Fiche TD avec le logiciel  : ter5

Gestion de données spatialisées vectorisées

S. Dray

La fiche présente un état des lieux concernant la gestion de données spatialisées dans R. On s'intéresse ici aux données vectorisées en s'appuyant sur l'utilisation des librairies `ade4`, `adehabitat`, `spdep`, `maptools` et `sp`. Les outils de cartographie sont également présentés.

Table des matières

1 Introduction **2**

2 Le grand bazar **3**

3 Interface SIG **4**

4 La librairie sp **6**
4.1 Classes 6
4.2 Méthodes 8
4.3 Quelques fonctions utiles 9
4.4 Conversions, importation et exportation 10

5 Cartographie **10**
5.1 Représentations surfaciques 11
5.2 Représentations ponctuelles 14
5.3 Courbes de niveaux 16

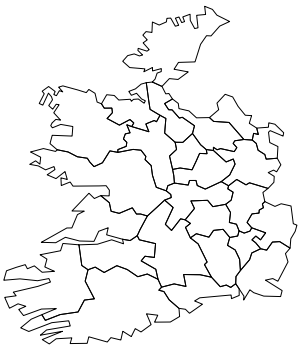
Références **17**

1 Introduction

Une grande partie des données acquises, en génétique, en écologie ou en biologie des populations est spatialisée. Les enregistrements de l'espace lui-même sont multiples.

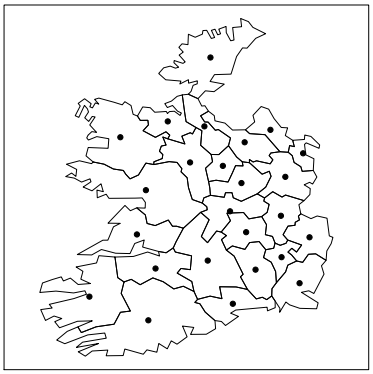
Enregistrements surfaciques : la mesure porte sur une surface bornée par une frontière. C'est le support des données socio-économiques. Un des articles fondateurs de ce domaine traite des comtés d'Irlande :

```
library(ade4)
data(irishdata)
names(irishdata)
 [1] "area"          "county.names" "xy"           "tab"          "contour"
 [6] "link"          "area.utm"     "xy.utm"       "link.utm"     "tab.utm"
[11] "contour.utm"
area.plot(irishdata$area.utm)
```



Enregistrements ponctuels : la mesure se réfère à deux coordonnées (x, y). On passe des données surfaciques aux données ponctuelles en choisissant un point particulier par unités :

```
area.plot(irishdata$area.utm)
s.label(irishdata$xy.utm, clab = 0, add.plot = T, cpoint = 2)
```



On peut également disposer de données linéaires (e.g., une rivière). Ce type de données étant assez rare, il ne sera pas évoqué dans ce document.

2 Le grand bazar

Du fait du mode de développement de R, un grand nombre de librairies liées à l'analyse de données spatiales a été implémenté parallèlement. Pratiquement, chaque librairie possède une structure propre pour gérer ce type de données. Il en résulte un grand nombre de classes, de fonctions de conversions et de difficultés pour l'utilisateur qui doit savoir jongler entre les différents formats. On présente ici un éventail non exhaustif de ces différentes classes pour les données surfaciques.

Dans la librairie `ade4`, les enregistrements surfaciques sont stockés sous la forme de `data.frame` de type `area`. Ce format est une simple adaptation de ce qui était disponible dans le logiciel ADE4 [Thioulouse et al., 1997]. Le `data.frame` contient 3 colonnes : une contenant les noms des entités géographiques et deux autres avec les coordonnées en x et y de chaque sommet.

```
head(irishdata$area.utm)
```

```
    reg      x       y
1 Carlow 240.62 5885.61
2 Carlow 245.93 5883.32
3 Carlow 251.98 5885.42
4 Carlow 253.77 5888.82
5 Carlow 268.26 5883.61
6 Carlow 262.92 5876.30
```

