Le package sp

Cyril Bernard (CEFE-CNRS)

25 janvier 2018

1. Présentation du package sp

1.1. sp: Classes and Methods for Spatial Data

Le package sp propose des structures de données pour gérer des données spatiales dans R, un peu à la manière de ce que l'on trouve dans les SIG

- des données ponctuelles, linéaires, surfaciques (= données vectorielles)
- des données matricielles (= données raster)

C'est le principal package pour la gestion de données vectorielles ! C'est pourquoi beaucoup d'autres packages dépendent de sp (exemples: rgdal, adehabitat, biomod2, maptools).

Pour les données raster, on utilise plus fréquemment le package ... raster (dont on parlera à la séance 2).

1.2. Ce que contient et ne contient pas sp

Les classes proposées par sp permettent de gérer la **partie géométrique** et la **partie attributaire** des données spatiales, ainsi que des métadonnées essentielles telles que le **système de coordonnées** employé dans les données. Il est possible de créer de toutes pièces des entités géographiques avec du code R !

En revanche, sp propose peu de fonctions pour **analyser** les données spatiales : ce sont d'autres packages dépendant de sp qui offrent cela.

Les entrées/sorties depuis ou vers des fichiers SIG (exemple: .shp, .kml, .gpx, .tif, .asc) ne sont gérées par sp. On utilise d'autres packages tel que rgdal ou maptools.

1.3. Interactions sp <−> raster

Le package raster est compatible avec le package sp, notamment lorsqu'il faut croiser des données vectorielles et raster.

Exemples :

- extraire la valeur d'un raster pour N points (`raster::extract`)
- calculer la moyenne, l'écrat type, etc. des pixels recouverts par un polygone (`raster::zonal`)

1.4. sp, rgeos, sf

Le package **rgeos** a longtemps été, en complément de sp, le seul moyen de faire certains calculs tels que : *zones tampons, intersection, calcul de longueur et de surface.* La logique de rgeos est difficile à appréhender, son efficacité n'est pas optimale.

Heureusement le package **sf** est arrivé ! Il est beaucoup plus rapide et simple d'utilisation. Mais il introduit un nouveau modèle de données différent de sf.

1.5. Conclusion

- Pour manipuler des données raster, les commandes de base sont dans le package **raster**
- Pour manipuler des données vectorielles, le package **sf** est simple et efficace
- Le package **sp** reste incontournable car ses classes de données spatiales se retrouvent dans d'autres packages tierces ...

1.6. Livres, sites

- ASDAR book (http://www.asdar-book.org/) à emprunter ou consulter à la bibliothèque du CEFE
- Geocomputation with R (https://bookdown.org/robinlovelace/geocompr/)

2. COURS : les classes de données spatiales dans sp

2.1. Les classes Spatial*

- SpatialPoints : pour les données spatiales ponctuelles
- SpatialLines : pour les données spatiales linéaires
- SpatialPolygons : pour les données spatiales surfaciques
- SpatialPixels et SpatialGrid : pour les données spatiales matricielles ou raster (sur une grille)

2.2. Créer un objet SpatialPoints à partir d'une matrice de coordonnées GPS

Définissons un `data.frame` object with 5 rows and 4 columns. The numeric vector lon and lat give the GPS coordinates of 5 points, in decimal degrees (WGS84) .

```
library(sp)
name <- c("Agropolis","Bois de Montmaur","CEFE","Station météo TE","FDS Bâtiment 4","Statue Peyrou")
lon <- c(3.86921,3.86898,3.86450,3.86306,3.86282,3.87093)
lat <- c(43.64541,43.64266,43.63881,43.63885,43.63464,43.61165)
color <- c("blue","green","blue","blue","blue","green")
df <- data.frame(name, lon, lat, color)
```

2.3. La classe SpatialPoints

La classe SpatialPoints est une structure de données pour stocker des points : seulement la partie "spatiale", pas la partie "attributs". Pour construire un objet SpatialPoints, nous avons besoin de :

