

# Utiliser des fichiers spatiaux dans des applications R

---

**BAUDOIN, Raymond**

*Muséum national d'Histoire naturelle  
Inventaire et suivi de la biodiversité*

MUSÉUM  
NATIONAL  
D'HISTOIRE  
NATURELLE



## Utiliser des fichiers spatiaux dans des applications R

Raymond Baudoin

Inventaire et suivi de la biodiversité - CBNBP

Département EGB

*baudoin@mnhn.fr*

Pouvoir manipuler des fichiers spatiaux sous R se révèle utile pour la présentation cartographique de résultats mais aussi pour effectuer des traitements statistiques sur les objets géographiques que n'offrent pas les SIG. La présentation de Clément CALENGE sur l'analyse de l'utilisation de l'espace par la faune sauvage avec l'emploi de la librairie `adehabitat` du 14 mai 2009 dans le cadre des séminR en est une bonne illustration.

Mais avant cela il faut pouvoir accéder aux données géographiques gérées par ailleurs par des SIG libres ou commerciaux c'est-à-dire pouvoir :

- lire ces fichiers vectoriels, rasters, images qu'ils soient de format texte ou binaire ;
- travailler sur les cartes pour les adapter au sujet d'étude. Faire des regroupements de cartes, modifier les projections, les origines, etc ;
- créer de nouveaux objets géographiques.

Actuellement le site du CRAN propose plus de 90 librairies qui offrent des fonctions traitant des objets géographiques. Dans cette présentation sont utilisées les librairies `adehabitat`, `SDMTools`, `maptools` mais surtout `rgdal` et `raster` qui s'appuient sur l'incontournable librairie `sp`.

Consulter la CRAN Task View: Analysis of Spatial Data  
<http://cran.r-project.org/web/views/Spatial.html>

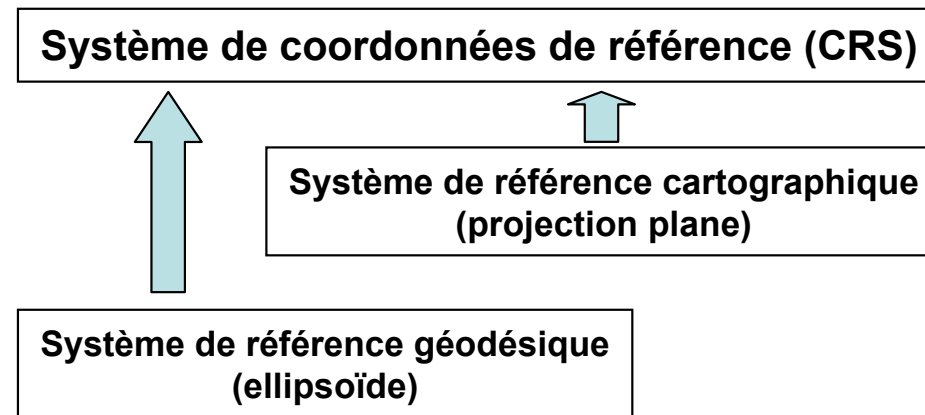
Ainsi que la mailing-list formée autour de l'utilisation de données géographiques et cartographiques  
[r-sig-geo@r-project.org](mailto:r-sig-geo@r-project.org) [website](#)

maintenues par Roger Bivand

# Systeme d'information géographique

## S.I.G.

Un système d'information géographique (SIG) est un système d'information capable de stocker, d'organiser et de présenter des données alphanumériques spatialement référencées par des coordonnées dans un système de référence (CRS).



L'usage du S.I.G. est la représentation de l'environnement spatial en se basant sur :

- des objets (primitives) géométriques : des points, des vecteurs (arcs), des polygones, etc.
- *des maillages (raster).*

Aux objets sont associées des informations attributaires telles que leur nature (route, voie ferrée, forêt, etc.) ou des informations contextuelles (nombre d'habitants, superficie, etc.).

# La base de données EPSG des paramètres géodésiques

L'ensemble de données EPSG est un référentiel structuré pour :

- identifier les coordonnées de telle sorte que ces coordonnées décrivent la position sans ambiguïté par la définition du système de coordonnées de référence (CRS).
- définir les transformations et conversions qui permettent de modifier les coordonnées d'un CRS à un autre.

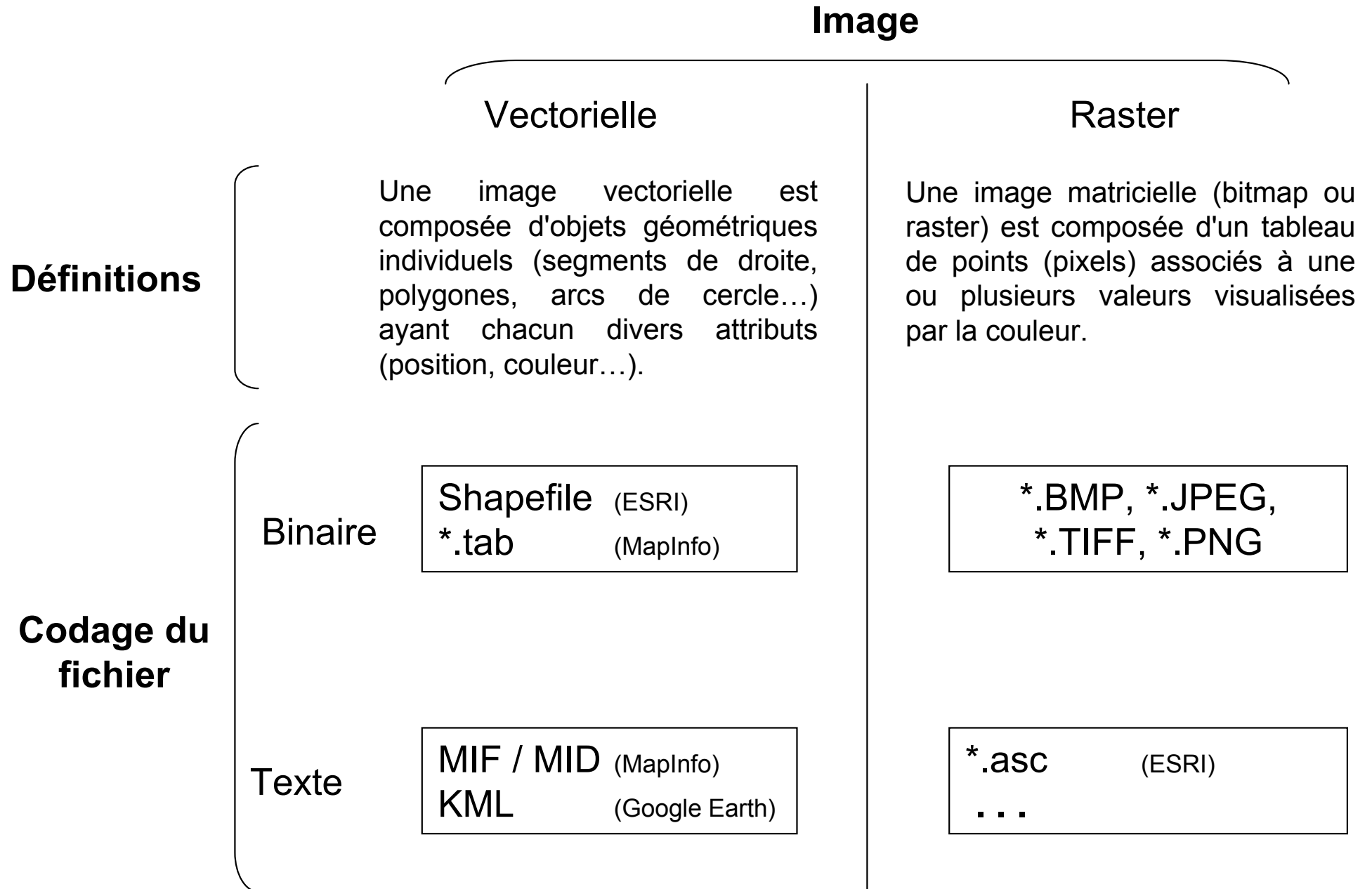
L'EPSG Geodetic Parameter Dataset est maintenu actuellement par le sous-comité de géodésie de l'OGP.