Pour un polygone avec n sommets, on a $n+1$ lignes. En effet, la dernière ligne d'un polygone doit être identique à la première afin de "refermer le polygone" :

```
table(irishdata$area.utm[, 1])
```

```
   Carlow     Cavan     Clare      Cork   Donegal    Galway     Kerry   Kildare
       19        27        28        50        59        54        39        25
 Kilkenny   Laoghis   Leitrim  Limerick  Longford     Louth      Mayo     Meath
       20        20        21        23        14        19        62        32
 Monaghan    Offaly Roscommon     Sligo Tipperary Waterford Westmeath   Wexford
       17        27        33        33        30        24        21        24
  Wicklow
       26
```

```
irishdata$area.utm[c(1, 19), ]
```

```
     reg      x       y
1  Carlow 240.62 5885.61
19 Carlow 240.62 5885.61
```

Dans `adehabitat`, on a le même type de structure de données qui définit une classe d'objet :

```
class(irishdata$area.utm)
```

```
[1] "data.frame"
```

```
library(adehabitat)
ir.area <- as.area(irishdata$area.utm)
class(ir.area)
```

```
[1] "area"       "data.frame"
```

Enfin, on peut évoquer les classes `polylist` et `Map` des librairies `spdep` et `maptools`. Les données sont stockées sous la forme de liste. Ces classes gèrent des structures de données plus complexes comme les lacs (trou dans un polygone) ou les îles (plusieurs polygones associés à une entité). De plus, la classe `Map` gère également les données attachées aux entités géographiques.

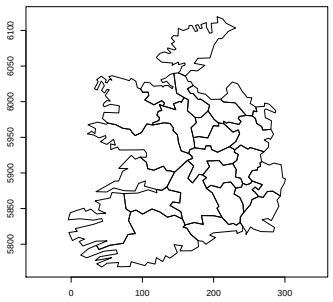
On dispose de fonctions de conversions pour naviguer entre les différentes structures de données. Dans `ade4`, les fonctions `area2poly` et `poly2area` permettent

de naviguer entre les objets de type `area` et la classe `polylist`. Dans `maptools`, on passe d'un objet de la classe `Map` à la classe `polylist` par la fonction `Map2poly`.

```
library(maptools)
ir.poly <- area2poly(irishdata$area.utm)
class(ir.poly)
```

```
[1] "polylist"
```

```
plot(ir.poly)
```



3 Interface SIG

La plupart des données spatialisées sont stockées dans des systèmes d'information géographique (SIG). Plusieurs librairies permettent d'interfacer R avec ce type de logiciel. On s'intéresse principalement aux fonctions d'importation / exportation de fichiers de type *shapefile* (format du logiciel Arcview).
Dans `maptools`, on a plusieurs solutions pour le faire. La première nécessite la librairie `shapefiles`. On prend l'exemple du découpage de la France en départements disponible sur le site de l'IGN (`http://www.ign.fr/telechargement/MPro/produit/GEOFLA_Dep/GEOFLA-dep-L2-SF.ZIP`) dont on a fait une copie ici `http://pbil.univ-lyon1.fr/R/donnees/francedep/`. Pour télécharger les fichiers :

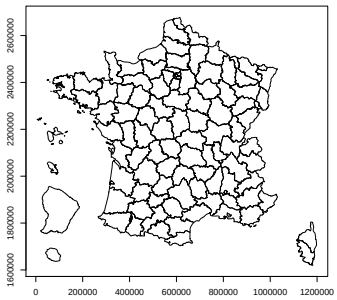
```
download.file("http://pbil.univ-lyon1.fr/R/donnees/francedep/dep_france_dom.shp",
    "dep_france_dom.shp", mode = "wb")
download.file("http://pbil.univ-lyon1.fr/R/donnees/francedep/dep_france_dom.shx",
    "dep_france_dom.shx", mode = "wb")
download.file("http://pbil.univ-lyon1.fr/R/donnees/francedep/dep_france_dom.dbf",
    "dep_france_dom.dbf", mode = "wb")
```

