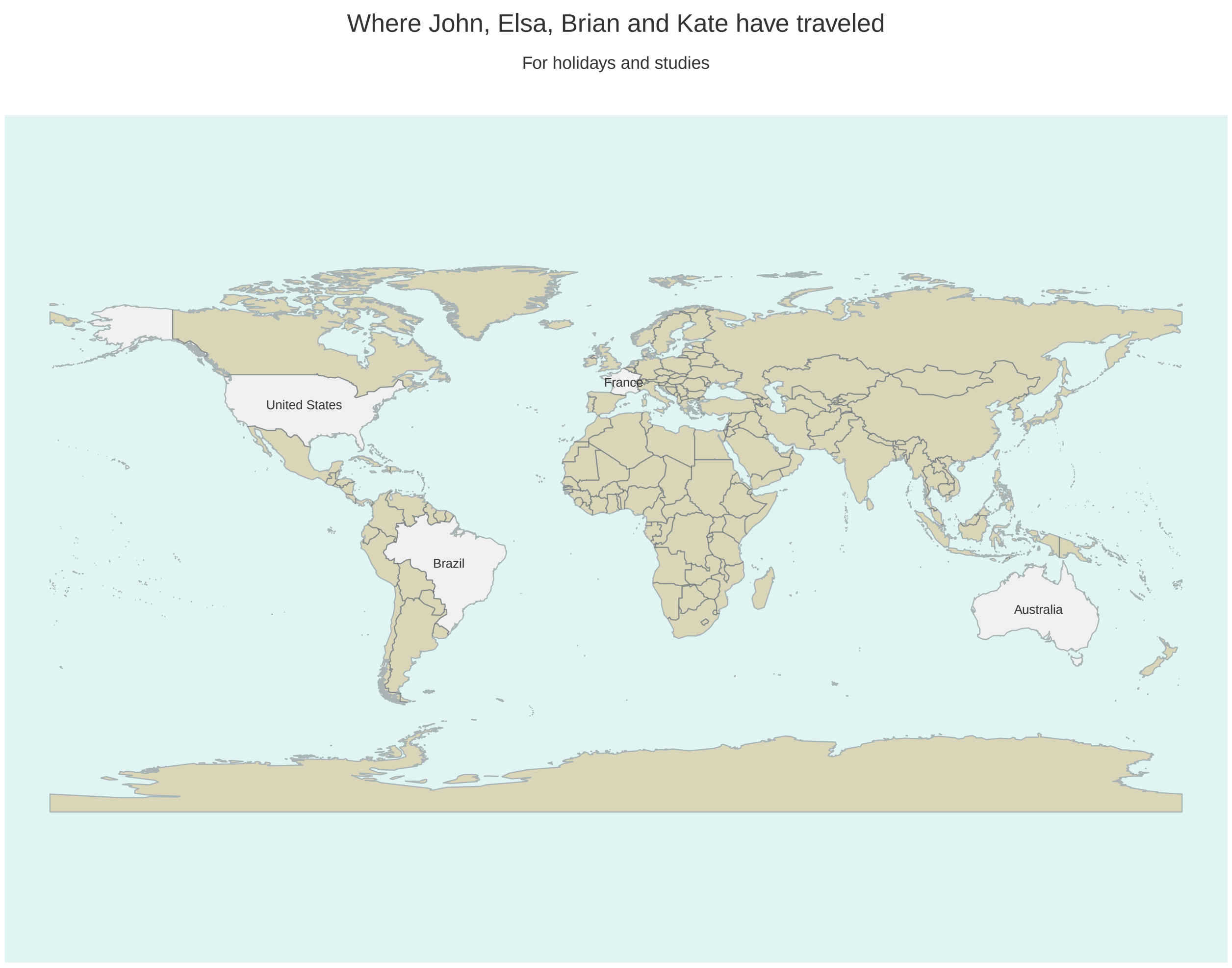
To make the graphic nicer, we can set set the backround color, border color, and a default color for colorizing the regions:

Where John, Elsa, Brian and Kate have traveled

For holidays and studies



```
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:

```
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:

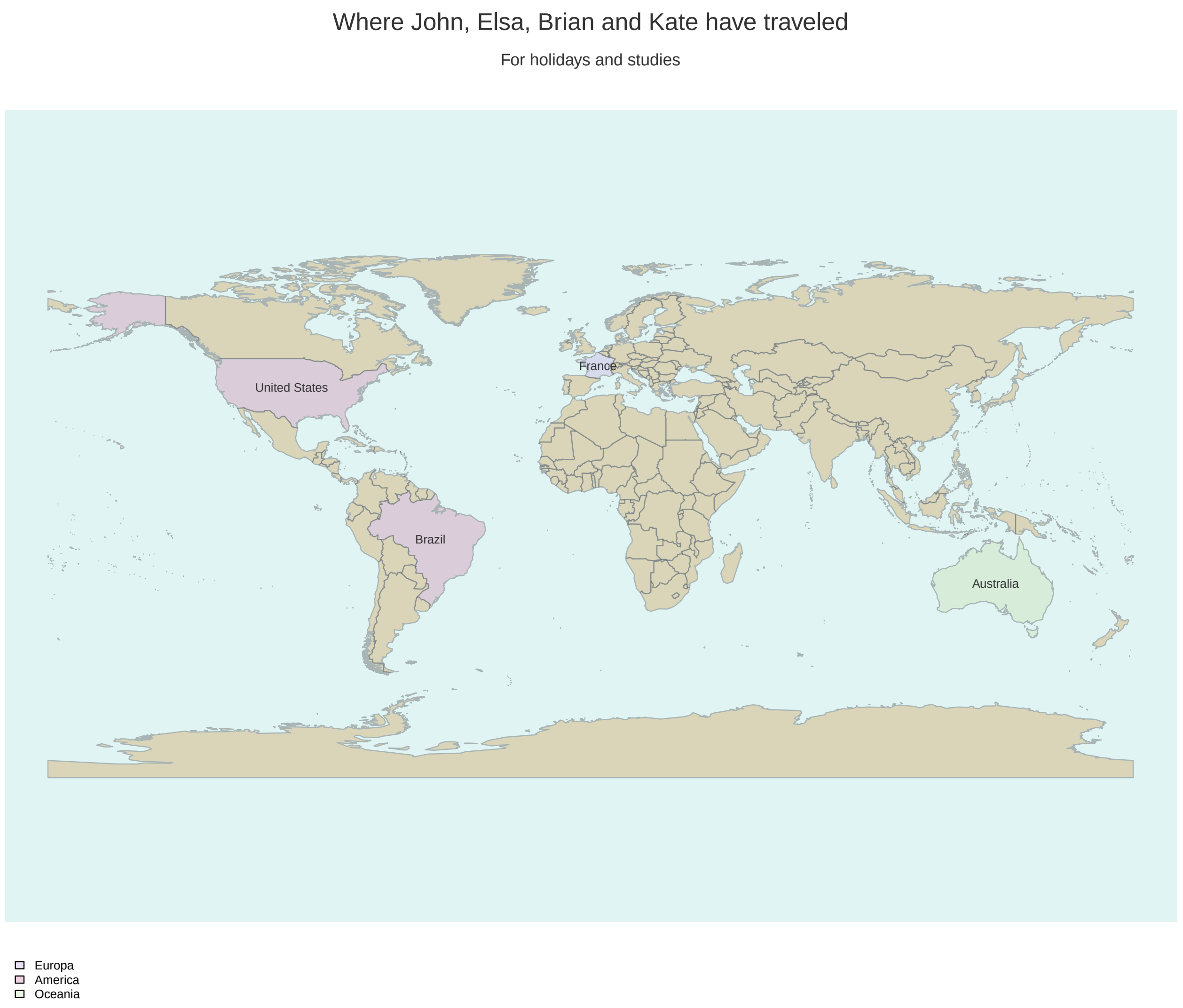
```r
spnet.color.variable(net1) <- "continent"
```

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 graphic:

```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 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'
```

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().

```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)
```