<http://www.epsg.org/>

## Signification de quelques sigles utilisés

CRS	: Coordinate Reference System
EPSG	: European Petroleum Survey Group
GDAL	: Geospatial Data Abstraction Library
OGC	: Open Geospatial Consortium
OGP	: International Association of Oil & Gas producers
SDE	: Spatial Database Engine (moteur de recherche dans des bases de données spatiales)
SIR	: Système International de Référence
SRS	: Spatial Referencing System

# Les principaux formats utilisés par les S.I.G.



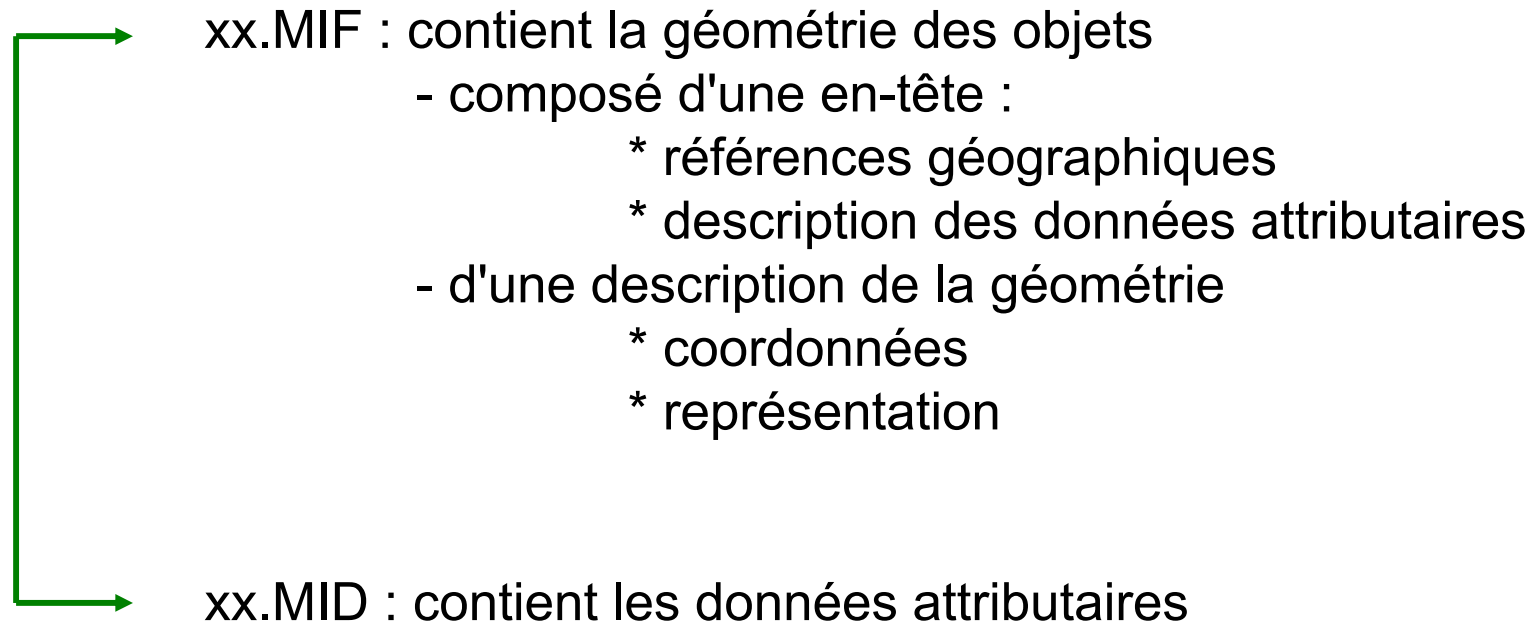
## Exemple de format texte

### ► Pour des données vectorielles

Une image vectorielle est composée d'objets géométriques individuels (segments de droite, polygones, arcs de cercle, etc.) ayant chacun divers attributs (position, couleur, etc.).

Fichiers MIF - MID : format d'échange du SIG MapInfo  
(MapInfo Interchange Format)

Deux fichiers associés



# Structure des fichiers MIF - MID

**.MIF**

En-tête

Version 300  
Charset "WindowsLatin1"  
Delimiter ","  
Index 3,12,14  
CoordSys Earth Projection 3, 1002, "m",  
0, 46.8, 45.898918964419,  
47.696014502038, 600000, 2200000  
Bounds (-113967455.417, -  
106367759.649) (115167455.417,  
122767151.185)  
Columns 14  
ID\_COM Integer  
NOM\_COM Char(50)  
NUM\_COM Char(5)  
STATUT Char(2)  
X\_COM Integer  
Y\_COM Integer  
SURFACE Integer  
POPU Integer  
NUM\_CAN Char(2)  
NUM\_ARR Char(1)  
NOM\_DEP Char(30)  
NUM\_DEP Char(3)  
NOM\_REG Char(30)  
NUM\_REG Char(2)  
Data

Description du MID

Description du MID

Objets

Region 1  
131  
479087 2463986  
479112 2463991  
479276.9 2464086  
479452 2464171  
479562 2464247  
...  
...  
478947 2463896  
479077 2463971  
479087 2463986  
Pen (1,2,0)  
Brush (2,16777215,16777215)  
Center 480103.6 2464572.6  
Region 1  
221  
508173 2418283  
508188 2418343  
508213 2418363  
508268 2418378.1  
508318 2418399  
508343.1 2418399

Description  
d'un  
polygone

**.MID**

270000001,"ACLOU","27001","06",480418,2464086,370,200,"09","2","EURE","27","HAUTE-NORMANDIE","23"  
270000002,"ACON","27002","06",509314,2419859,916,400,"25","3","EURE","27","HAUTE-NORMANDIE","23"  
270000003,"ACQUIGNY","27003","06",515518,2463665,1783,1400,"39","3","EURE","27","HAUTE-NORMANDIE","23"  
270000004,"AIGLEVILLE","27004","06",533415,2445735,324,200,"26","3","EURE","27","HAUTE-NORMANDIE","23"  
270000005,"AILLY","27005","06",520222,2462811,1555,800,"42","1","EURE","27","HAUTE-NORMANDIE","23"



# fichier MIF : gestion des polygones disjoints

*exemple de la commune du Val-de-Reuil (insee : 27701)*

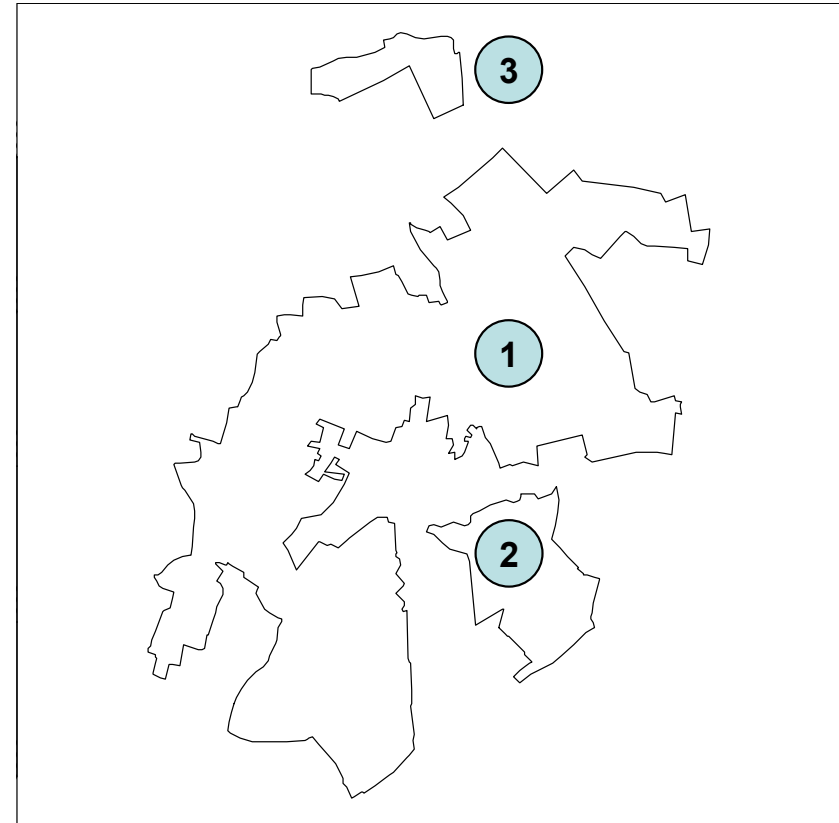
nombre de polygones

```
[22] "Region 3"
[23] " 390" ← nombre de sommets
[24] "515719 2470940"
[25] "515754 2471010"
[26] "515819 2471131"
[27] "515894 2471236"
. . .
. . .
[409] "515684 2470555.1"
[410] "515639.1 2470595"
[411] "515659 2470705"
[412] "515699 2470820"
[413] "515719 2470940"
[414] " 73" ← nombre de sommets
[415] "519146.4 2471924"
[416] "519209.1 2472150.7"
[417] "519184 2472157.7"
. . .
. . .
[486] "519216 2471842"
[487] "519146.4 2471924"
[488] " 41" ← nombre de sommets
[489] "518682 2478654.1"
[490] "518303.6 2478485.3"
. . .
. . .
[527] "518636 2479335"
[528] "518667 2479000"
[529] "518682 2478654.1"
Pen (1,2,0)
Brush (2,16777215,16777215)
Center 515828.8 2473912
```

1

2

3



# Exemple de format texte

## ► Pour des données raster

Une image matricielle (bitmap ou raster) est composée d'un tableau de points (pixels) associés à une ou plusieurs valeurs visualisées par la couleur.