Le fichier est lu par la fonction `read.shapefile` avant d'être transformé en `polylist` :

```
library(shapefiles)
fr.shp <- read.shapefile("dep_france_dom")
class(fr.shp)
```

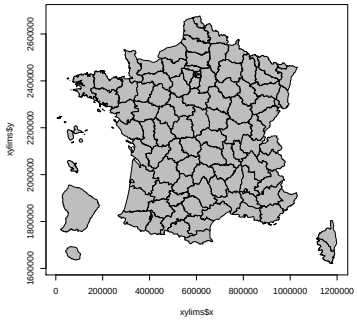
```
[1] "list"
```

```
fr.poly <- shape2poly(fr.shp)
plot(fr.poly)
```



La seconde alternative est basée sur la bibliothèque de fonctions C `shapelib` (`http://shapelib.maptools.org/`). La fonction `read.shape` renvoie un objet de la classe `Map` :

```
 fr.Map <- read.shape("dep_france_dom.shp")
Shapefile type: Polygon, (5), # of Shapes: 100
 class(fr.Map)
[1] "Map"
 names(fr.Map)
[1] "Shapes"   "att.data"
 head(fr.Map$att.data)
  ID_GEOFLA CODE_CHF_L       NOM_CHF_L X_CHF_LIEU Y_CHF_LIEU X_CENTROID Y_CENTROID
1        49        053 BOURG-EN-BRESSE       8231      21379       8244      21380
2       812        408            LAON       6932      25081       6929      25085
3      1418        190        MOULINS        6763      21743       6758      21740
4      1603        070 DIGNE-LES-BAINS       9124      19067       9124      19065
5      1802        061             GAP       8973      19579       8961      19602
6      2002        088            NICE       9977      18680       9952      18690
                NOM_DEPT CODE_REG                NOM_REGION CODE_DEPT
1                    AIN       82                RHONE-ALPES        01
2                  AISNE       22                   PICARDIE        02
3                 ALLIER       83                   AUVERGNE        03
4 ALPES-DE-HAUTE-PROVENCE       93 PROVENCE-ALPES-COTE-D'AZUR        04
5            HAUTES-ALPES       93 PROVENCE-ALPES-COTE-D'AZUR        05
6         ALPES-MARITIMES       93 PROVENCE-ALPES-COTE-D'AZUR        06
 plot(fr.Map)
```



Comme indiqué auparavant, la classe `Map` contient l'information géographique (`Shapes`) et les données associées (`att.data`).

Pour exporter, la fonction `write.polylistShape` permet d'associer un tableau de données de type `data.frame` à une `polylist` et de les sauvegarder dans un fichier de type `shapefile` que l'on pourra lire directement dans un SIG.

4 La librairie sp

La multitude de formats de gestion de données spatiales a conduit au développement de la librairie `sp`. L'objectif de cette librairie est de fournir des classes et des méthodes permettant de gérer efficacement l'information spatialisée dans R. L'établissement d'une classe de référence permet alors de faciliter la conversion entre les différents formats et ainsi de simplifier l'utilisation des méthodes implémentées dans différentes librairies sur un même jeu de données. Les classes définies dans `sp` jouent et vont jouer un rôle central dans l'exploitation des données spatiales dans R. Le développement des classes `Map` et `polylist` et des fonctions associées n'est plus suivi depuis la sortie de `sp`. Le format `area`, bien plus limité, devrait disparaître à terme. La librairie `sp` permet de gérer les polygones, lignes, points et les données *raster* avec ou sans données associées. Elle utilise des classes et des méthodes de type S4. On se focalisera ici uniquement sur les classes `Polygon`, `Polygons`, `SpatialPolygons` et `SpatialPolygonsDataFrame`.