- une matrice à 2 colonnes (avec des coordonnées X Y, ou "longitude latitude")
- si possible, un objet CRS généré avec la **définition proj 4** du système de coordonnées. Ici la définition vient du site epsg.io : http://epsg.io/4326.

```
matcoords <- as.matrix(df[,c("lon","lat")])
spts <- SpatialPoints(matcoords, proj4string = CRS("+proj=longlat +datum=WGS84 +no_defs"))
# the following proj4string definition with EPSG ID is equivalent to the explicit definition ...
spts <- SpatialPoints(matcoords, proj4string = CRS("+init=EPSG:4326"))
slotNames(spts)
```

```
## [1] "coords"      "bbox"        "proj4string"
```

2.4. Relation d'héritage entre la classe Spatial et SpatialPoints

La classe S4 `SpatialPoints` hérite de la classe `Spatial` et l'étend

```
library(sp)
getClass("SpatialPoints")
```

```
## Class "SpatialPoints" [package "sp"]
##
## Slots:
##
## Name:       coords        bbox proj4string
## Class:      matrix      matrix         CRS
##
## Extends: "Spatial"
##
## Known Subclasses:
## Class "SpatialPointsDataFrame", directly
## Class "SpatialPixels", directly
## Class "SpatialPixelsDataFrame", by class "SpatialPixels", distance 2
```

2.5. La classe Spatial

La classe S4 `Spatial` est la plus générique. Elle a 2 "slots" : bbox (matrix) et proj4string (CRS)

```
getClass("Spatial")
```

```
## Class "Spatial" [package "sp"]
##
## Slots:
##
## Name:         bbox proj4string
## Class:      matrix         CRS
##
## Known Subclasses:
## Class "SpatialPoints", directly
## Class "SpatialMultiPoints", directly
## Class "SpatialGrid", directly
## Class "SpatialLines", directly
## Class "SpatialPolygons", directly
## Class "SpatialPointsDataFrame", by class "SpatialPoints", distance 2
## Class "SpatialPixels", by class "SpatialPoints", distance 2
## Class "SpatialMultiPointsDataFrame", by class "SpatialMultiPoints", distance 2
## Class "SpatialGridDataFrame", by class "SpatialGrid", distance 2
## Class "SpatialLinesDataFrame", by class "SpatialLines", distance 2
## Class "SpatialPixelsDataFrame", by class "SpatialPoints", distance 3
## Class "SpatialPolygonsDataFrame", by class "SpatialPolygons", distance 2
```

2.6. Les classes Spatial*DataFrame

- SpatialPointsDataFrame : pour les données spatiales ponctuelles et leurs attributs
- SpatialLinesDataFrame : pour les données spatiales linéaires et leurs attributs
- SpatialPolygonsDataFrame : pour les données spatiales surfaciques et leurs attributs

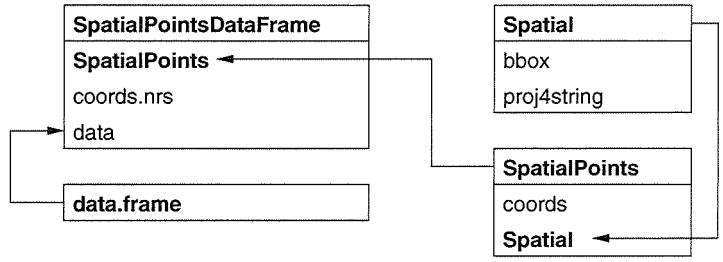

Fig. 2.2 Spatial points classes and their slots; *arrows* show subclass extensions

Figure 1:

- SpatialPixelsDataFrame et SpatialGridDataFrame : pour des données spatiales matricielles ou raster accompagnées de leur(s) valeur(s)