Fichier ASCII format ArcGis text raster (.asc)

En-tête

```
ncols      2442
nrows      3719
xllcorner  575350,517
yllcorner  2444630,98
cellsize   30
NODATA_value -9999
```

L'ouvrir avec un éditeur de texte pour :

- Vérifier que le séparateur décimal soit un point

- Vérifier la cohérence de cette valeur avec le codage des pixels

Valeurs pour chaque pixel

```
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -
9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
...
...
91 92 93 93 91 90 89 89 89 89 89 88 88 88 87 86 86 86 86 86 86 85 81 79 77 78 78 77 76 75
74 76 76 73 70 70 69 70 70 72 73 74 75 76 75 76 75 73 73 68 64 60 60 60 60 65 69 75 77 71 62
59 57 56 57 58 62 66 75 78 78 77 77 77 75 74 73 73 72 73 74 75 75 77 79 79 79 78 78 77 76 74
73 70 69 68 68 68 67 66 65 63 62 60 60 59 59 -9999 -9999 -9999 -9999
-9999 -9999 82 80 79 80 83 87 92 92 91 91 89 89 88 88 90 88 86 87 88 88 87 84 84 80 78 78 78
77 77 76 75 74 73 72 72 71 72 71 71 72 74 77 78 79 80 82 85 86 90 92 96 98 98 96 91 88
...
```

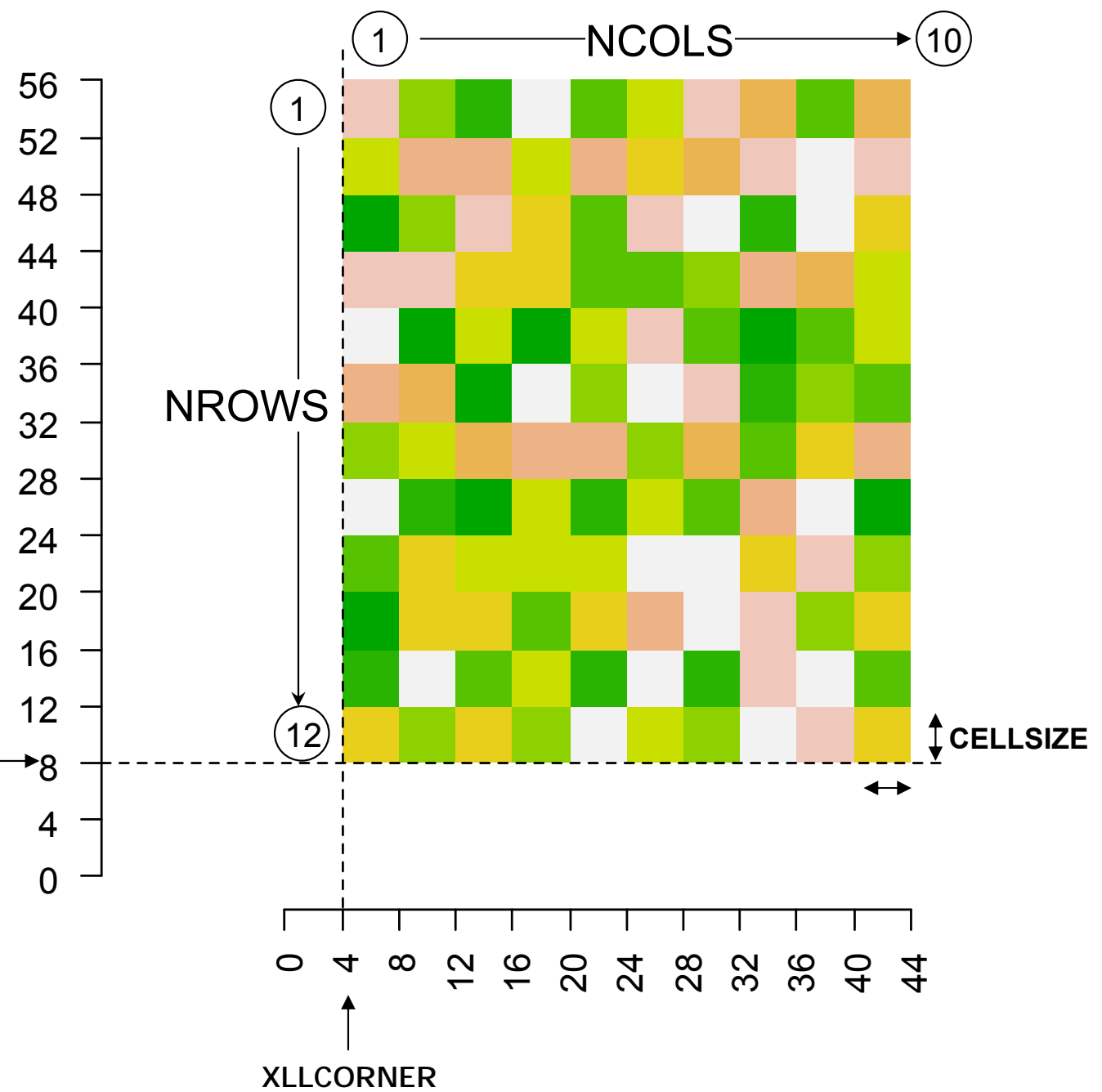
# Fichier raster (.asc) et sa représentation

```

NCOLS 10
NROWS 12
XLLCORNER 4
YLLCORNER 8
CELLSIZE 4
NODATA_value -3.4e+38
----->
2 7 9 1 8 6 2 4 8 4
6 3 3 6 3 5 4 2 1 2
10 7 2 5 8 2 1 9 1 5
2 2 5 5 8 8 7 3 4 6
1 10 6 10 6 2 8 10 8 6
3 4 10 1 7 1 2 9 7 8
7 6 4 3 3 7 4 8 5 3
1 9 10 6 9 6 8 3 1 10
8 5 6 6 6 1 1 5 2 7
10 5 5 8 5 3 1 2 7 5
9 1 8 6 9 1 9 2 1 8
5 7 5 7 1 6 7 1 2 5
    
```

YLLCORNER →

exemple : "rst1.asc"



LIRE DES  
FICHIERS  
CARTOGR  
APHIQUES

# Importer des données géographiques dans



## Fonctions suivant les formats

<b>Données vectorielles</b>		<b>fichier</b>	<b>fonctions</b>	<b>librairies</b>
<b>codage</b>	Binaire	Shapefile *.tab	readShapeSpatial () <b>readOGR ()</b>	maptools <b>rgdal</b>
	Texte	MIF / MID GML, KML	<b>readOGR ()</b>	<b>rgdal</b>

<b>Données raster</b>		<b>fichier</b>	<b>fonctions</b>	<b>librairies</b>
<b>codage</b>	Binaire	Images (bmp, jpeg, png)	readGDAL () <b>raster ()</b>	rgdal <b>raster</b>
	Texte	*.asc	<b>raster ()</b> import.asc () read.asc () readGDAL ()	<b>raster</b> ade4habitat SDMTools rgdal

# library (rgdal)

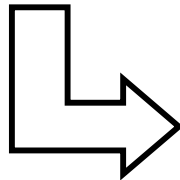
Utilise le package sp et les runtimes GDAL et PROJ.4

## Lecture de données vectorielles

Aux formats : ESRI Shapefile, ESRI ArcSDE, MapInfo (tab et mif/mid), GML, KML, DXF, ...

```
# Afficher les formats supportés (pilotes)  
ogrDrivers ()
```

**readOGR** (dsn, layer, verbose = TRUE, p4s=NULL, stringsAsFactors=default.stringsAsFactors(), drop\_unsupported\_fields=FALSE, input\_field\_name\_encoding=NULL, ...)



dsn : nom d'un répertoire ou nom de fichier suivant le pilote

layer : nom de la couche à lire

p4s : système de référence cartographique (CRS)

drop\_unsupported\_fields : si TRUE saute les champs autre que String, Integer et Real; Date, heure et DateTime sont converties en String.

input\_field\_name\_encoding : codepage pour le nom de dsn et de layer.

**Retourne un objet spatial de classe SpatialPointsDataFrame, SpatialLinesDataFrame ou SpatialPolygonsDataFrame.**

### ➤ Premier exemple : lecture d'un fichier .tab de MapInfo

```
d78 <- readOGR (Dept78.TAB, layer="Dept78", stringsAsFactors=FALSE)  
plot (d78, col="snow2")
```



# Objet de classe SpatialPolygonsDataFrame [package "sp"]

Données vectorielles

str (d78)

Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots

..@ data :'data.frame': 1 obs. of 6 variables:

```
.. ..$ Code           : chr "78"
.. ..$ Nom            : chr "YVELINES"
.. ..$ INSEE_Région   : chr "11"
.. ..$ Id_BDCarto     : int 78
.. ..$ Abscisse_Département : int 583975
.. ..$ Ordonnée_Département : int 2423234
```

} données  
attributaires

..@ polygons :List of 1

```
.. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
.. .. . . .@ Polygons :List of 1
.. .. . . . . . $ :Formal class 'Polygon' [package "sp"] with 5 slots
.. .. . . . . . . . @ labpt : num [1:2] 563622 2424181
.. .. . . . . . . . @ area : num 2.31e+09
.. .. . . . . . . . @ hole : logi FALSE
.. .. . . . . . . . @ ringDir: int 1
.. .. . . . . . . . @ coords : num [1:3355, 1:2] 570261 570196 570161 570162 570277 ...
.. .. . . .@ plotOrder: int 1
.. .. . . .@ labpt : num [1:2] 563622 2424181
.. .. . . .@ ID : chr "1"
.. .. . . .@ area : num 2.31e+09
```

..@ plotOrder : int 1

..@ bbox : num [1:2, 1:2] 534891 2382272 592091 2454474

.. ..- attr(\*, "dimnames")=List of 2

.. .. . \$ : chr [1:2] "x" "y"

.. .. . \$ : chr [1:2] "min" "max"

..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots

```
.. .. .@ projargs: chr " +proj=lcc +lat_1=45.898918964419 +lat_2=47.696014502
+lat_0=46.8 +lon_0=0| __truncated__
```

Fonctions associées

ogrInfo (dsn="Dept78.TAB", layer="Dept78")

correspond à un str() des  
données attributaires

ogrFIDs("Dept78.TAB", "Dept78")

Liste les ID  
des polygones

bbox (d78)

	min	max
x	534891	592091
y	2382272	2454474

Emprise de la carte

OGRSpatialRef ("Dept78.TAB", "Dept78")