4.1 Classes

Une entité surfacique définit un objet de la classe `Polygon`. On peut créer ce type d'objet en entrant les coordonnées des sommets comme argument de la fonction `Polygon` (le premier sommet doit être rentré à la fin pour refermer) :

```
Sr1 <- Polygon(cbind(c(2, 4, 4, 1, 2), c(2, 3, 5, 4, 2)))
class(Sr1)
```

```
[1] "Polygon"
attr(,"package")
[1] "sp"
```

```
Sr2 <- Polygon(cbind(c(5, 4, 2, 5), c(2, 3, 2, 2)))
Sr3 <- Polygon(cbind(c(4, 4, 5, 10, 4), c(5, 3, 2, 5, 5)))
```

Un objet de la classe `Polygons` contient une liste de `Polygon` à laquelle est associée un identifiant :

```
Srs1 <- Polygons(list(Sr1), ID = "s1")
class(Srs1)
```

```
[1] "Polygons"
attr(,"package")
[1] "sp"
```

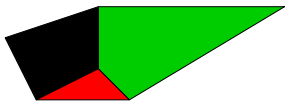
```
Srs2 <- Polygons(list(Sr2), ID = "s2")
Srs3 <- Polygons(list(Sr3), ID = "s3")
```

Un objet de la classe `SpatialPolygons` contient une liste de `Polygons` :

```
SPp <- SpatialPolygons(list(Srs1, Srs2, Srs3))
class(SPp)
```

```
[1] "SpatialPolygons"
attr(,"package")
[1] "sp"
```

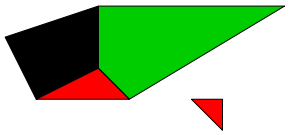
```
plot(SPp, col = 1:3)
```



La fonction `SpatialPolygons` peut également prendre comme argument un ordre pour représenter les données (utile si des entités se superposent) et une projection.

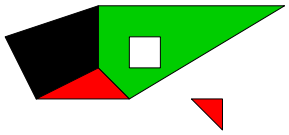
On peut gérer les îles :

```
Sr4 <- Polygon(cbind(c(7, 8, 8, 7), c(2, 2, 1, 2)))
Srs2 <- Polygons(list(Sr2, Sr4), ID = "s2+s4")
SPp <- SpatialPolygons(list(Srs1, Srs2, Srs3))
plot(SPp, col = 1:3)
```



et les lacs :

```
Sr5 <- Polygon(cbind(c(5, 6, 6, 5, 5), c(4, 4, 3, 3, 4)), hole = TRUE)
Srs3 <- Polygons(list(Sr3, Sr5), ID = "s3-s5")
SPp <- SpatialPolygons(list(Srs1, Srs2, Srs3))
plot(SPp, col = 1:3, pbg = "white")
```



Finalement, on peut associer un jeu de données (`data.frame`) à un objet `SpatialPolygons`. Pour cela, il faut que les noms des lignes du `data.frame` soient identiques aux IDs du `SpatialPolygons` :

```
df1 <- data.frame(v1 = 1:3, v2 = letters[1:3], v3 = c(1, 1, 5),
    row.names = c("s1", "s2+s4", "s3-s5"))
df1
```

```
      v1 v2 v3
s1     1  a  1
s2+s4  2  b  1
s3-s5  3  c  5
```

```
SPpdf <- SpatialPolygonsDataFrame(SPp, df1)
class(SPpdf)
```

```
[1] "SpatialPolygonsDataFrame"
attr(,"package")
[1] "sp"
```

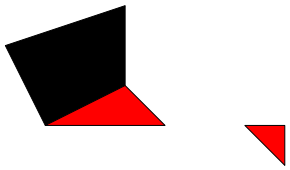
L'objet résultant est de la classe `SpatialPolygonsDataFrame`.