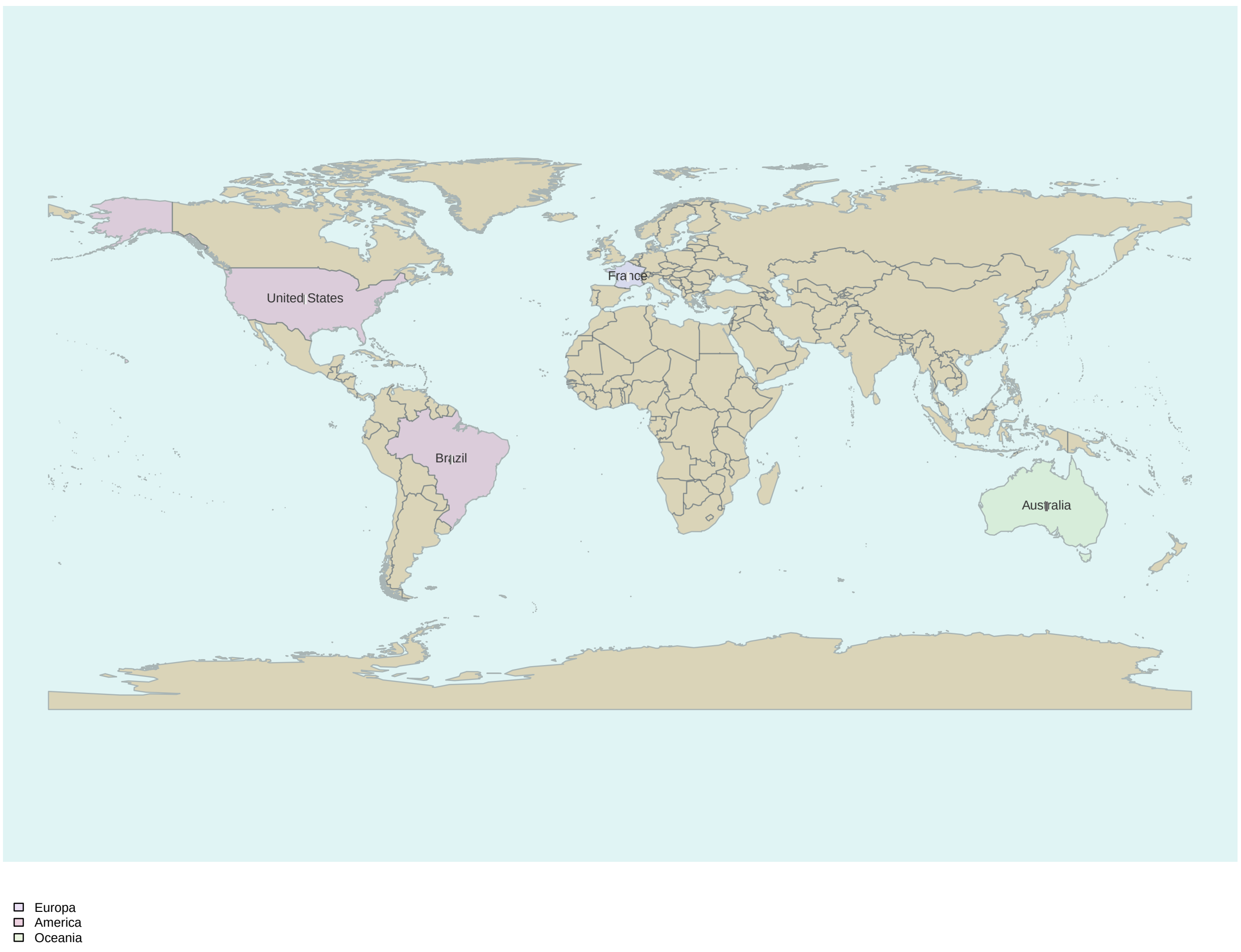
The plot is by default in black and white. On the graphs, individual barplots appear at the bottom of the boxes. To pull them up a little, use the functions `spnet.barplot.bound.lower` and `spnet.barplot.bound.uper` to adjust. The coordinate system is computed from the center of the grid (as for the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

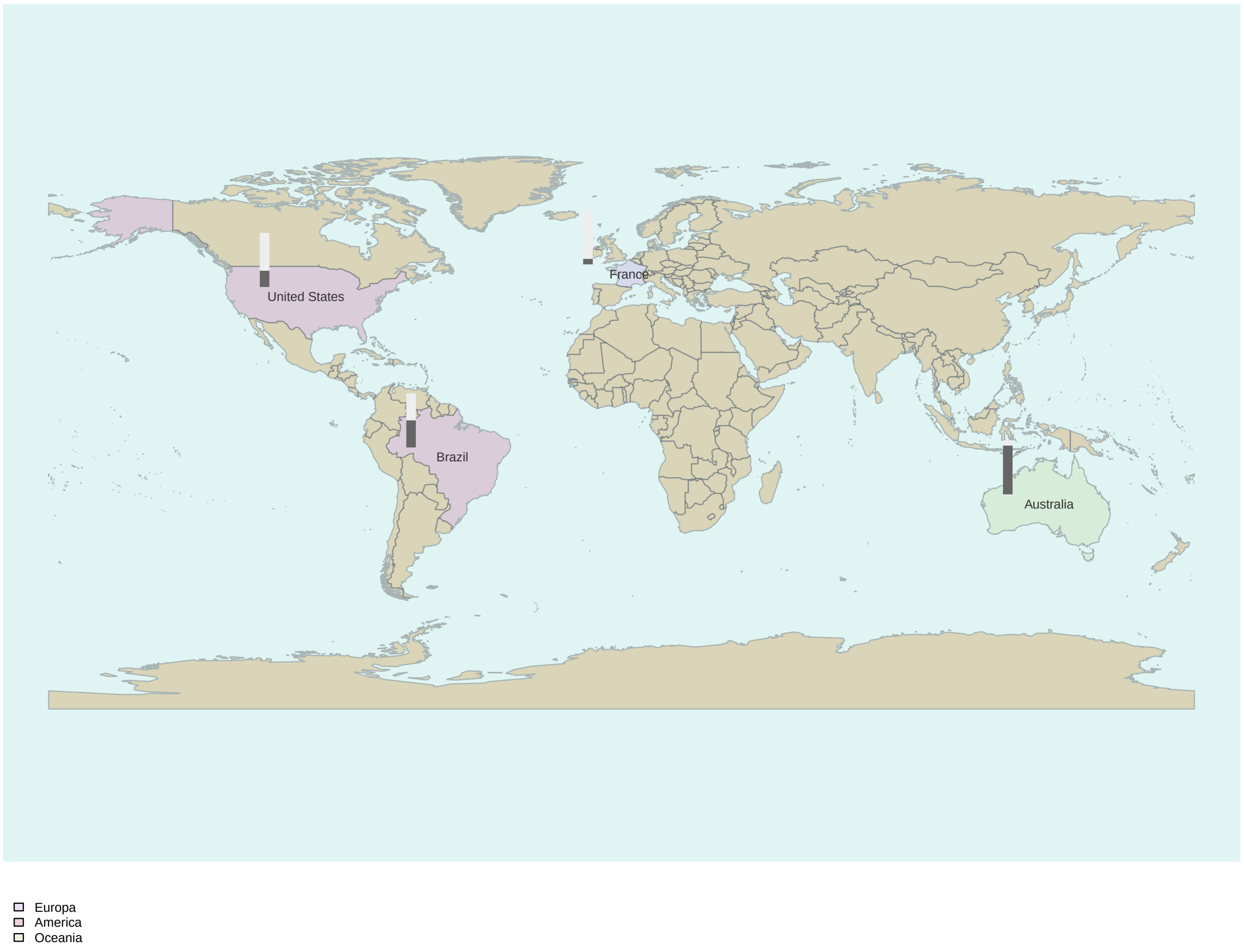
Where John, Elsa, Brian and Kate have traveled