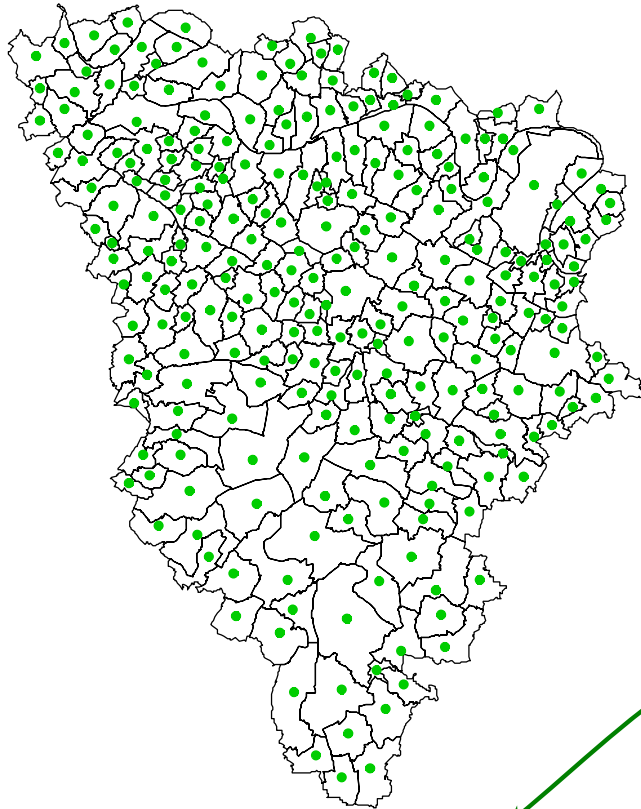
Affiche le CRS

library (rgdal) - Données vectorielles

# Lecture de fichiers MIF / MID (MapInfo)

```
file <- "Com78.MIF"  
c78 <- readOGR (file, layer="Com78")  
plot (c78)
```

Mettre le nom du fichier  
sans l'extention



**PAS DE MELANGE DE GEOMETRIES DANS UNE COUCHE**  
*que des polygones, points ...*

```
c78p <- readOGR (dsn="Com78p.MIF", layer="Com78p")  
Erreur dans ogrInfo(dsn = dsn, layer = layer,  
input_field_name_encoding = input_field_name_encoding) :  
Multiple incompatible geometries: 1:3
```

```
ogrInfo (dsn = file, layer="Com78")
```

```
Source: "C:\Mes documents  
sauvegardés\Enseignement\Séminaire  
R\Séminaire_201103\Cartes_78_L2E\Com78.MIF  
", layer: "Com78"
```

```
Driver: MapInfo File number of rows 262
```

```
Feature type: wkbPolygon with 2 dimensions
```

```
+proj=lcc +lat_1=45.89891889 +lat_2=47.69601444  
+lat_0=46.8 +lon_0=0 +x_0=600000  
+y_0=2200000 +a=6378249.2  
+b=6356515.000000472 +towgs84=-168,-  
60,320,-0,-0,-0,0 +pm=2.337229166667  
+units=m +no_defs
```

```
Number of fields: 9
```

	name	type	length	typeName
1	ID	2	10	Real
2	NOM	4	50	String
3	INSEE	4	5	String
4	X	2	8	Real
5	Y	2	8	Real
6	SUPERFICIE	2	10	Real
7	POPULATION	2	8	Real
8	NOM_DEPT	4	30	String
9	NOM_REGION	4	30	String

```
points (c78@data$X, c78@data$Y,pch=19,col=3)
```



# Lecture de fichiers Shapefile (ESRI)

Réparties en plusieurs fichiers : \*.shp, \*.shx, \*.dbf, \*.prj

## ➤ library (rgdal)

```
c78sh <- readOGR (dsn="ESRI", layer="Com78_region")
```

Nom du répertoire  
contenant les fichiers

nom du fichier sans  
l'extention

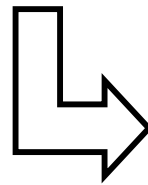
OGR data source with **driver: ESRI Shapefile**

Source: "ESRI", layer: "Com78\_region"

with 262 features and 9 fields

```
Source: "ESRI", layer: "Com78_region"
Driver: ESRI Shapefile number of rows 262
Feature type: wkbPolygon with 2 dimensions
+proj=lcc +lat_1=45.89891889
+lat_2=47.69601444 +lat_0=46.8
+lon_0=2.337229104484 +x_0=600000
+y_0=2200000 +ellps=clrk80 +units=m +no_defs
Number of fields: 9
```

```
ogrInfo (dsn="ESRI", layer="Com78_region")
```



	name	type	length	typeName
1	ID	0	10	Integer
2	NOM	4	50	String
3	INSEE	4	5	String
4	X	0	8	Integer
5	Y	0	8	Integer
6	SUPERFICIE	0	10	Integer
7	POPULATION	0	8	Integer
8	NOM_DEPT	4	30	String
9	NOM_REGION	4	30	String

## ➤ library (maptools)

```
c78sh <- readShapeSpatial (Com78_region.shp)
```

Retourne aussi un objet  
SpatialPolygonsDataFrame

# Lecture de fichiers KML

(Google Earth)

library (rgdal) - Données vectorielles

Avec Google Earth utiliser le système de référence international : WGS 84

```
d78k <- readOGR ("d78WGS84.KML", layer="Contour78")
```

```
plot (d78k )
```

Nom du fichier

Champ name du fichier KML

## Début du fichier d78WGS84.KML

```
<?xml version="1.0" encoding="UTF-8"?>  
<kml xmlns="http://earth.google.com/kml/2.2">  
<Document>  
<name>Contour78</name>  
<description><![CDATA[]]></description>  
<Style id="1">
```

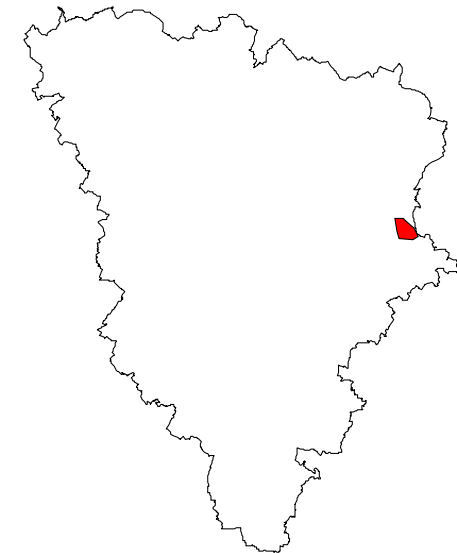
# Fichier KML créé avec Google Earth

```
Vk <- readOGR ("Versailles - Polygone.kml", layer="Versailles - Polygone.kml")
```

```
plot (Vk, add=TRUE, col=2 )
```

# Création et écriture d'un KML par la fonction kmlPolygone de Maptools

```
kmlPolygone (obj=d78WGS84, kmlfile="d78WGS84.KML", name="Contour",  
description="", col=NULL, visibility=1, lwd=1, border=1,  
kmlname="Contour78", kmldescription="")
```



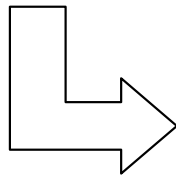
# library (rgdal) - Données raster

```
Liste des pilotes existants  
Drv <- gdalDrivers()  
Drv [which(Drv[,3] | Drv[,4]), c(1,2)]
```

```
Pilotes fichiers  
images : JPEG,  
PNG, BMP,  
GTiff
```

```
Pilotes fichiers rasters :  
AAIGrid Arc/Info ASCII Grid  
EHdr ESRI .hdr Labelled  
ENVI ENVI .hdr Labelled  
GSAG Golden Software ASCII Grid (.gri)  
GTiff GeoTIFF  
XPM X11 PixMap Format  
...
```

```
readGDAL (fname, offset, region.dim, output.dim, band, p4s=NULL, ...,  
half.cell=c(0.5, 0.5), silent = FALSE)
```



fname : Nom de la carte avec l'extension identifiant le pilote  
offset : Nombre de lignes et de colonnes de l'origine (commence au coin supérieur gauche) \*  
region.dim : Le nombre de lignes et de colonnes à lire dans l'ensemble de données \*  
output.dim : Le nombre de lignes et de colonnes à retourner dans l'objet créé \*  
p4s : Système de référence cartographique (CRS)  
half.cell : Utilisé pour ajuster l'intra-cellule de compensation à partir du coin au centre, le plus souvent par défaut, mais peut être réglé pour c = (0,0) si nécessaire \*  
silent : Si TRUE, le commentaire et les erreurs non fatales du pilote supprimés

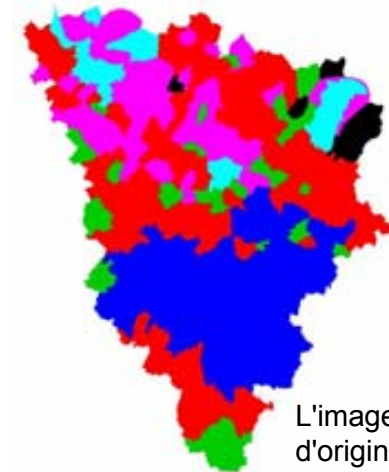
\* peut changer

## Lecture d'un fichier image

```
Veg <- readGDAL ("Veg78.jpg")
```

```
image (Veg) # pour l'affichage
```

```
Retourne un objet spatial SpatialGridDataFrame
```



L'image  
d'origine

# Objet de classe SpatialGridDataFrame [package "sp"]

## Pour les données raster

str (Veg)

**Formal class 'SpatialGridDataFrame' [package "sp"] with 6 slots**

..@ data : 'data.frame': 450912 obs. of 3 variables:

.. ..\$ band1: int [1:450912] 255 255 255 255 255 255 255 255 255 255 255 ...

.. ..\$ band2: int [1:450912] 255 255 255 255 255 255 255 255 255 255 255 ...