4.2 Méthodes

Il existe plusieurs méthodes associées aux classes de `sp`. Ce premier groupe inclut des méthodes standards :

- `[` permet de sélectionner des entités et/ou des variables du tableau de données.

```
SPp2 <- SPp[1:2]
plot(SPp2, col = 1:2, pbg = "white")
```



```
SPpdf2 <- SPpdf[1:2, 1]
as.data.frame(SPpdf)
      v1 v2 v3
s1     1  a  1
s2+s4  2  b  1
s3-s5  3  c  5
as.data.frame(SPpdf2)
      v1
s1     1
s2+s4  2
```

- `[[` permet de sélectionner une colonne du tableau de données.

```
SPpdf[[2]]
```

```
[1] a b c
Levels: a b c
```

- `[[<-` permet d'assigner une valeur à une colonne du tableau de données.

```
SPpdf[[2]]
```

```
[1] a b c
Levels: a b c
```

```
SPpdf[[2]] <- factor(letters[7:9])
SPpdf[[2]]
```

```
[1] g h i
Levels: g h i
```

- `print`, `summary`, `plot`, `dim`, `names`, `as.data.frame`, ...

D'autres méthodes sont plutôt orientées sur la partie spatiale :

- `dimensions` renvoie le nombre de dimensions spatiales.

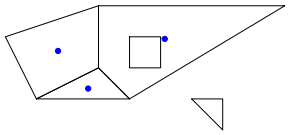
```
dimensions(SPpdf)
```
```
[1] 2
```

- `bbox` renvoie les coordonnées du rectangle contenant l'ensemble des données (*bounding box*).

```
bbox(SPpdf)
```
```
   min max
r1   1  10
r2   1   5
```

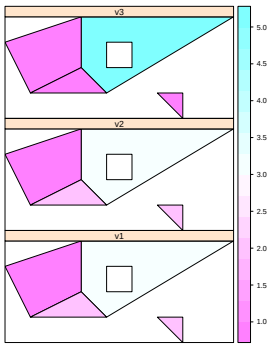
- `coordinates` renvoie les coordonnées spatiales. Pour un objet surfacique, la fonction renvoie les coordonnées des centres des polygones.

```
coordinates(SPpdf)
```
```
          [,1]     [,2]
[1,] 2.696970 3.545455
[2,] 3.666667 2.333333
[3,] 6.133333 3.933333
```
```
plot(SPpdf)
points(coordinates(SPpdf), col = "blue", pch = 20, cex = 2)
```



- `spplot` permet de faire une représentation graphique des variables associées. Nous y reviendrons plus tard.

```
spplot(SPpdf)
```



- `gridded`, `transform`, `overlay`, `spsample`

4.3 Quelques fonctions utiles

On présente ici quelques fonctions utiles pour manipuler les objets de type `SpatialPolygons` ou `SpatialPolygonsDataFrame` :

- `spCbind` rajoute une ou plusieurs variables au tableau de données associé d'un `SpatialPolygonsDataFrame` :

```
v4 <- 9:11
SPpdf3 <- spCbind(SPpdf, v4)
```

```
 as(SPpdf3, "data.frame")
      v1 v2 v3 v4
s1     1  g  1  9
s2+s4  2  h  1 10
s3-s5  3  i  5 11
```

- `spRbind` rajoute des enregistrements à un `SpatialPolygonsDataFrame` ou un `SpatialPolygons` :

```
 SPpdf4 <- SPpdf[1, ]
 SPpdf5 <- SPpdf[2, ]
 SPpdf6 <- spRbind(SPpdf4, SPpdf5)
```

- `spChFIDs` change les IDs d'un `SpatialPolygonsDataFrame` ou d'un `SpatialPolygons` :

```
 SPpdf7 <- spChFIDs(SPpdf6, c("lab1", "lab2"))
 as(SPpdf7, "data.frame")
     v1 v2 v3
lab1  1  g  1
lab2  2  h  1
```

- `polygons` permet d'assigner ou de récupérer la partie spatiale d'un objet :

```
 class(polygons(SPpdf))
[1] "SpatialPolygons"
attr(,"package")
[1] "sp"
```

