The sp Package

Jordy et Kab

20/12/2020

I- Présentation du Package SP

Le package SP fournit des classes et méthodes pour les données spatiales dans R. Il permet d'afficher des fonds de carte, d'inspecter une table attributaire, etc.. Il s'agit là des fonctions importantes qui, liées à d'autres librairies, peuvent aider à réaliser tous les types de cartes.

C'est le tout premier package général développé pour R dont le rôle est de fournir des classes et des méthodes pour les types de données spatiales. Avec pour objectifs principaux de normaliser la façon dont les données spatiales seraient traitées dans R et de permettre une meilleure interaction entre les différents packages d'analyse qui utilisent des données spatiales, son développement a commencé au début des années 2000. Cinq années plus tard, une première version est disponible sur le CRAN et fournit des classes et des méthodes nécessaires à la création des points, des lignes, des polygones et des grilles et pour opérer sur ceux-ci. Ainsi, plusieurs autres packages vont dépendre de SP. Pas moins de 350 packages d'analyse spatiale utilisent les types de données spatiales qui sont implémentés dans SP.

II- Sa structure

La structure fondamentale de tout objet spatial dans SP est la classe Spatial. Elle comporte deux lots:

- ☒ une boîte de délimitation
- ☒ un objet de classe CRS pour définir le système de référence des coordonnées

Cette structure de base est ensuite étendue, en fonction des caractéristiques de l'objet spatial (point, ligne, polygone).

Pour construire manuellement un objet spatial en sp, il y a trois étapes importantes à suivre.

1°) Créer des objets géométriques (topologie): c'est l'étape où sont générés des points, des lignes ou encore des polygones à partir d'ensemble de coordonnées. On retrouvera également des objets Ligne(ou objet Polygone) qui sont des listes de plusieurs lignes (ou plusieurs colonnes) ayant un d »nominateur commun.

2°) Créer des objets spatiaux Objet spatial* (* signifie Points, Lignes ou Polygones): cette étape ajoute la case de délimitation (automatiquement) et la case pour le système de référence de coordonnées ou CRS (à remplir de façon manuelle). Les SpatialPoints peuvent être directement générés à partir des coordonnées. Les objets SpatialLines et SpatialPolygons quant à eux sont générés en utilisant des listes d'objets.

3°) Ajouter des attributs: cette dernière est facultative. Elle permet d'ajouter un cadre de données avec des données d'attributs, ce qui transformera l'objet Spatial* en un objet Spatial*DataFrame.

Pour mieux illustrer cette présentation, voici des exemples d'usage de SP:

```
library("sp")
library("raster")
library("rgdal") #input/output, projections
```

```
## rgdal: version: 1.5-18, (SVN revision 1082)
## Geospatial Data Abstraction Library extensions to R successfully loaded
## Loaded GDAL runtime: GDAL 2.2.2, released 2017/09/15
## Path to GDAL shared files: /usr/share/gdal/2.2
## GDAL binary built with GEOS: TRUE
## Loaded PROJ runtime: Rel. 4.9.2, 08 September 2015, [PJ_VERSION: 492]
## Path to PROJ shared files: (autodetected)
## Linking to sp version:1.4-4
```

```
library("rgeos") #GEOMETRY OPS
```

```
## rgeos version: 0.5-5, (SVN revision 640)
##  GEOS runtime version: 3.5.1-CAPI-1.9.1
##  Linking to sp version: 1.4-4
##  Polygon checking: TRUE
```

```
library("spdep") #spatial dependance
```

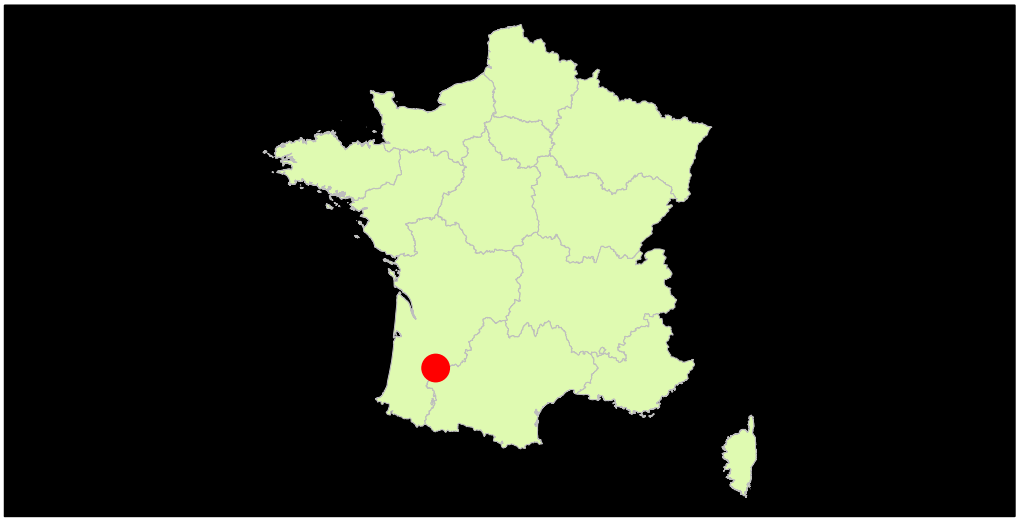
```
## Loading required package: spData
```

```
## To access larger datasets in this package, install the spDataLarge
## package with: `install.packages('spDataLarge',
## repos='https://nowosad.github.io/drat/', type='source')`
```

```
## Loading required package: sf
```

```
## Linking to GEOS 3.5.1, GDAL 2.2.2, PROJ 4.9.2
```