.. ..\$ band3: int [1:450912] 255 255 255 255 255 255 255 255 255 255 255 ...

**Si image : gestion  
des couleurs  
sinon, valeurs  
lues.**

bmp : 1 band  
jpeg, png : 3 bands

..@ grid : Formal class 'GridTopology' [package "sp"] with 3 slots

.. ..@ cellcentre.offset: Named num [1:2] 0.5 0.5

.. ..- attr(\*, "names")= chr [1:2] "x" "y"

.. ..@ cellsize : num [1:2] 1 1 ← **Taille du pixel**

.. ..@ cells.dim : int [1:2] 672 671 ← **ncol, nrow soit y,x**

..@ grid.index : int(0)

..@ coords : num [1:2, 1:2] 0.5 671.5 0.5 670.5

.. ..- attr(\*, "dimnames")=List of 2

.. ..\$ : NULL

.. ..\$ : chr [1:2] "x" "y"

..@ bbox : num [1:2, 1:2] 0 0 672 671 ← **xmin, ymin, xmax, ymax**

.. ..- attr(\*, "dimnames")=List of 2

.. ..\$ : chr [1:2] "x" "y"

.. ..\$ : chr [1:2] "min" "max"

..@ proj4string: Formal class 'CRS' [package "sp"] with 1 slots

.. ..@ projargs: chr NA ← **pas de CRS par défaut pour une image  
(absent du fichier)**

Ajout possible  
d'autres données  
dans @data

# library (raster)

## Une fonction unique pour créer ou lire un raster raster (x, crs=NA, ...)

### ➤ Création d'un raster

```
nrows <- 12; ncols <- 10; cellsize <- 4; xllcorner <- 4; yllcorner <- 8
```

```
rst <- raster (nrows = nrows, ncols = ncols,  
              xmn = xllcorner, xmx = xllcorner + (ncols*cellsize),  
              ymn = yllcorner, ymx = yllcorner + (nrows*cellsize))
```

```
rst [ ] <- sample(1:10, size=nrows*ncols, replace = TRUE)
```

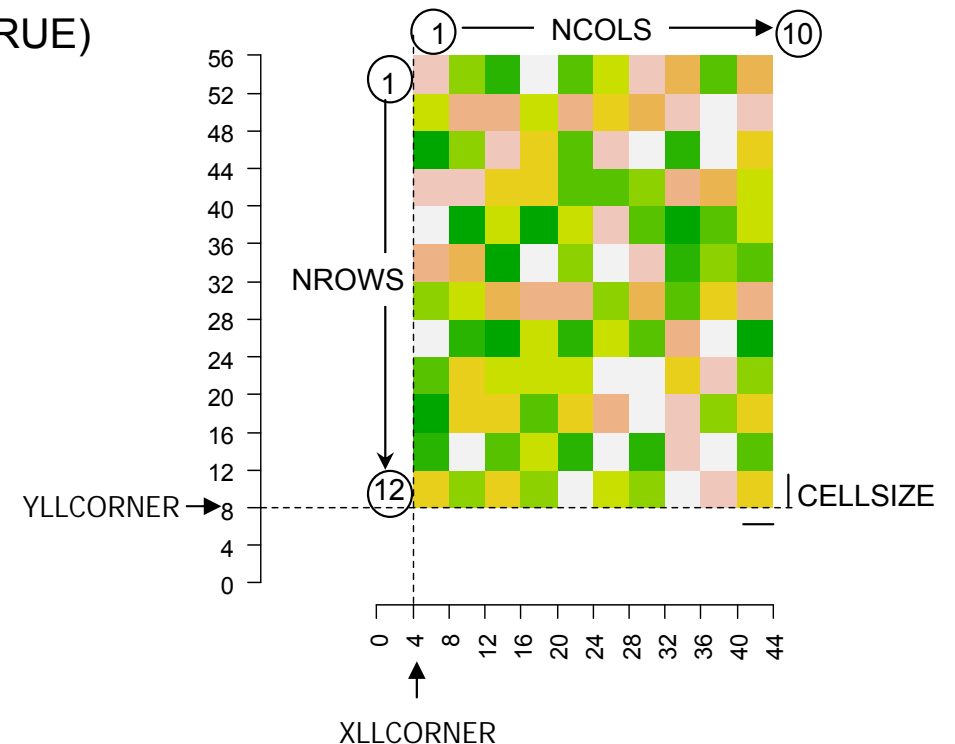
Affecte les valeurs

### ➤ Ecriture d'un raster au format asc

```
writeRaster ( rst, filename="rst1.asc")
```

### ➤ Lecture d'un fichier raster

```
rst1 <- raster("rst1.asc")  
plot (rst1, asp=1)
```



# Objet de classe raster

```
"rst1.asc"
NCOLS 10
NROWS 12
XLLCORNER 4
YLLCORNER 8
CELLSIZE 4
```

## rst

```
class      : RasterLayer
dimensions : 12, 10, 1 (nrow, ncol, nlayers)
resolution : 4, 4 (x, y)
extent     : 4, 44, 8, 56 (xmin, xmax, ymin, ymax)
projection : +proj=longlat +datum=WGS84
values     : in memory
min value  : 1
max value  : 10
```

## rst@extent

```
class      : Extent
xmin       : 4
xmax       : 44
ymin       : 8
ymax       : 56
```

```
xmax <- xllcorner + (NCOLS*cellsize)
ymax <- yllcorner + (NROWS*cellsize)
```

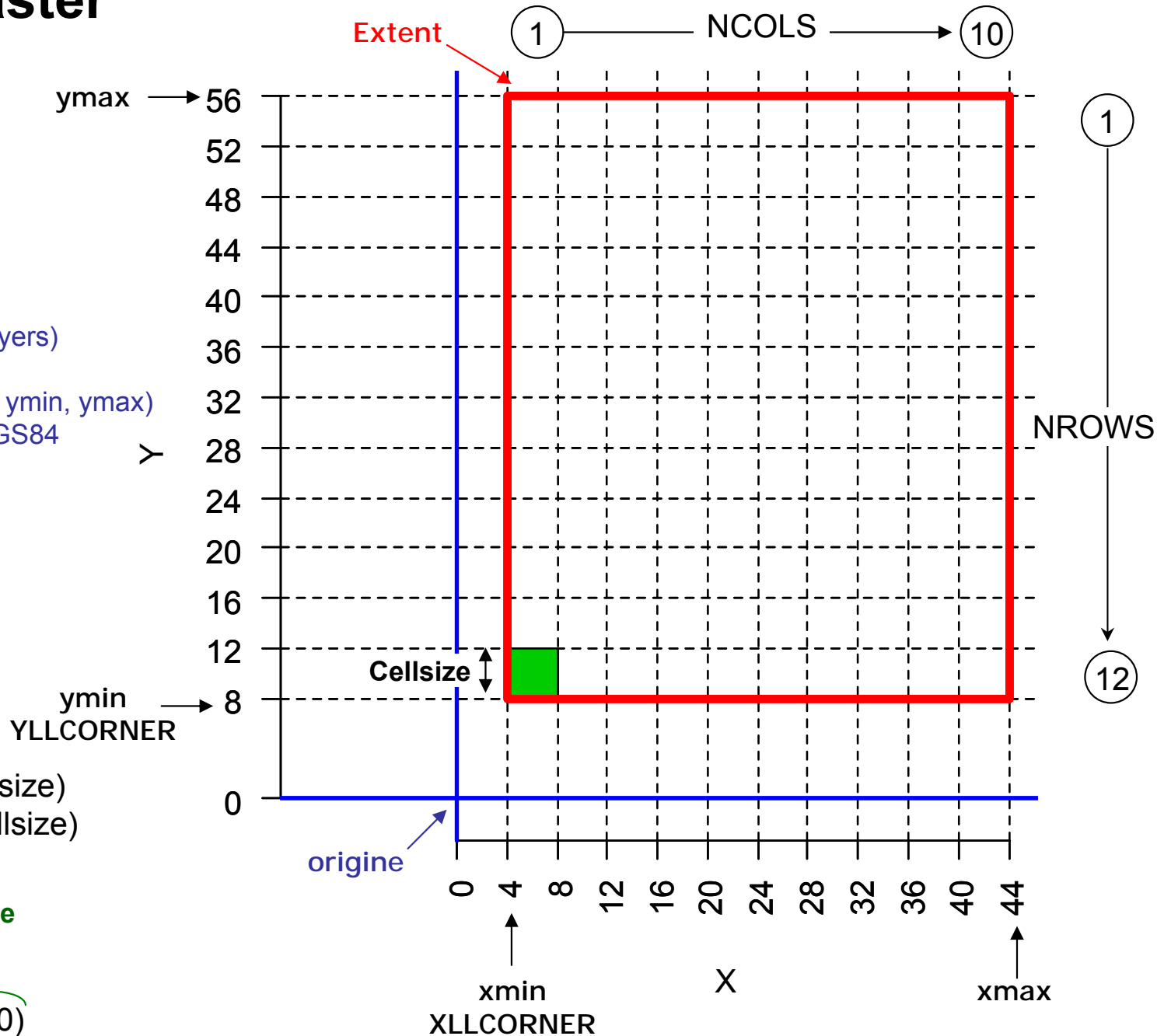
## origin (rst)