4.4 Conversions, importation et exportation

De nombreuses fonctions permettent de convertir différentes classes vers celles définies dans `sp` (et inversement). On citera notamment les fonctions de `adehabitat` :

- `area2spol` convertit un objet `area` en `SpatialPolygons`.
- `spol2area` convertit un objet `SpatialPolygons` ou `SpatialPolygonsDataFrame` en `area`.
- `attpol2area` récupère les données associées d'un `SpatialPolygonsDataFrame` et renvoie un `data.frame`

Voir également dans `maptools` les fonctions de conversions avec les classes de `spatstat`, `maps` ou `PBSmapping` notamment.

Les fonctions `readShapePoly` et `writePolyShape` permettent d'importer et d'exporter des fichiers *shapefile* vers des objets de type `SpatialPolygonsDataFrame` :

```
fr.SPpdf <- readShapePoly("dep_france_dom")
```

5 Cartographie

La représentation cartographique de données ou de résultats est une étape primordiale lorsqu'on analyse des données spatialisées. On illustre avec les résultats du premier tour de l'élection présidentielle de 1988. Les données sont disponibles dans la pile de données `elec88` :

```
 data(elec88)
 names(elec88)
[1] "tab"     "res"     "lab"     "area"    "contour" "xy"      "neig"
```

Cette liste contient notamment un découpage en départements de type `area` (`area`), les coordonnées des centres des départements (`xy`) et un tableau de données (`tab`).

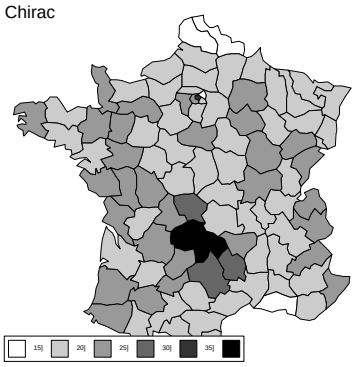
5.1 Représentations surfaciques

On représente le fond de carte :

```
area.plot(elec88$area)
```

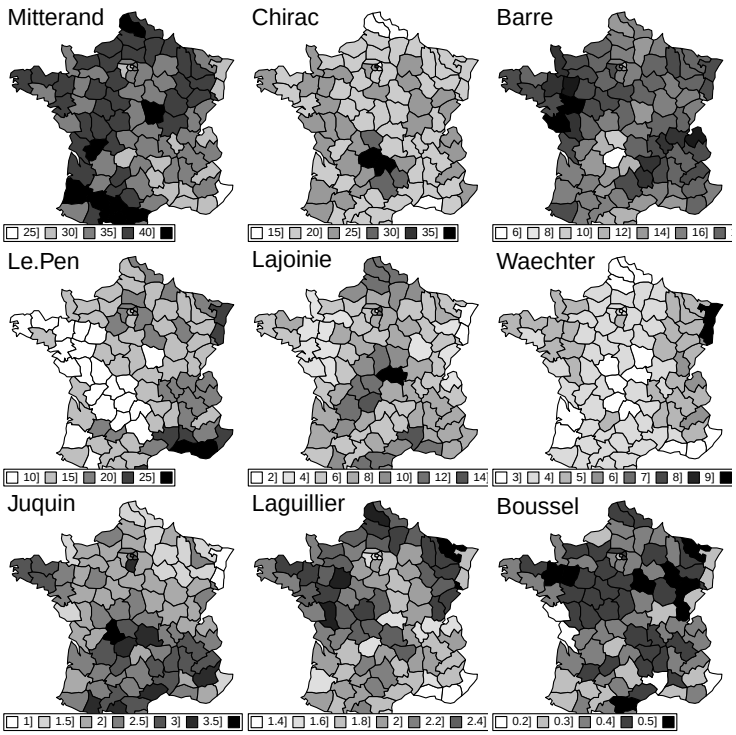
Pour représenter le score du candidat Chirac :

```
area.plot(elec88$area, values = elec88$tab$Chirac, sub = "Chirac",
    csub = 2)
```