2.7. Building a SpatialPointDataFrame from a data.frame with coordinates

It can be achieved with coordinates method with the name of X and Y columns.

```
spts_df <- df
# turn a data.frame into a SpatialPointsDataFrame by providing X Y columns
coordinates(spts_df) <- c("lon","lat")
# ceci fonctionne aussi :
spts_df <- df
coordinates(spts_df) <- ~lon+lat
# define the CRS (optional)
proj4string(spts_df) <- CRS("+init=EPSG:4326")
slotNames(spts_df)
```

```
## [1] "data"        "coords.nrs"  "coords"      "bbox"        "proj4string"
```

2.8. Relation d'héritage entre les classes Spatial, *SpatialPoints* et SpatialPoints-DataFrame

Le schema suivant tiré du "ASDAR Book" (http://www.asdar-book.org/), p.35 nous montre la composition de la class SpatialPoints.

```
library(sp)
getClass("SpatialPoints")
```

```
## Class "SpatialPoints" [package "sp"]
##
## Slots:
##
## Name:       coords        bbox proj4string
## Class:      matrix      matrix         CRS
##
## Extends: "Spatial"
##
## Known Subclasses:
## Class "SpatialPointsDataFrame", directly
```

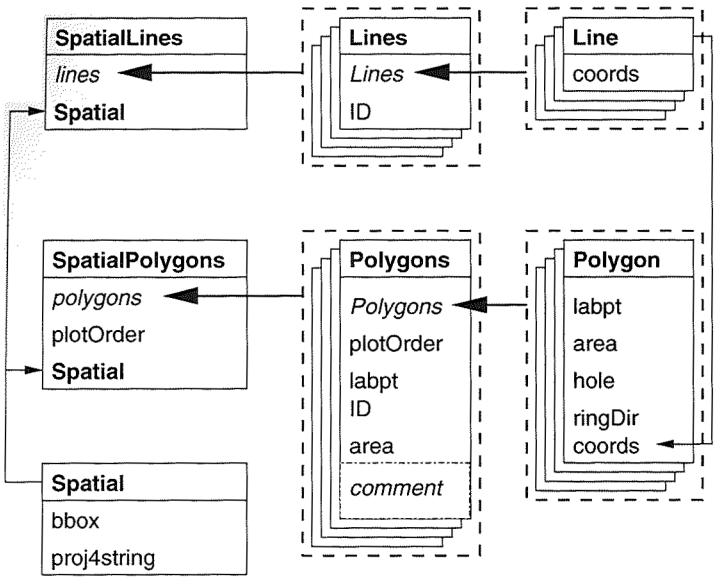

Fig. 2.4 SpatialLines and SpatialPolygons classes and slots; *thin arrows* show subclass extensions, *thick arrows* the inclusion of lists of objects

Figure 2:

```
## Class "SpatialPixels", directly
## Class "SpatialPixelsDataFrame", by class "SpatialPixels", distance 2
```

2.9. SpatialLines et SpatialPolygons

Relier les points ! Créer les objets SpatialLines et SpatialLinesDataFrame avec R. Le schema suivant tiré du "ASDAR Book" (http://www.asdar-book.org/), p.40 nous montre la composition de la class SpatialPolygons et SpatialLines

A **SpatialLines** object can be made from a **list of Lines** objects. A **Lines** object is a **list of Line** objects . A **Line object** is made of a **matrix of coordinates**, just as a set of ordered points.