For holidays and studies

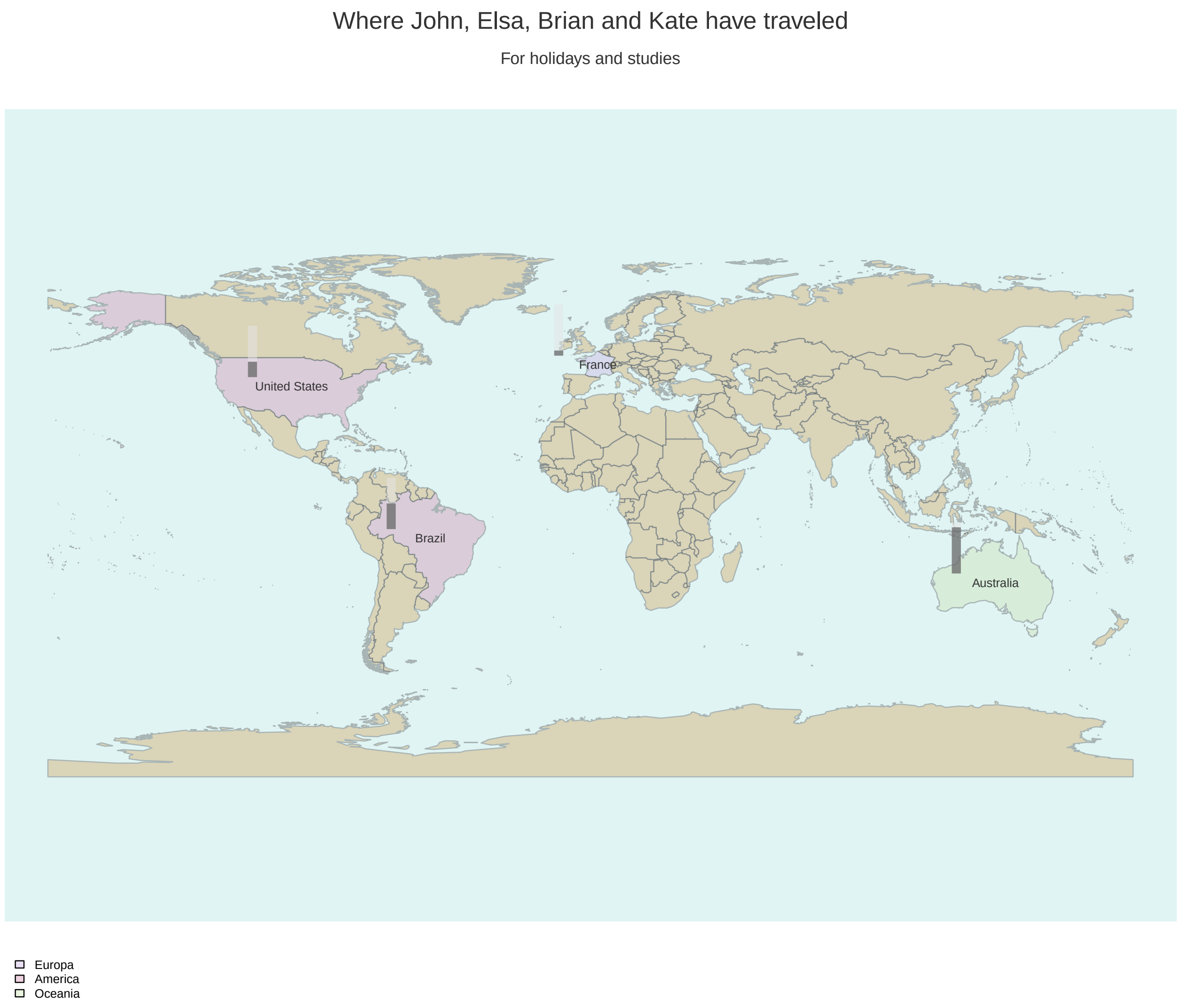


Where John, Elsa, Brian and Kate have traveled

For holidays and studies



```r
spnet.barplot.fgcolor(ex.bp) <- "#33333388"
spnet.barplot.bgcolor(ex.bp) <- "#E6E6E688"
plot(ex.bp)
```



To find colors, you can go to a website like ColorPicker.com. To give a transparency to the color, put "66" at the end.