Pour représenter les scores de tous les candidats :

```
par(mfrow = c(3, 3))
for (i in 1:9) {
    x <- elec88$tab[, i]
    area.plot(elec88$area, val = x, sub = names(elec88$tab)[i],
        csub = 3, cleg = 1.5)
}
```



Pour illustrer l'utilisation de `sp`, on utilise le fond de carte IGN. L'information de `elec88` concerne 94 départements, il y a en 100 dans les données IGN. On enlève les départements supplémentaires (DOM-TOM, Corse) du fond de carte IGN :

```
dep94 <- substr(rownames(elec88$tab), 2, 10)
dep94[1:9] <- paste("0", dep94[1:9], sep = "")
fr.SPpdf94 <- fr.SPpdf[as.character(fr.SPpdf$CODE_DEPT) %in% dep94,
    ]
plot(fr.SPpdf94)
```

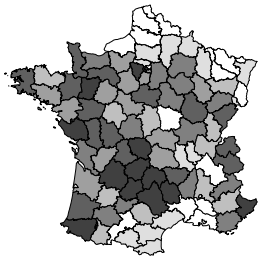
On peut alors créer un nouvel objet `SpatialPolygonsDataFrame` dans lequel on associe le tableau concernant les élections :

```
fr.SPpdf94 <- spChFIDs(fr.SPpdf94, row.names(elec88$tab))
fr.SPpdf94.el <- SpatialPolygonsDataFrame(polygons(fr.SPpdf94),
    elec88$tab)
names(fr.SPpdf94.el)
```

```
[1] "Mitterand"  "Chirac"     "Barre"      "Le.Pen"     "Lajoinie"   "Waechter"
[7] "Juquin"     "Laguillier" "Boussel"
```

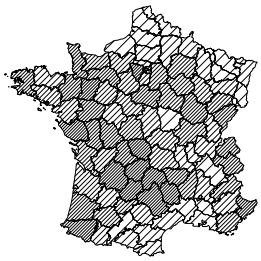
La carte pour le candidat Chirac en niveaux de gris :

```
brks <- quantile(fr.SPpdf94.el$Chirac, seq(0, 1, 1/7))
cols <- grey((length(brks):2)/length(brks))
plot(fr.SPpdf94.el, col = cols[findInterval(fr.SPpdf94.el$Chirac,
    brks, all.inside = TRUE)])
```



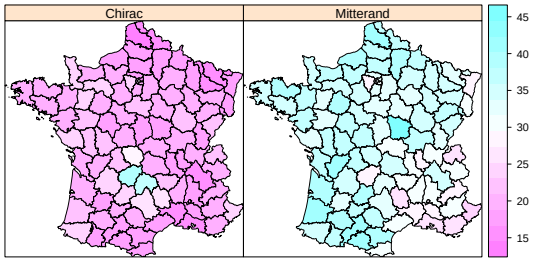
La même carte avec des densités de lignes :

```
dens <- (2:length(brks)) * 3
plot(fr.SPpdf94.el, density = dens[findInterval(fr.SPpdf94.el$Chirac,
    brks, all.inside = TRUE)])
```



On peut également utiliser la fonction `spplot` qui est plus évoluée. La carte pour les candidats Mitterand et Chirac :

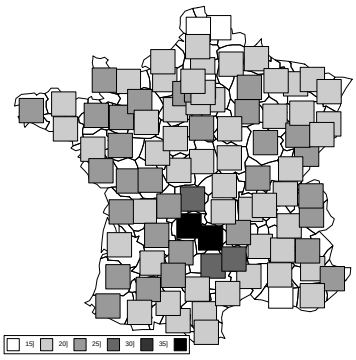
```
spplot(fr.SPpdf94.el, c("Chirac", "Mitterand"))
```