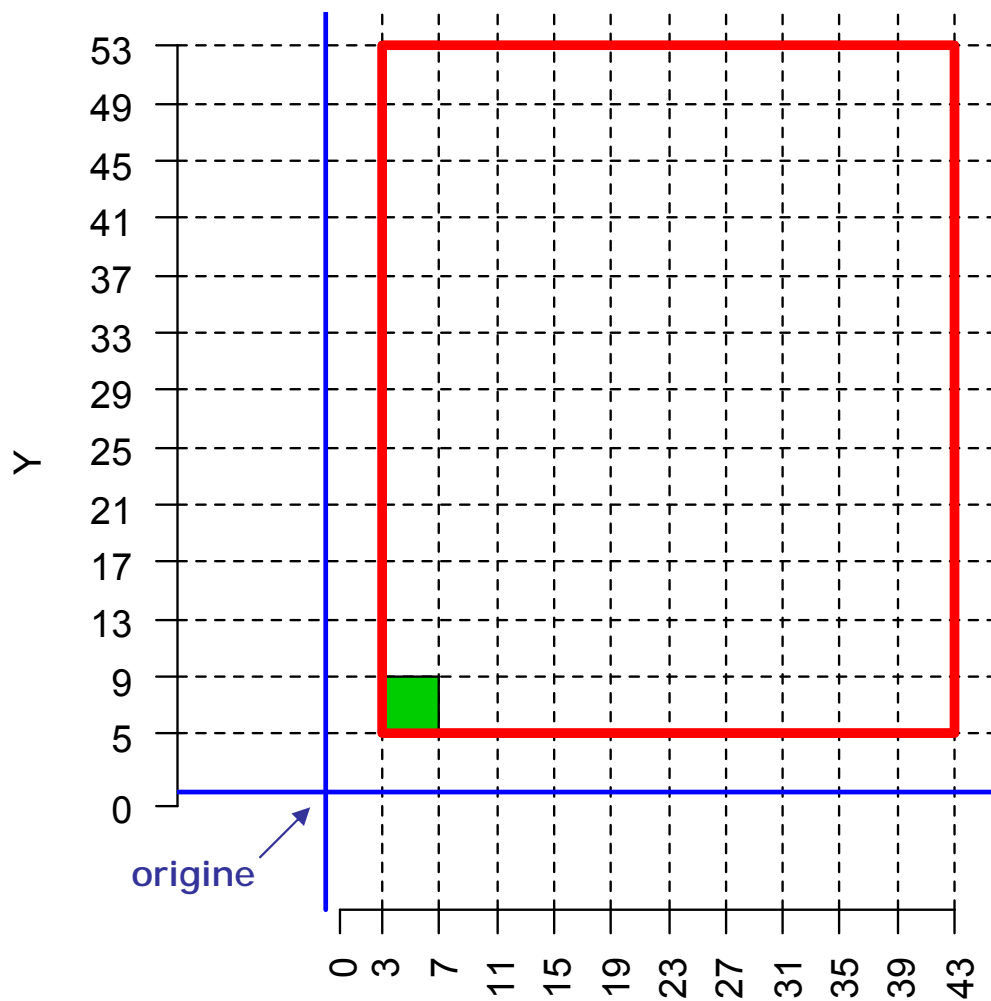
```
[1] 0 0
```

# Calcul de l'origine

nombre entier de cellule

```
x <- round (xllcorner / cellsize, 0)
Ox <- xllcorner - (x*cellsize) # origine en x
y <- round (yllcorner / cellsize, 0)
Oy <- yllcorner - (y*cellsize) # origine en y
```





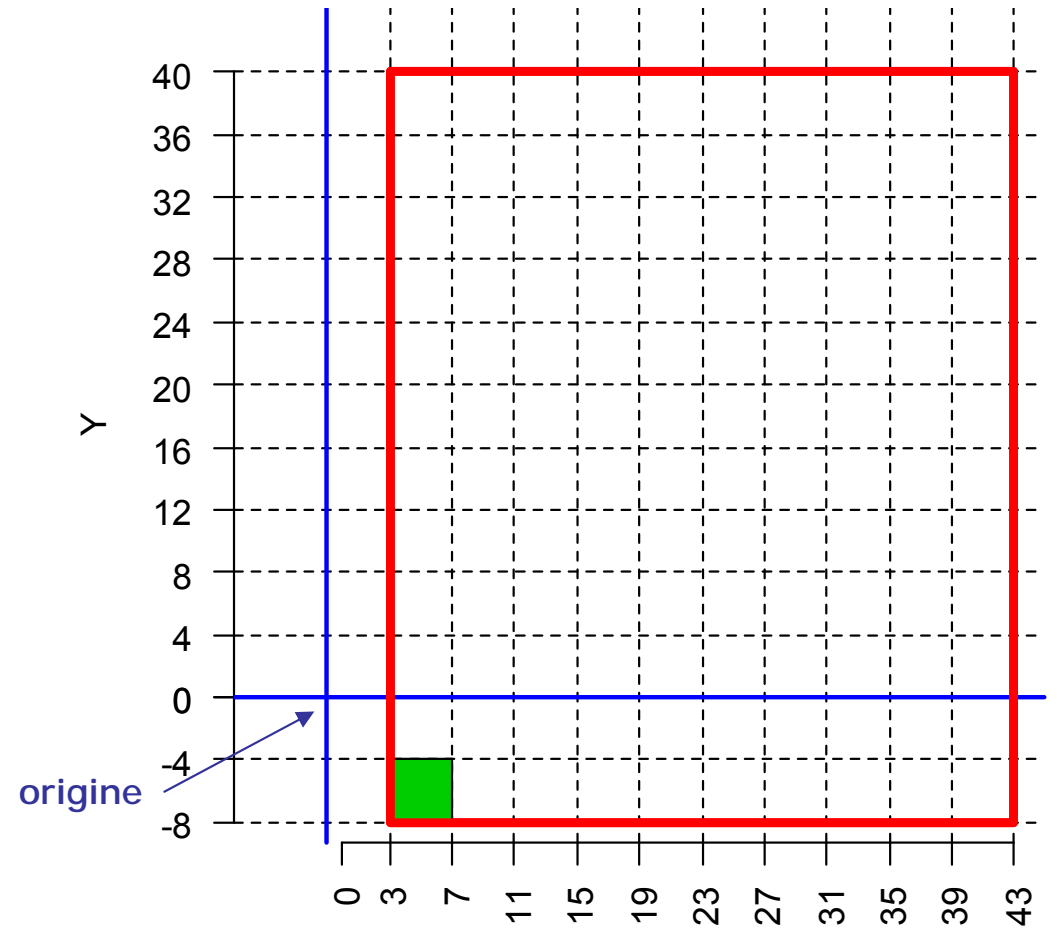
```

rst2.asc
NCOLS      10
NROWS      12
XLLCORNER  3
YLLCORNER  5
CELLSIZE   4
  
```

```

origin (rst)
-1 1
extent (rst)
class : Extent
xmin : 3
xmax : 43
ymin : 5
ymax : 53
  
```

### Remarque sur les coordonnées de l'origine d'un raster



```

rst4.asc
NCOLS      10
NROWS      12
XLLCORNER  3
YLLCORNER -8
CELLSIZE   4
  
```

```

origin (rst)
-1 0
extent (rst)
class : Extent
xmin : 3
xmax : 43
ymin : -8
ymax : 40
  
```

**MODIFIER  
DES  
OBJETS  
CARTOGRA  
PHIQUES**



# Systeme géographique de référence

library (rgdal)

Avoir le bon CRS !

## Accès aux données du fichier des codes de référence EPSG

```
EPSG <- make_EPSG ()
```

Recherche du bon code, exemple pour les projections Lambert xx

```
(EPSG_Lambert <- EPSG [grep("Lambert", EPSG$note), 1:2] )
```

Trois codes incontournables

rownames	Code EPSG	note	Arguments PROJ.4 pour la définition du système de référence (CRS)
249	<b>4326</b>	<b>WGS 84</b>	+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs
584	<b>2154</b>	<b>RGF93 / Lambert-93</b>	+proj=lcc +lat_1=49 +lat_2=44 +lat_0=46.5 +lon_0=3 +x_0=700000 +y_0=6600000 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs
2985	<b>27572</b>	<b>NTF (Paris) / Lambert zone II</b>	+proj=lcc +lat_1=46.8 +lat_0=46.8 +lon_0=0 +k_0=0.99987742 +x_0=600000 +y_0=2200000 +a=6378249.2 +b=6356515 +towgs84=-168,-60,320,0,0,0,0 +pm=paris +units=m +no_defs

WGS 84 : système géodésique mondial (World Geodesic System 1984). Celui de Google Earth.

NTF : Nouvelle Triangulation pour la France (1873 - 1991).

RGF : Réseau Géodésique Français (à partir de 1990).

RGF93 : Réseau géodésique français légal pour la métropole depuis le décret du 26.12.2000.

Lambert : système de projection utilisé pour la France (4 zones).

Lambert93 : projection associée au système géodésique RGF93 (compatible avec WGS84).