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

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

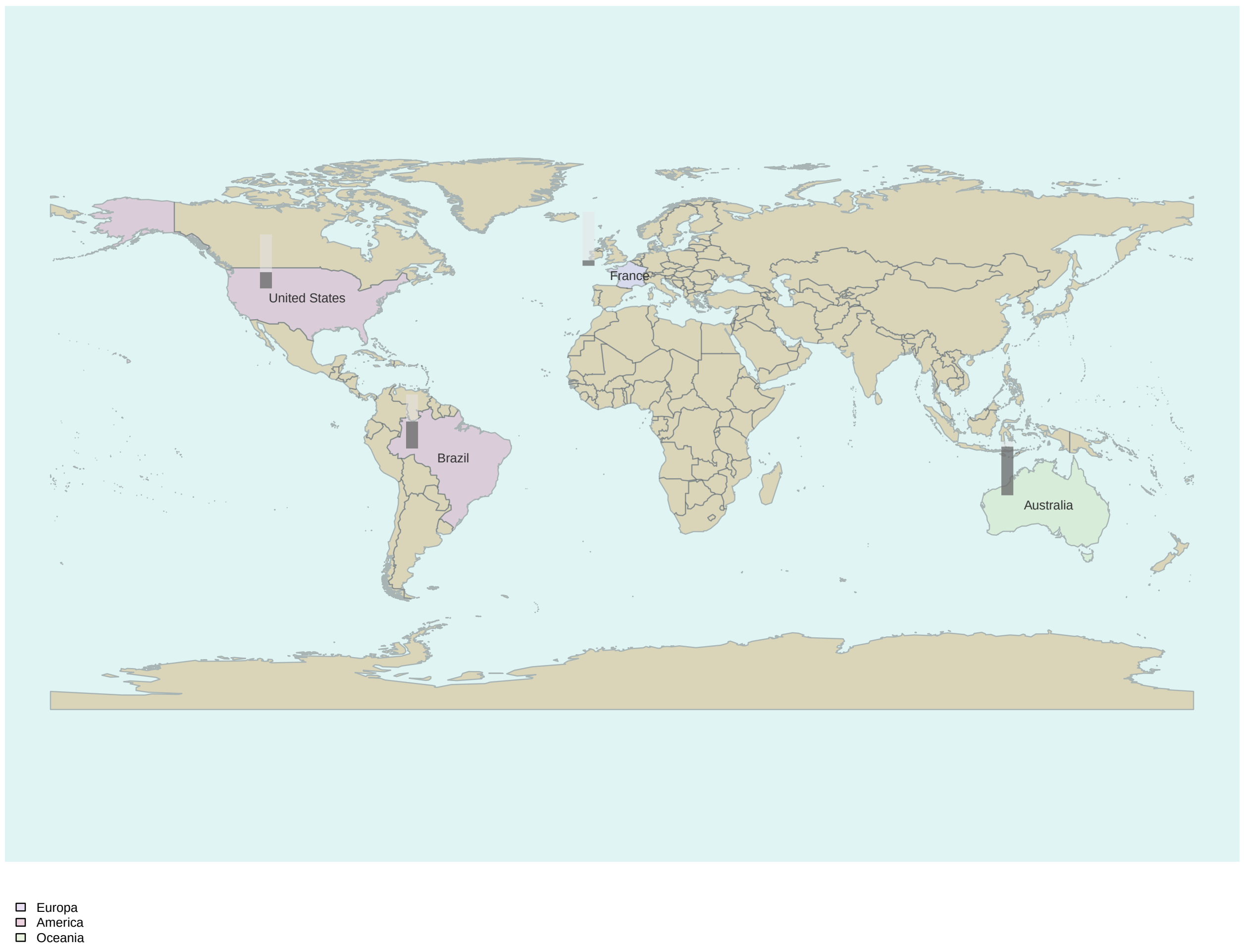
Put symbols on the map

You can add a symbol for 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)
```

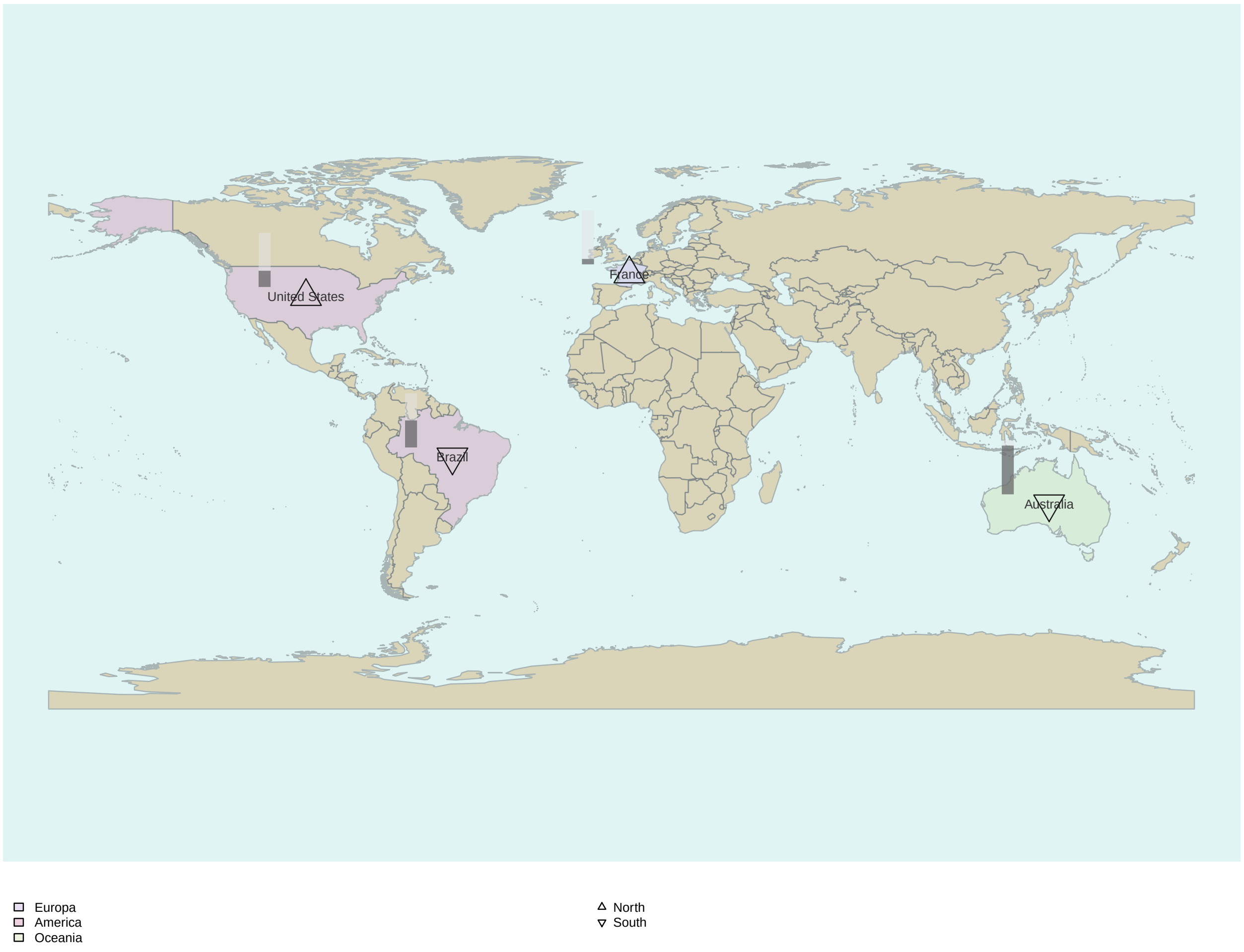
Where John, Elsa, Brian and Kate have traveled

For holidays and studies



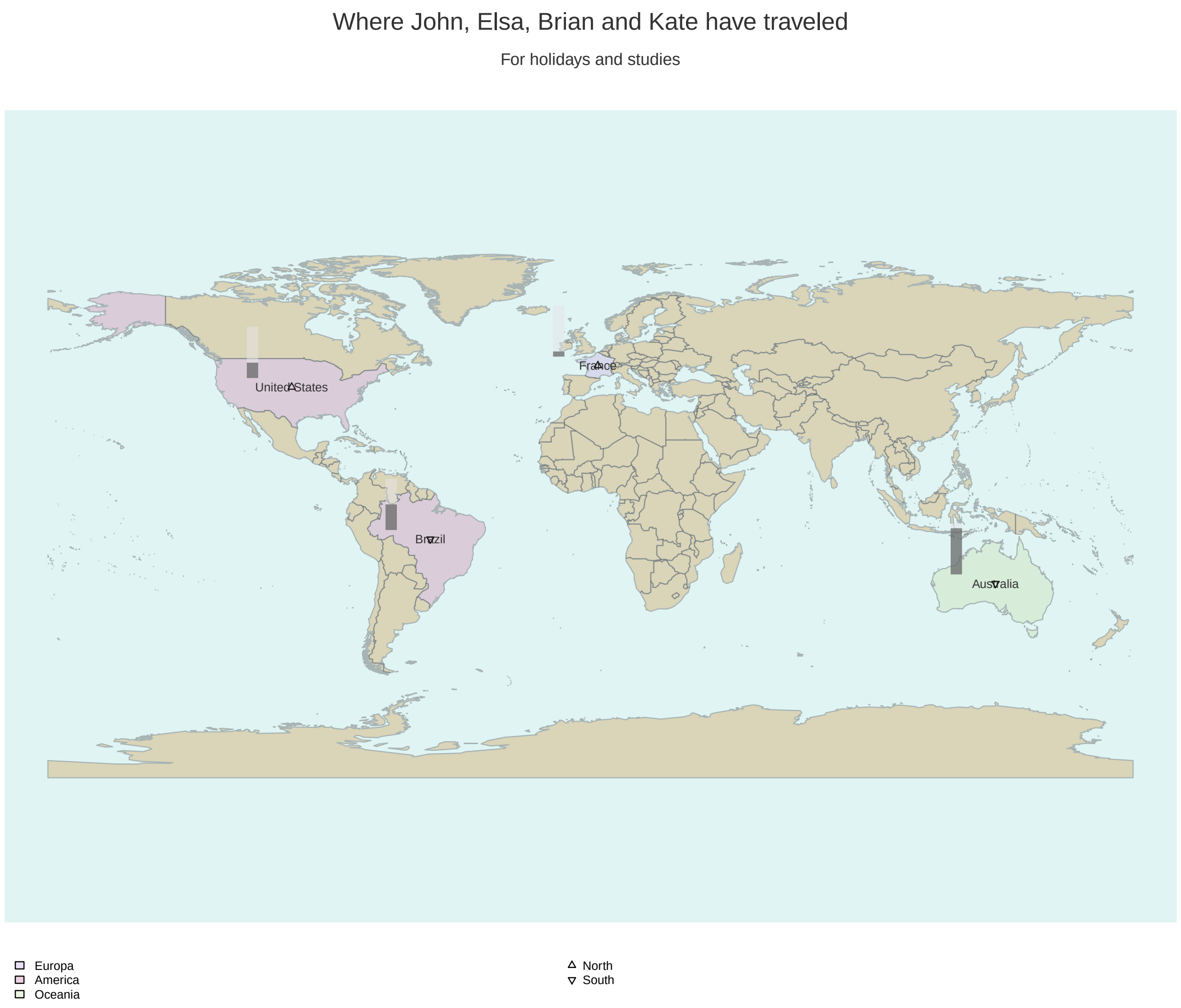
Where John, Elsa, Brian and Kate have traveled

For holidays and studies



To fix size, position and color of the symbol:

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



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

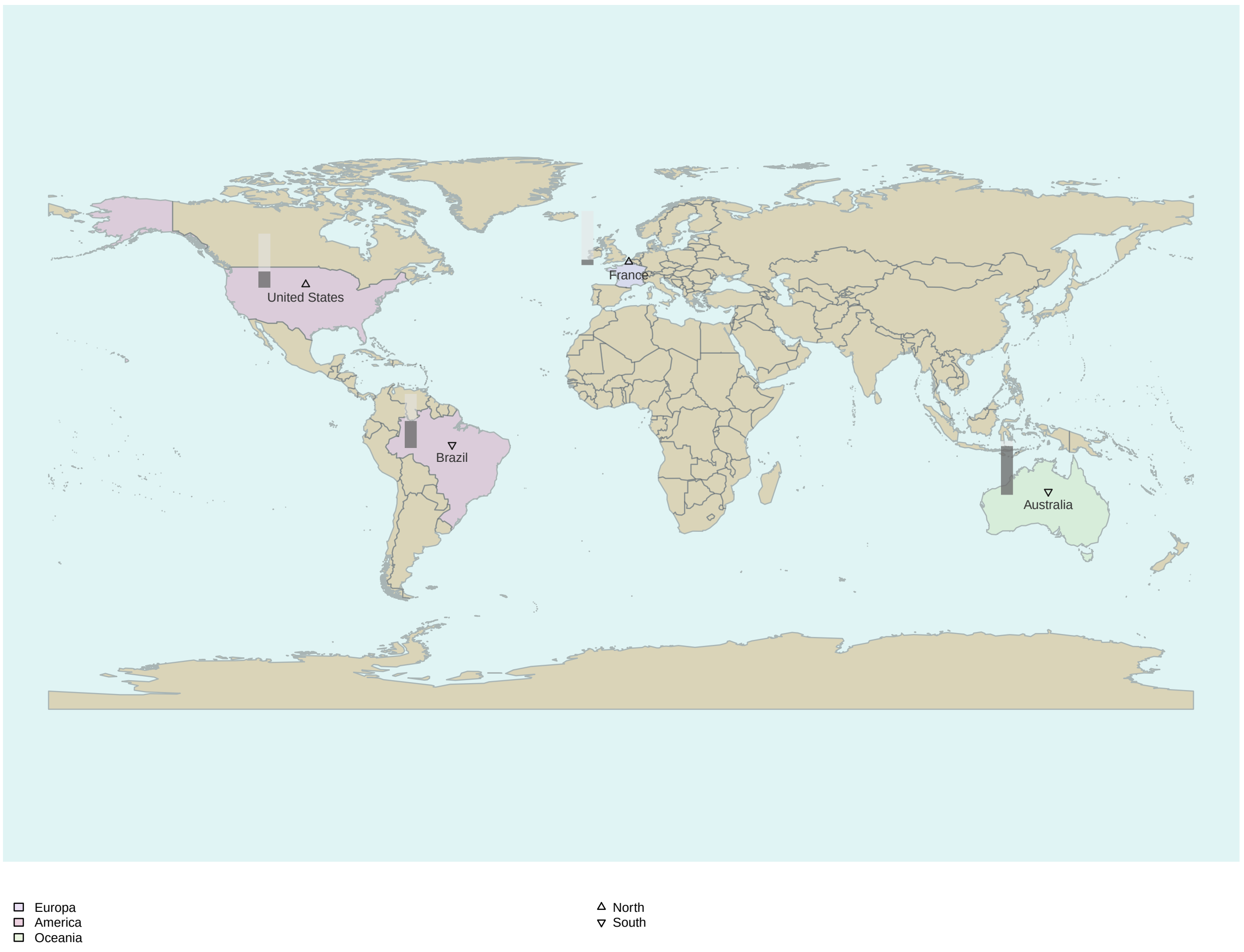
```r
spnet.symbol.color(ex.bp) <- '#A52A2A88'
plot(ex.bp)
```

Define and draft networks on the graph

To add a network:

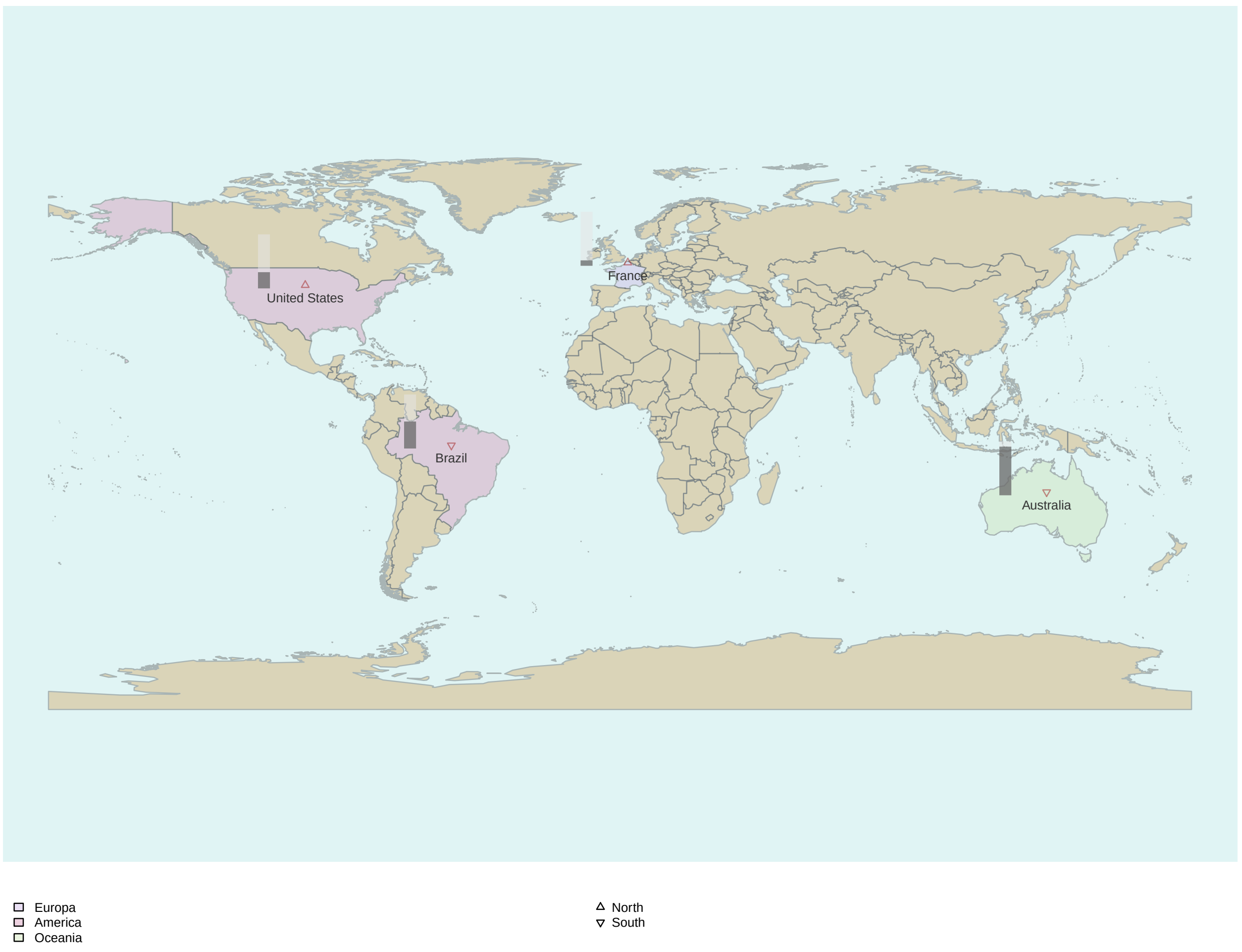
Where John, Elsa, Brian and Kate have traveled

For holidays and studies



Where John, Elsa, Brian and Kate have traveled

For holidays and studies