5.2 Représentations ponctuelles

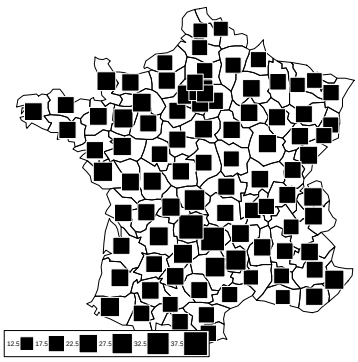
Une alternative à la coloration de surfaces consiste à représenter des données ponctuelles. On associe un point à chaque polygone (son centre par exemple) et on cartographie alors sur les points. La fonction `s.value` de librairie `ade4` permet ce type de représentation.
Par niveaux de gris :

```
area.plot(elec88$area)
s.value(elec88$xy, elec88$tab$Chirac, method = "greylevel", add.plot = TRUE)
```



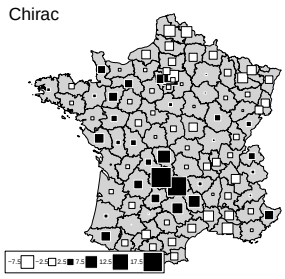
Par taille variable des symboles :

```
area.plot(elec88$area)
s.value(elec88$xy, elec88$tab$Chirac, add.plot = TRUE)
```



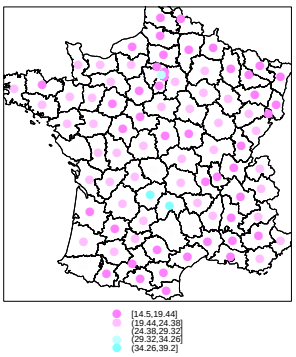
Si on centre les données, sur le fond de carte IGN :

```
plot(fr.SPpdf94.el, col = "lightgrey")
s.value(coordinates(fr.SPpdf94.el), scale(elec88$tab$Chirac, scale = FALSE),
    add.plot = TRUE, sub = "Chirac", csub = 2)
```



La librairie `sp` offre le même type de graphique :

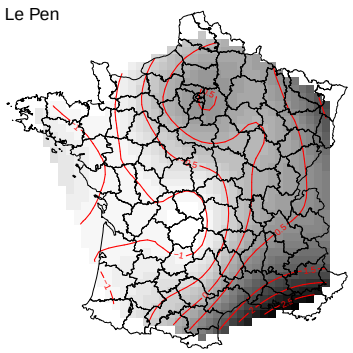
```
fr.Spoints <- SpatialPointsDataFrame(coordinates(fr.SPpdf94.el),
    as.data.frame(fr.SPpdf94.el))
spplot(fr.Spoints, "Chirac", sp.layout = list("sp.polygons", fr.SPpdf94.el),
    cex = 2)
```



5.3 Courbes de niveaux

Finalement, on peut évoquer les courbes de niveaux. C'est un jeu algorithmique sur un ensemble de valeurs placées sur un réseau. Les points sont placés entre une mesure supérieure au seuil et une mesure inférieure au seuil par interpolation linéaire. Quelles que soient les valeurs, on a une solution unique. Les mesures ne sont pas en général sur un réseau. On estime alors à partir des données en (x,y) des valeurs sur un réseau qui recouvre la zone d'étude. Il existe des méthodes simples (régression polynomiale) ou sophistiquées (krigeage). Une des plus efficace est la régression locale étendue de une à deux dimensions (*loess*) utilisée dans la fonction `s.image` :

```
s.image(coordinates(fr.SPpdf94.el), elec88$tab$Le.Pen, kgrid = 5,
    sub = "Le Pen", csub = 2)
plot(fr.SPpdf94.el, add = T)
```



Références

J. Thioulouse, D. Chessel, S. Dolédec, and J.M. Olivier. ADE-4 : a multivariate analysis and graphical display software. *Statistics and Computing*, 7 :75–83, 1997.