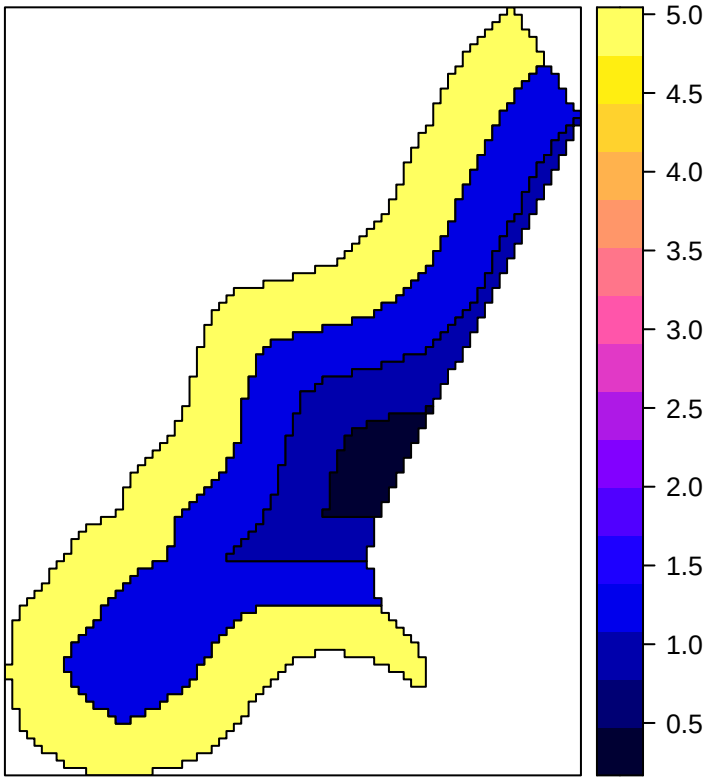
```
# Exemple 1: Afficher un point par un couple de valeurs longitude/latitude

adm_fr <- getData('GADM', country='FRA', level=1)
# GADM pour indiquer que je veux des contours administratifs
# FRA est le code de la France*
#   1 indique la granularité de l'affichage, ici la région
plot(adm_fr,lwd=0.5, border="grey", col="#DFFAB1", bg="black");box()
points(0,44, col="red", pch=16, cex=2) # coordonnées du point à afficher
```



```
# Exemple 2: Construire deux polygones et les tracer

x1 <- c(0, 72,72,0) ; y1 <- c(0,0,40,40) # Coordonnées x et y d'une première salle
c1 <- data.frame(x1, y1)
```

```
p1 <- Polygon(coords = c1, hole = F);
P1 <- Polygons(srl = list(p1), ID = "Appart de Kab") # Conversion en polygone

x2 <- c(0, 76,76,0) ; y2 <- c(72,72, 148,148) # Coordonnées x et y d'une autre salle
c2 <- data.frame(x2, y2)
p2 <- Polygon(coords = c2, hole = F);
P2 <- Polygons(srl = list(p2), ID = "Appart de sa copine") # Conversion en polygone
SP <- SpatialPolygons(Srl = list(P1,P2)) # Création del'objet spatial
temperatures = c(18, 25) # définition d'une température d'affichage pour chaque salle
temperatures[temperatures<19.5] <- "#0080FF"
temperatures[temperatures>22] <- "#FE2E2E"
plot(SP,col=temperatures,border = "white") # Affichage selon les températures
```

```
#Exemple 3: Charger des données et les mettre en forme

data("meuse.grid") # Meuse.grid est un jeu de données intégré
coordinates(meuse.grid) <- ~x+y
gridded(meuse.grid) <- TRUE
data("meuse")
coordinates(meuse) <- ~x+y
i = cut(meuse.grid$dist, c(0,.25,.5,.75,1), include.lowest = TRUE)
library("rgeos")
# Découper la Meuse en fonction des valeurs de i
ab = gUnaryUnion(as(meuse.grid, "SpatialPolygons"), i) # On peut essayer aussi meuse.grid$part.a à la p
x = aggregate(meuse[c("cadmium")], ab, mean)
spplot(x)
```




```r
#Exemple 4: Synthèse sur chargement de données,création d'objets et mise en forme

# ETAPE 0 - Chargement des données
adm_gr <- getData('GADM', country='GRC', level=1)
# Limiter la région chargée pour réduire le temps d'affichage
adm <- adm_gr[match(toupper(adm_gr$NAME_1[7]),toupper(adm_gr$NAME_1)),]# Péloponnèse
# Étape 1 - Tracer un fond bleu "marin"
plot.new()
par(mar=c(4,4,3,4),bg="transparent") # définition des marges
# on pré-trace la carte pour programmer ses limites d'affichage automatiquement
plot(adm,xlim=c(23.05,23.20),ylim=c(36.422,36.53),col="white",border="white",lwd=50) ; box()
# FOND DE MER : tracé simple d'un polygone
polygon(x = c(20, 30, 30, 20, 20), y = c(-10, -10, 50, 50, -10), col = "#bbe6fb", border = "transparent
# Étape 2 - Délimiter par des contours
plot(adm,col="white",border="#34bdf2",lwd=7,add=T)
#Étape 3 - Afficher la zone avec une couleur
plot(adm, col="#FAFAB1", border="grey", lwd=1,add=T)
```

Sources