Lines in R is like a *Polyline* feature in a Shapefile, or a *MULTILINESTRING* feature in WKT notation : https://en.wikipedia.org/wiki/Well-known_text#Geometric_objects)

```
# build 2 Lines object with ID slot = L1 and L2
matcoords1 <- as.matrix(df[,c("lon","lat")])
matcoords2 <- cbind(runif(5, -0.001, 0.001) + 3.8676, runif(5, -0.001, 0.001) + 43.6423)
line_1 <- Line(matcoords1)
line_2 <- Line(matcoords2)
lines_1 <- Lines(list(line_1), "L1")
lines_2 <- Lines(list(line_2), "L2")
splines <- SpatialLines(list(lines_1, lines_2))
str(splines)
```

```
## Formal class 'SpatialLines' [package "sp"] with 3 slots
##   ..@ lines      :List of 2
##   .. ..$ :Formal class 'Lines' [package "sp"] with 2 slots
```

```
##   .. .. .. ..@ Lines:List of 1
##   .. .. .. .. ..$ :Formal class 'Line' [package "sp"] with 1 slot
##   .. .. .. .. .. .. ..@ coords: num [1:6, 1:2] 3.87 3.87 3.86 3.86 3.86 ...
##   .. .. .. .. .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. .. .. .. .. ..$ : NULL
##   .. .. .. .. .. .. .. .. ..$ : chr [1:2] "lon" "lat"
##   .. .. .. ..@ ID   : chr "L1"
##   .. ..$ :Formal class 'Lines' [package "sp"] with 2 slots
##   .. .. .. ..@ Lines:List of 1
##   .. .. .. .. ..$ :Formal class 'Line' [package "sp"] with 1 slot
##   .. .. .. .. .. .. ..@ coords: num [1:5, 1:2] 3.87 3.87 3.87 3.87 3.87 ...
##   .. .. .. ..@ ID   : chr "L2"
##   ..@ bbox       : num [1:2, 1:2] 3.86 43.61 3.87 43.65
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:2] "x" "y"
##   .. .. ..$ : chr [1:2] "min" "max"
##   ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
##   .. .. ..@ projargs: chr NA
```

A **SpatialLinesDataFrame** object is the combination between a **SpatialLines** object and a **data.frame**. Use the **ID slot** from the SpatialLines object and the **row names** from the data.frame to make them match.

```
# build a data.frame object with 2 columns and ID as the rows names.
NAME=c("LINE1", "RANDOM2")
LENGTH_M = SpatialLinesLengths(splines, longlat=T) * 1000
df_demo <- data.frame(NAME, LENGTH_M)
row.names(df_demo) <- c("L1","L2")
splines_df <- SpatialLinesDataFrame(splines, df_demo)
## save the SpatialLinesDataFrame as a shapefile
#writeLinesShape(splines_df, fn="some_lines")
```

Lire un shapefile avec rgdal

Ecrire un shapefile avec rgdal

COMMENT FAIRE POUR ...

Comment lire des coordonnées en degrés minutes secondes ?

Comment calculer une matrice de distance entre des points ?

Comment convertir des données spatiales d'un système de coordonnées vers un autre ?

Transforming coordinates from a system to another require the `rgdal` package. `rgdal` provides drivers for an important number of raster and vector formats (see all the formats on the website of the GDAL library and its OGR sub-library). It also provides the **spTransform** function that makes possible to transform coordinates. It is possible to apply the **spTransform** on any `Spatial*` or `Spatial*DataFrame` class. The system coordinates of the input object must have been defined with **proj4string** parameter. When calling **spTransform** we only have to specify output coordinate system.

```r
# check input CRS
proj4string(spts_df)
```

```
## [1] "+init=EPSG:4326 +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"
```

```r
# transformation to RGF93 / Lambert93
spts_df_l93 <- spTransform(spts_df, CRS("+init=EPSG:2154"))
spts_df_l93@coords
```

```
##        lon     lat
## 1 770148.4 6283316
## 2 770133.2 6283010
## 3 769776.3 6282578
## 4 769660.0 6282581
## 5 769645.8 6282113
## 6 770328.5 6279565
```

Comment sélectionner les points qui intersectent un polygone ?

Comment enregistrer vos données en KML pour les visualiser dans Google Earth ?

Comment visualiser vos données sur un fond de carte OpenStreetMap avec Leaflet ?

Exercice

jointures restaurant bar : densité de bars X grille arret de tramway situé à castelnau le lez arbres remarquables X parcs et jardin

iris : choroplethe

transformation WGS84 > L93