```
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'
```

Detail of ex.bp object mention “Number of network(s): 1” in “Network data” section.

We add a second network:

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

We define a matrix manually for the example:

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

And attache it to our `spnet` object:

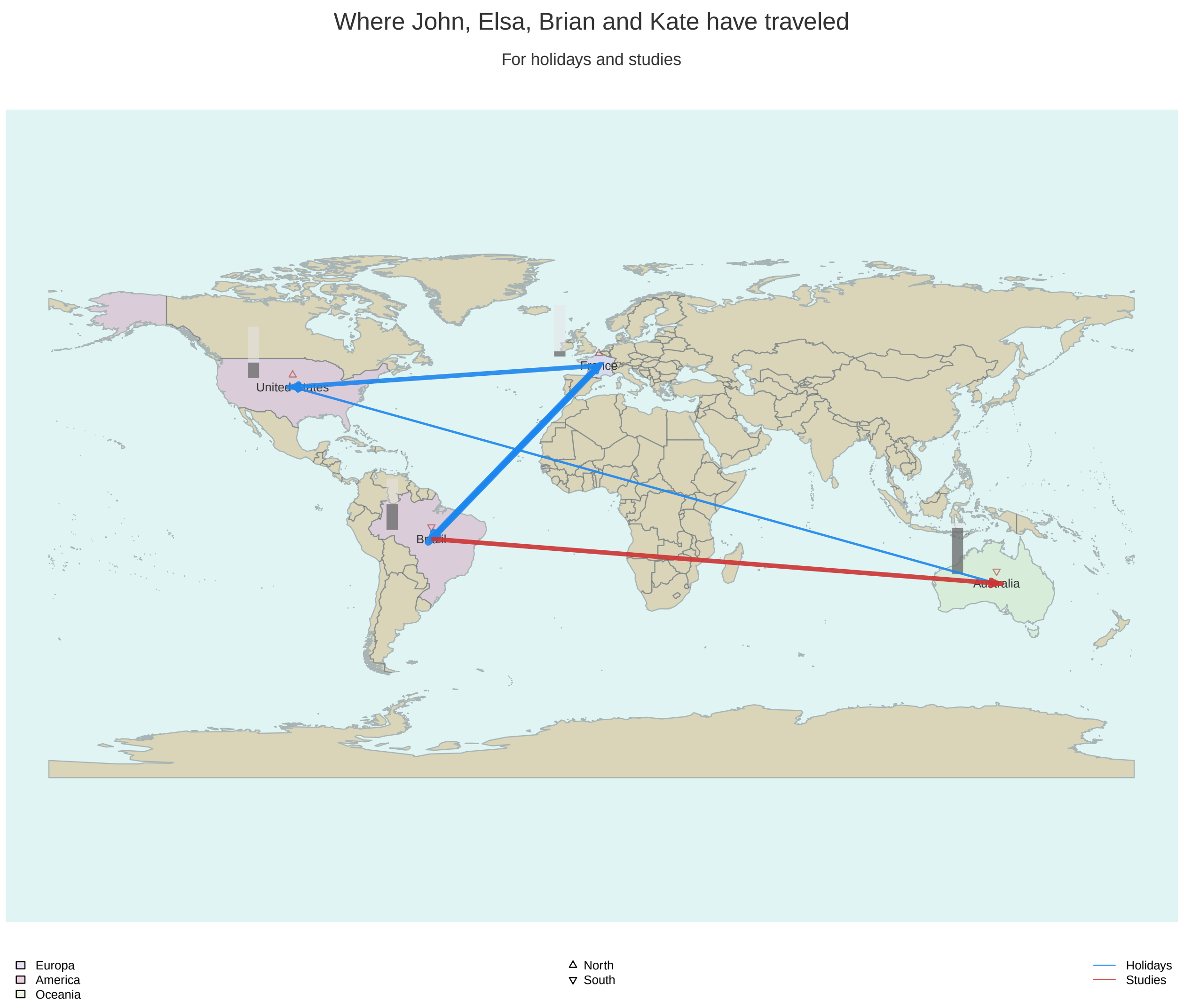
```
spnet.network.data(ex.bp, 'n1') <- matrix_int
spnet.network.data(ex.bp, 'n2') <- matrix_int
```

Matrix can of course also be imported from a datafile.

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
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)
```