**L'ensemble des couches doit être dans le même système de projection**

# Changer de système de référence

pour des objets de classe : Spatial... [ librairie sp ]

library (rgdal)

ici : SpatialPolygonsDataFrame

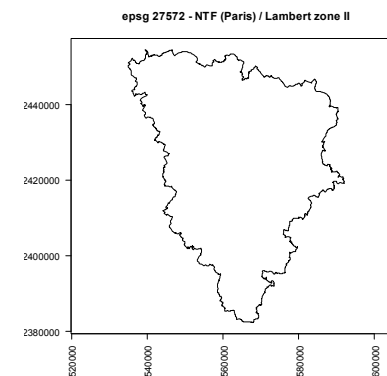
```
d78 <- readOGR ("Dept78.TAB", layer="Dept78")
```

```
proj4string (d78) # Affiche le système de référence (CRS)
```

```
"+proj=lcc +lat_1=45.898918964419 +lat_2=47.696014502038 +lat_0=46.8 +lon_0=0  
+x_0=600000 +y_0=2200000 +a=6378249.2 +b=6356515.000000472 +towgs84=-168,-  
60,320,-0,-0,-0,0 +pm=2.337229166667 +units=m +no_defs"
```

```
plot (d78, axes = TRUE, las=2)
```

```
title ("epsg 27572 - NTF (Paris) / Lambert zone II")
```



## ➤ Reprojecter en RGF93, code EPSG : 2154

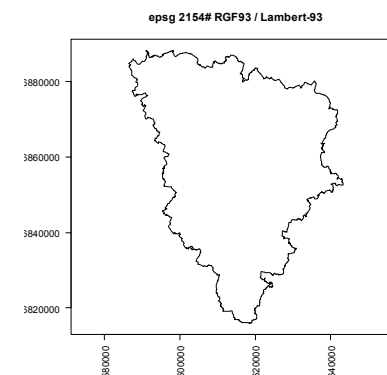
```
d78RGF93 <- spTransform (d78, CRS ("+init=epsg:2154"))
```

```
proj4string (d78RGF93)
```

```
"+init=epsg:2154 +proj=lcc +lat_1=49 +lat_2=44 +lat_0=46.5 +lon_0=3 +x_0=700000  
+y_0=6600000 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs"
```

```
plot (d78RGF93, axes = TRUE, las=2)
```

```
title ("epsg 2154 # RGF93 / Lambert-93")
```



## ➤ Reprojecter en WGS 84 , code EPSG : 4326

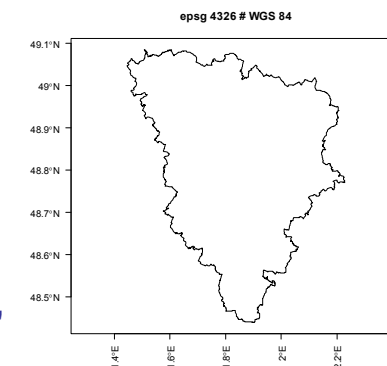
```
d78WGS84 <- spTransform (d78, CRS ("+init=epsg:4326"))
```

```
proj4string (d78WGS84)
```

```
"+init=epsg:4326 +proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs +towgs84=0,0,0"
```

```
plot (d78WGS84, axes = TRUE, las=2)
```

```
title ("epsg 4326 # WGS 84")
```



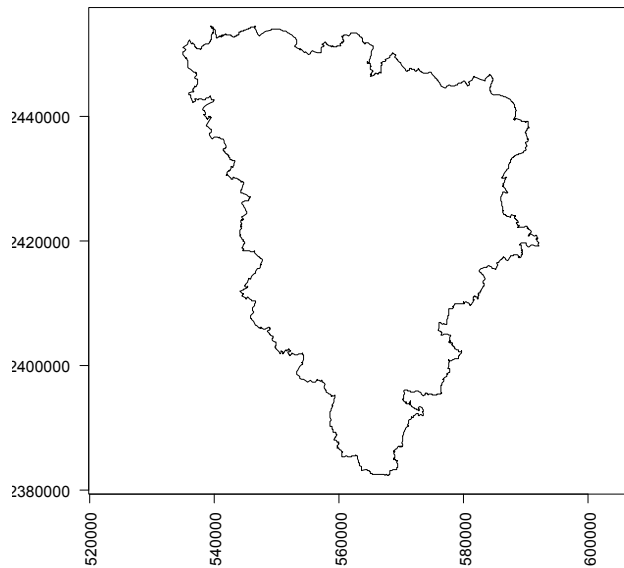
# Comparaison des représentations suivant le CRS

NTF / Lambert II  
code EPSG : 27572

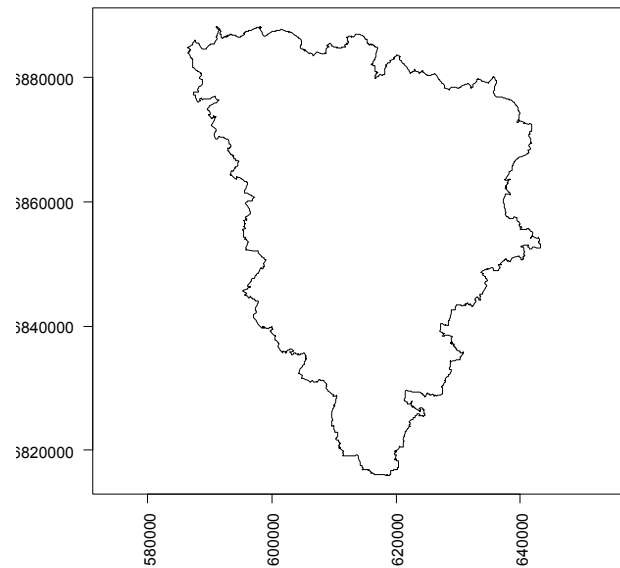
RGF93 / Lambert93  
code EPSG : 2154

WGS 84  
code EPSG : 4326

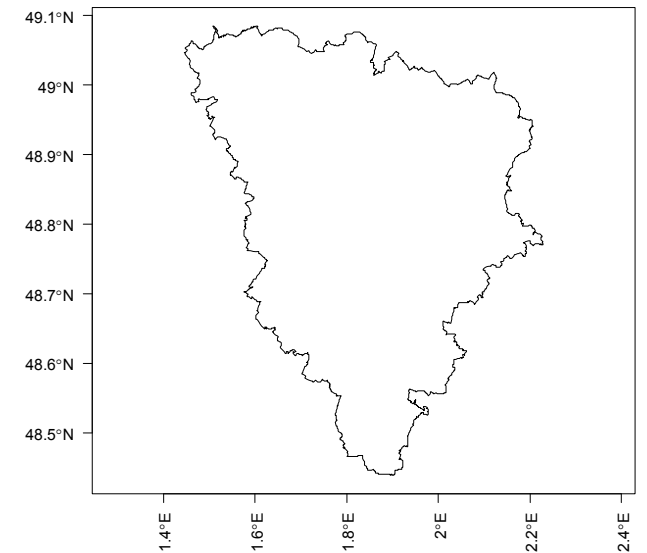
epsg 27572 - NTF (Paris) / Lambert zone II



epsg 2154# RGF93 / Lambert-93



epsg 4326 # WGS 84



## Emprise des cartes

bbox(d78)

	min	max
x	534891	592091
y	2382272	2454474

bbox(d78RGF93)

	min	max
x	586431.7	643333.8
y	6815921.5	6888312.4

bbox(d78WGS84 )

	min	max
x	1.446436	2.228964
y	48.438499	49.085414

# Changer de système de référence pour des objets de classe RasterLayer [ librairie raster ]

```
Veg <- raster ("d78Veg.asc")
```

```
projection (Veg) # [1] "NA" Affiche le système de référence
```

```
VegWGS84 <- projectRaster (Veg, crs = proj4string (d78WGS84))
```

raster d'origine

Nouveau système de référence. Ici, celui du SpatialPolygonsDataFrame d78WGS84

Mais

Erreur dans projectRaster :

input projection is NA

donc :

**1 - Affecter le BON système de référence**

```
projection (Veg) <- proj4string (d78)
```

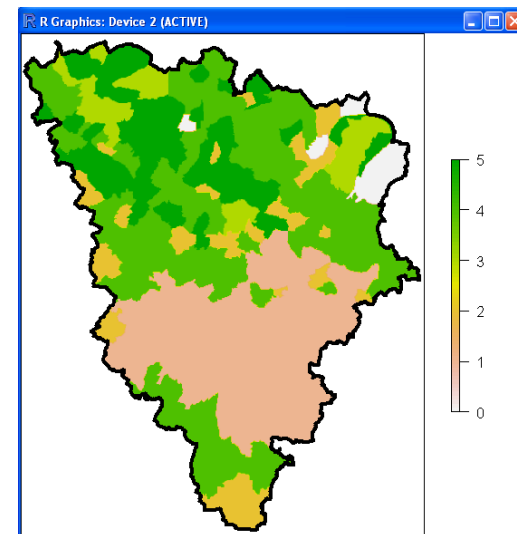
cohérent avec les coordonnées

Puis

**2 - Recalculer les coordonnées des cellules par la fonction projectRaster ()**

```
VegWGS84 <- projectRaster (Veg, crs = proj4string (d78WGS84))
```

```
projection (VegWGS84) # [1] "+init=epsg:4326 +proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs +towgs84=0,0,0 +over"
```



plot (VegWGS84)  
plot (d78WGS84, add=TRUE, lwd=4)

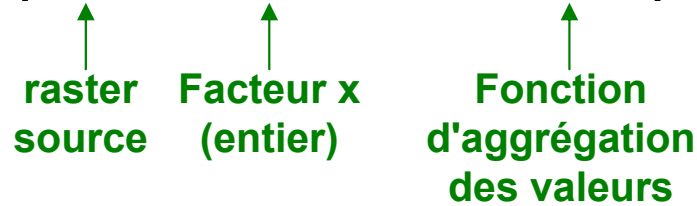
**Attention** les x, y de la résolution (pixel) peuvent devenir différents. Problème pour l'écriture au format asc