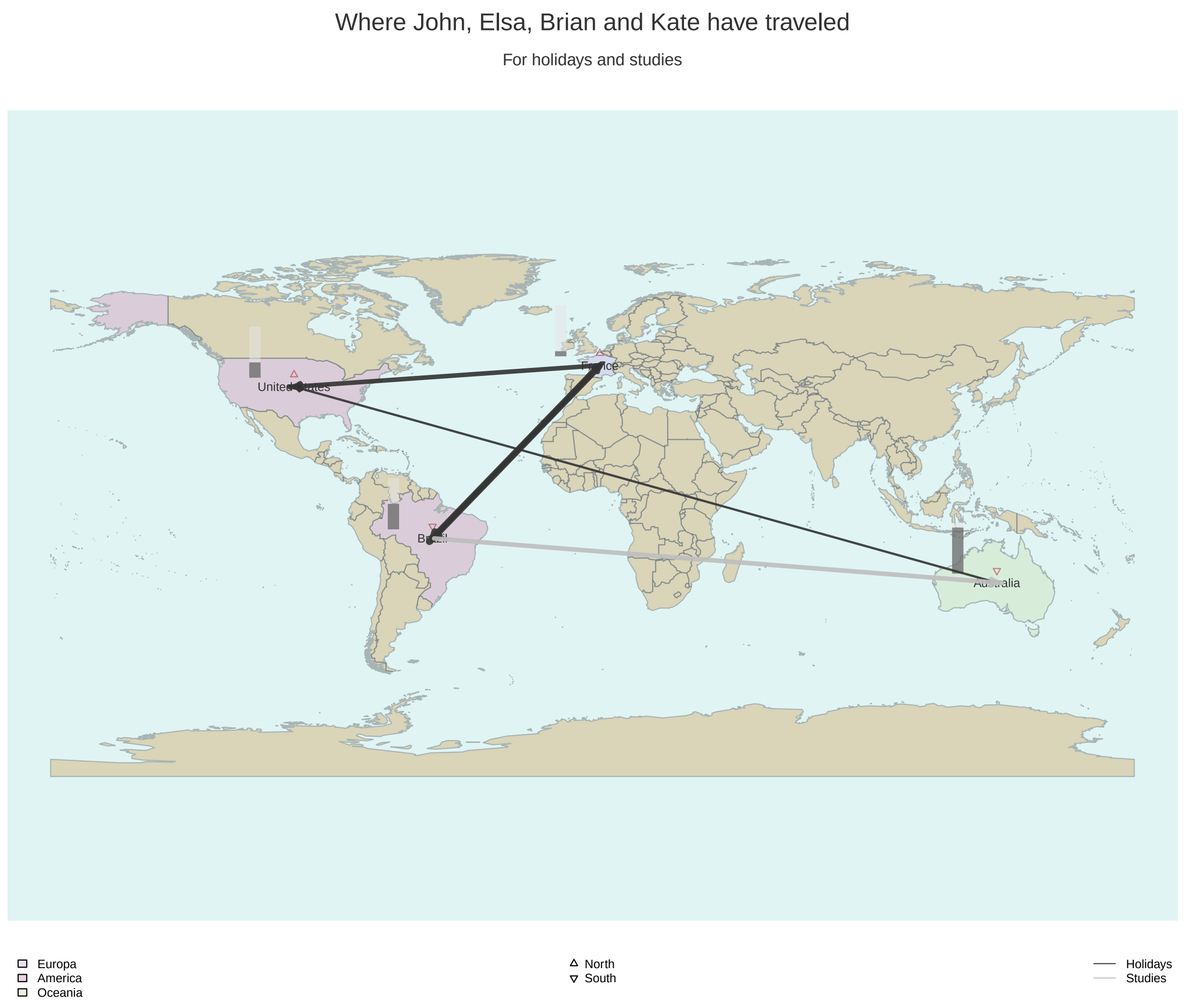


There are 'by default' colors, we change it:

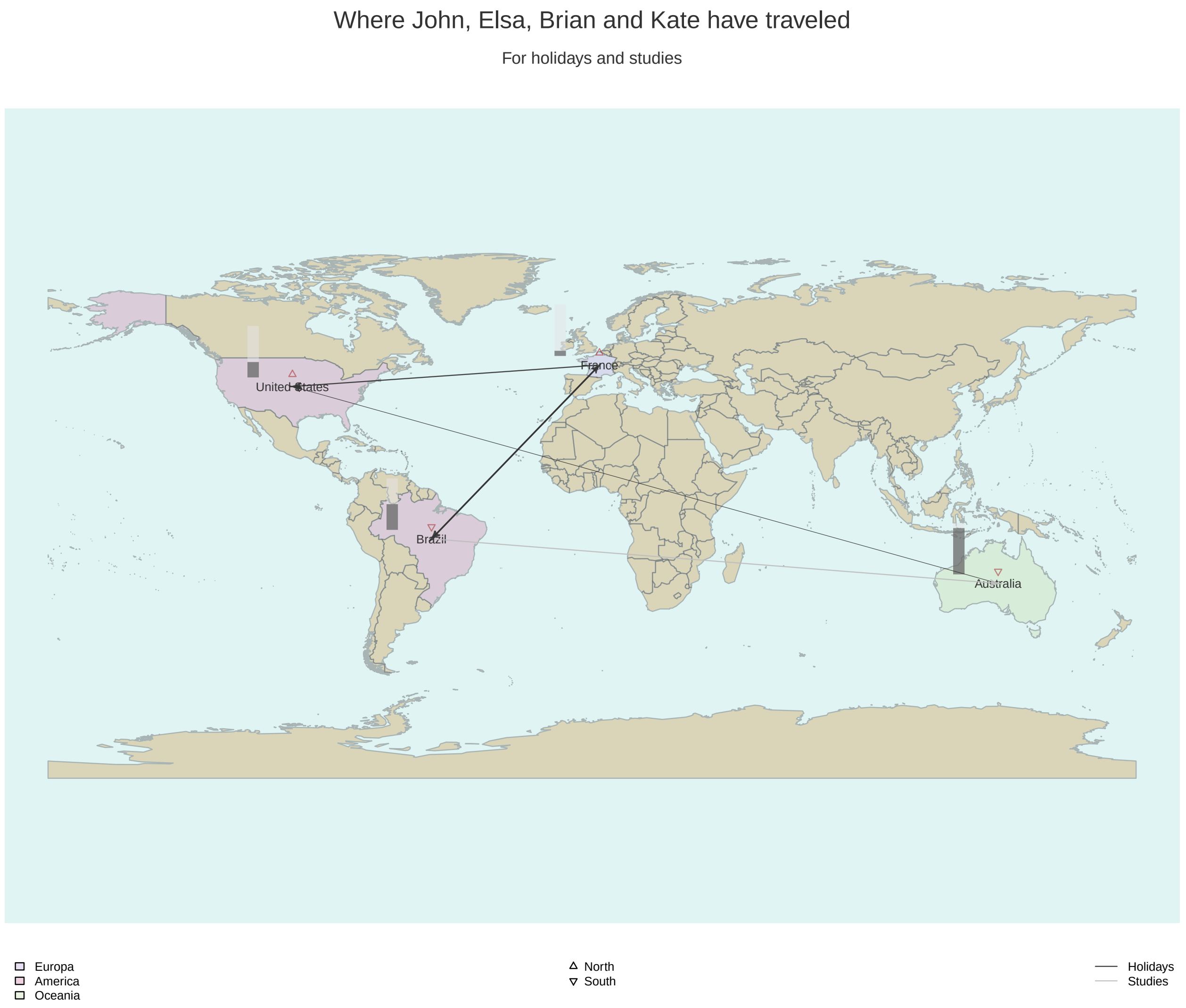
```
spnet.network.arrow.color(ex.bp, 'n1') <- '#33333366'
spnet.network.arrow.color(ex.bp, 'n2') <- 'grey'
plot(ex.bp)
```

We also reduce thinkness and length of arrows:

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


Where John, Elsa, Brian and Kate have traveled
For holidays and studies
United States
France
Brazil
Australia
Europa
America
Oceania
North
South
Holidays
Studies

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



We shift 'Holidays' a little bit upper:

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

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

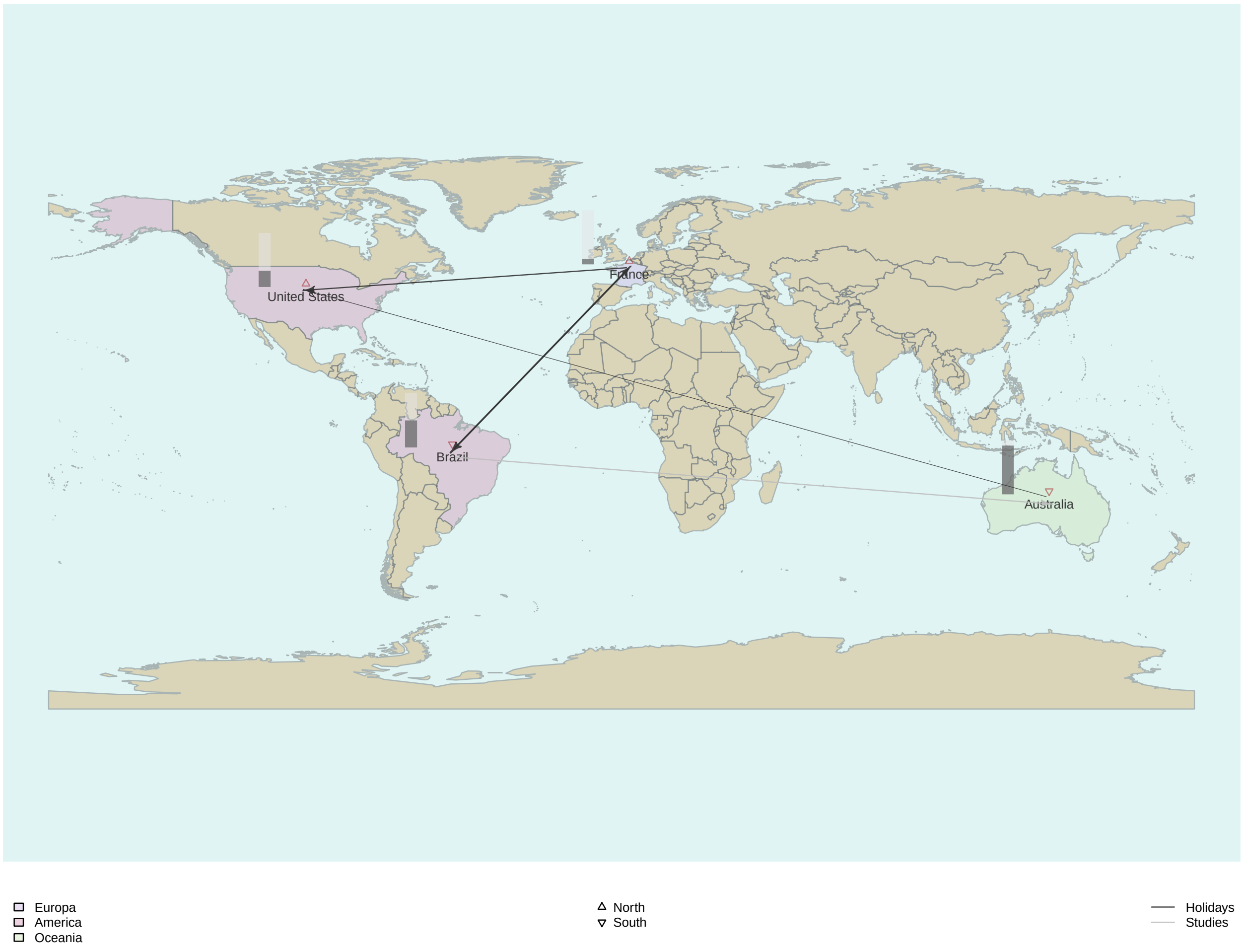
```r
spnet.network.arrow.shift.y(ex.bp, 'n2') <- -2
spnet.network.arrow.opacity(ex.bp, 'n2') <- 0.9
plot(ex.bp)
```

Title and legend

We give a title to the graph:

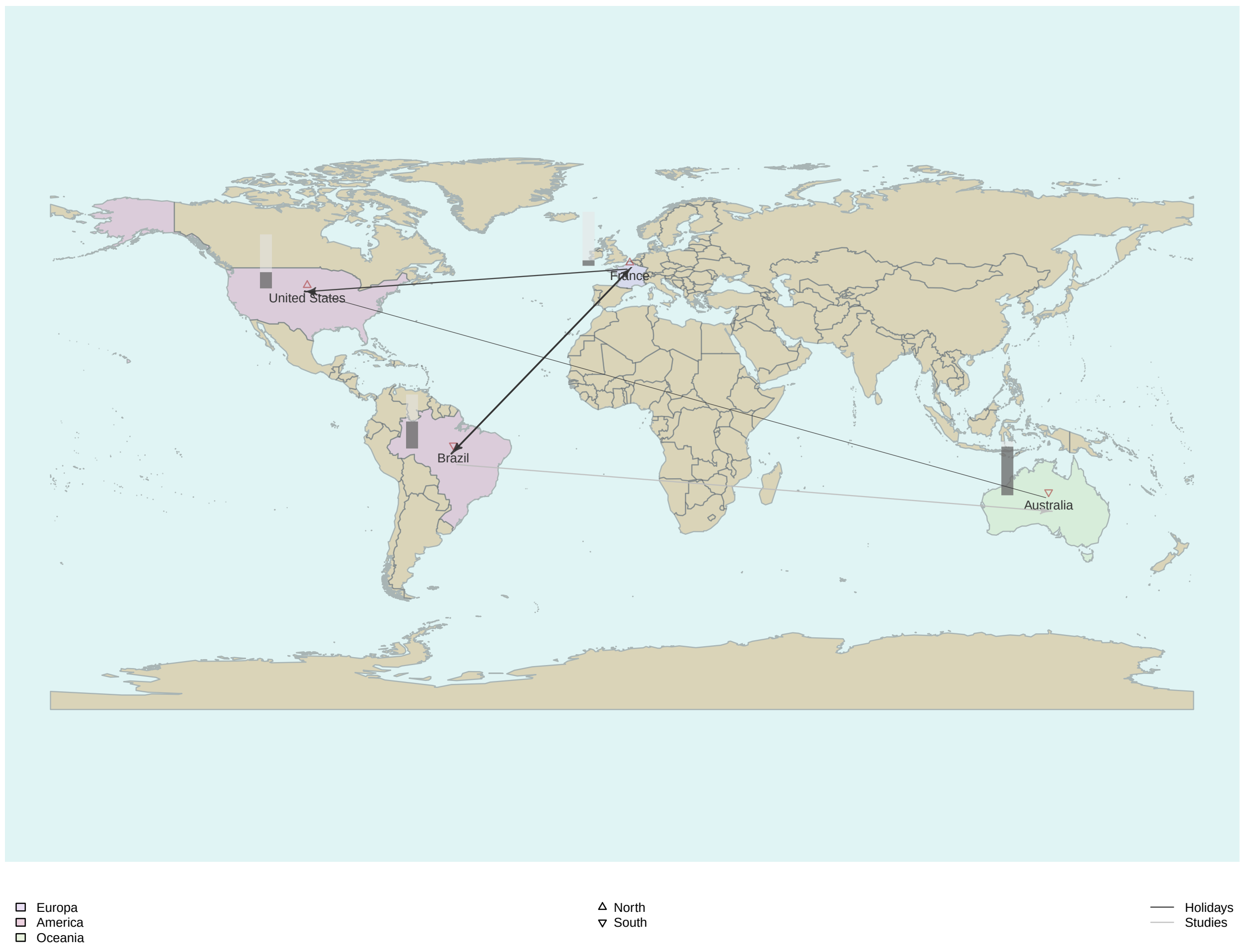
Where John, Elsa, Brian and Kate have traveled

For holidays and studies

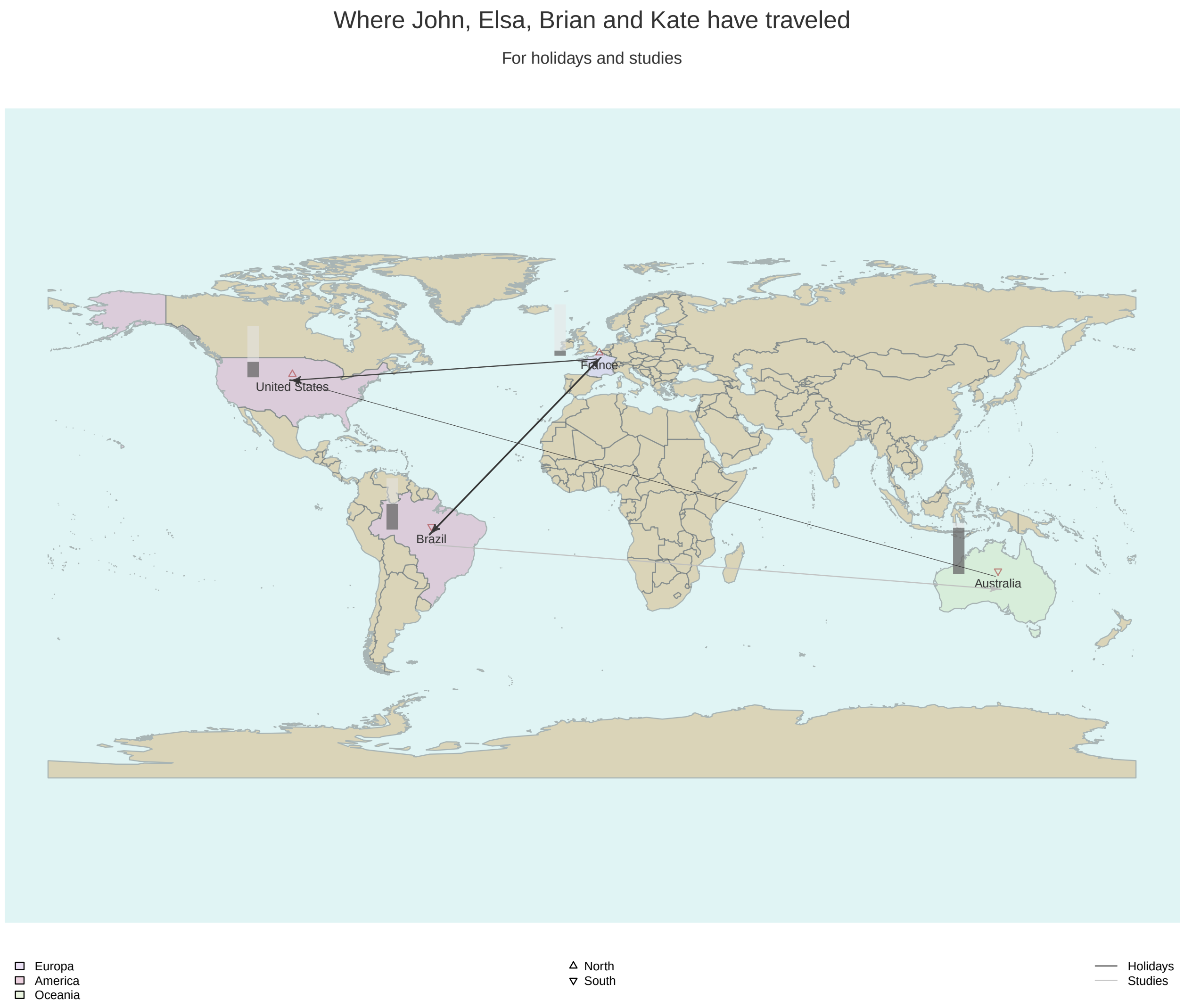


Where John, Elsa, Brian and Kate have traveled

For holidays and studies



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



We make the legend a little bit bigger, network lines more visible and colored variable legend on 3 columns:

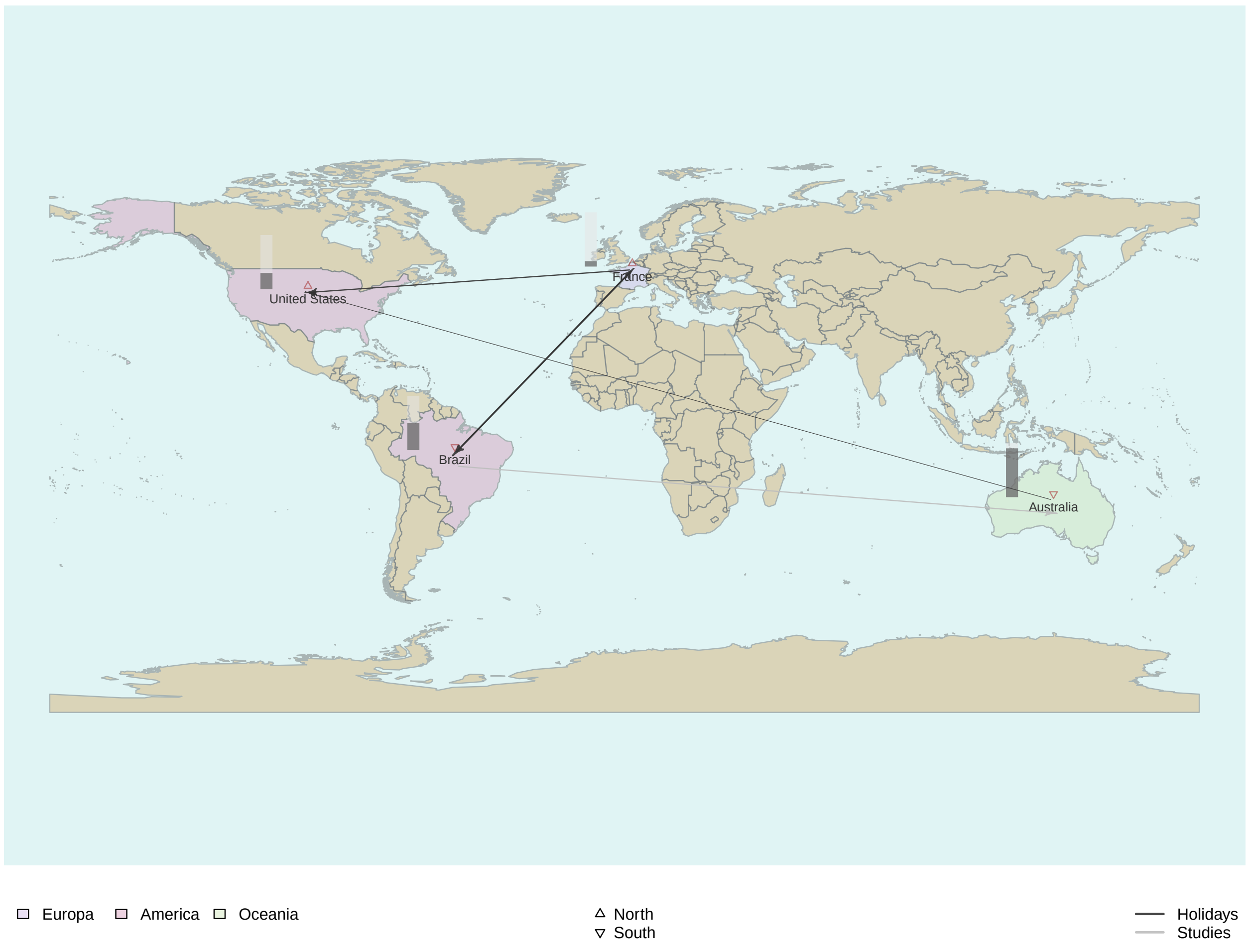
```r
spnet.legend.cex(ex.bp) <- 1.2
spnet.legend.line.width(ex.bp) <- 2
spnet.legend.ncol(ex.bp) <- 3
plot(ex.bp)
```

To hide the legend:

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

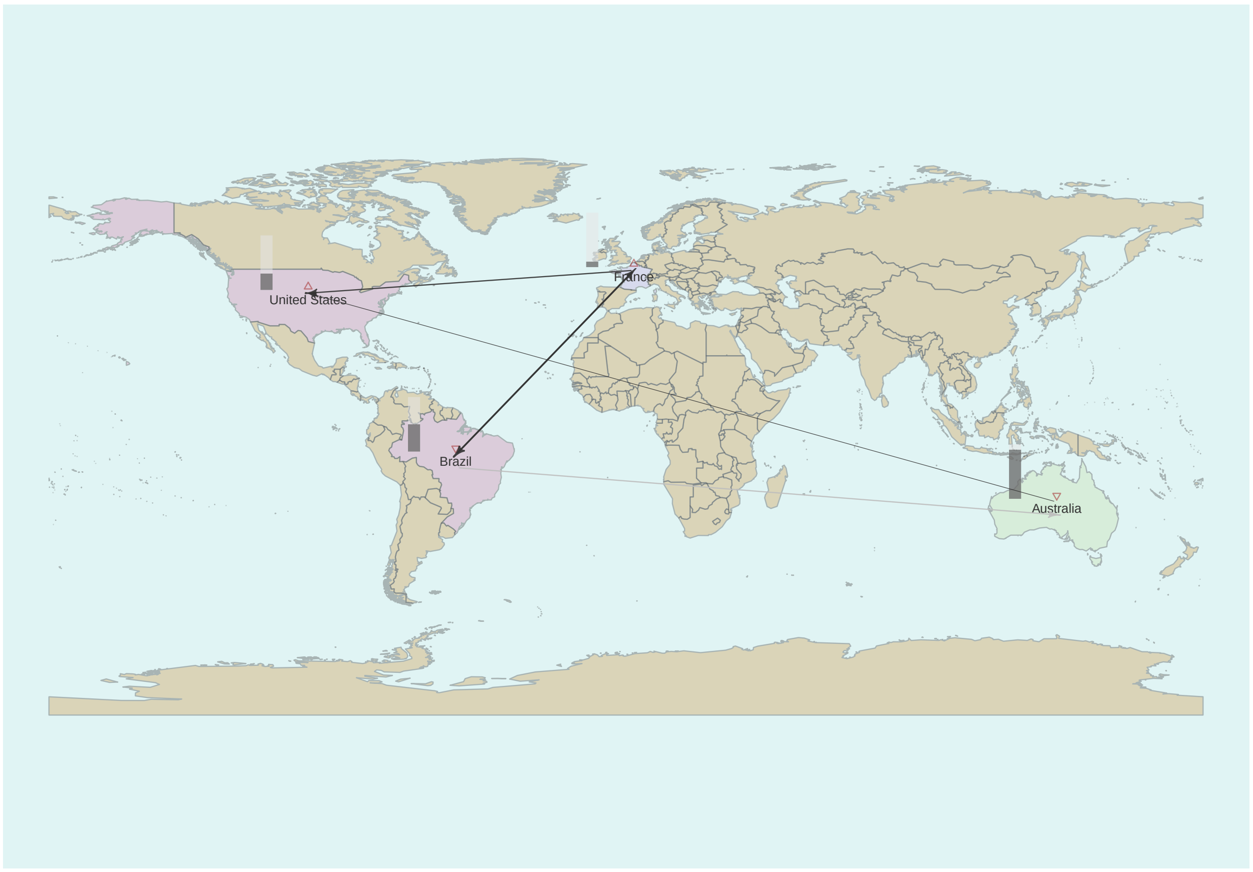
Where John, Elsa, Brian and Kate have traveled

For holidays and studies



Where John, Elsa, Brian and Kate have traveled

For holidays and studies


France
United States
Brazil
Australia