# Modifier la grille d'un raster

## Diminuer la résolution

```
rst1 <- raster ("rst1.asc")
```

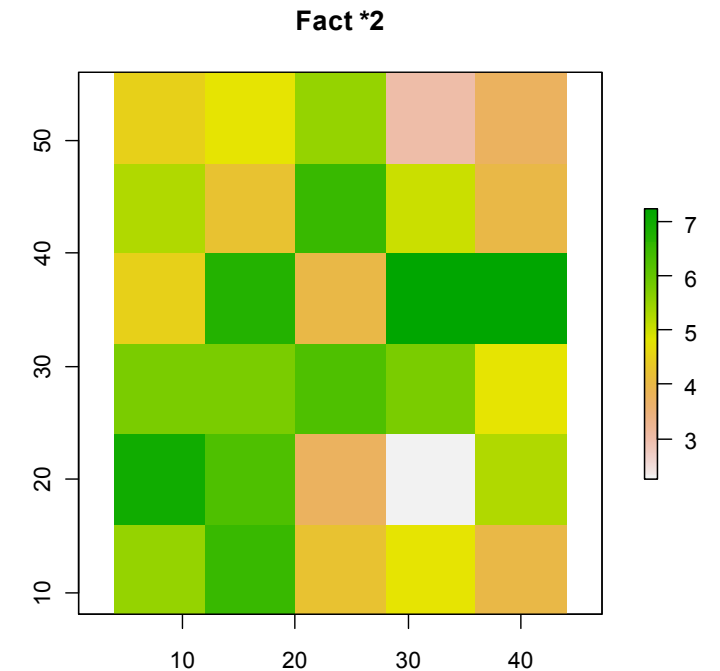
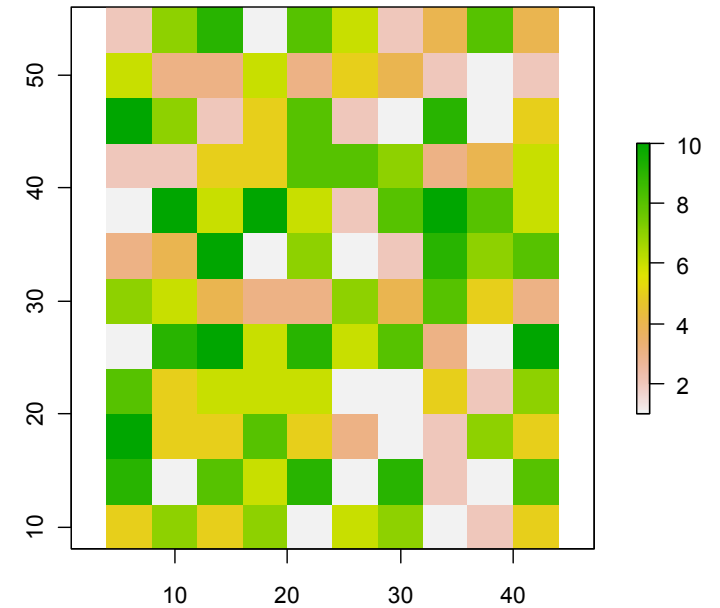
```
rstA2 <- aggregate (rst1, fact=2, fun=mean)
```



```
plot (rst1, asp=1)
x11()
plot (rstA2, asp=1, main="Fact *2")
```

	<b>rst1</b>	<b>rstA2</b>	
class	: RasterLayer	: RasterLayer	
dimensions	: 12, 10, 1	: 6, 5, 1	(nrow, ncol, nlayers)
resolution	: 4, 4	: 8, 8	(x, y)
extent	: 4, 44, 8, 56	: 4, 44, 8, 56	(xmin, xmax, ymin, ymax)
min value	: 1	: 2.25	
max value	: 10	: 7.25	

# resample pour modifier l'origine et la résolution et croiser les valeurs



Modifier la grille d'un raster

## Diminuer la résolution : exemple

```
d78Veg <- raster ("d78Veg.asc",crs="+init=epsg:27572")
```

```
d78Veg10 <- aggregate (d78Veg, fact=10, fun=mean)
```

```
plot (d78Veg10, main="d78Veg10")
```

```
plot (d78Veg, main="d78Veg")
```

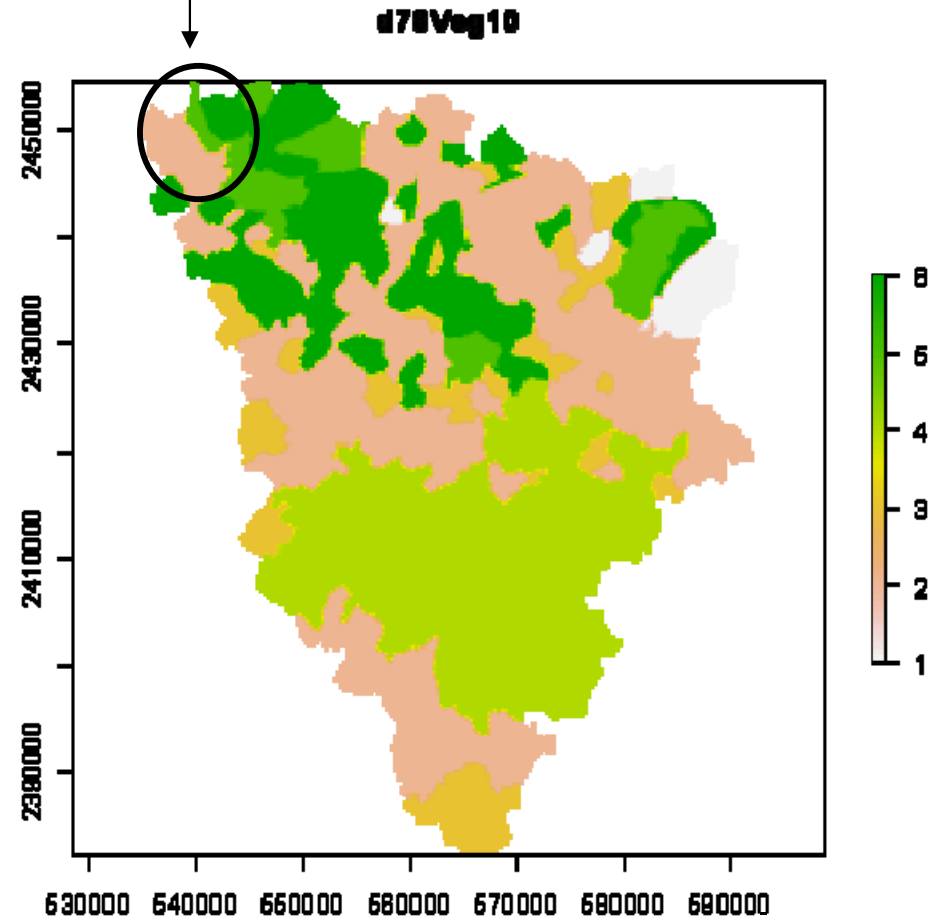
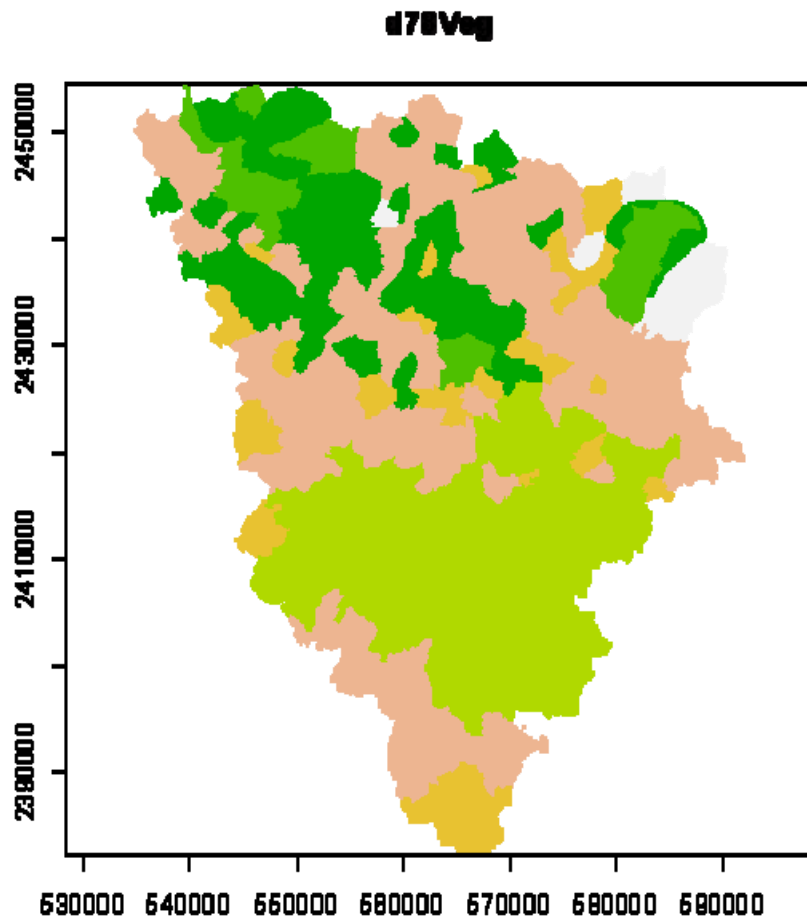
**d78Veg**

dimensions : 2408, 1907, 1  
resolution : 30, 30

**d78Veg10**

dimensions : 241, 191, 1  
resolution : 300, 300

Problème de valeur pour les  
pixels en limite de zone



```
writeRaster (d78Veg10, filename="d78Veg10.asc")
```

# Modifier la grille d'un raster

## Augmenter la résolution

```
rst1 <- raster ("rst1.asc")
```

```
rstD2 <- disaggregate (rst1, fact=2, method='ngb')
```

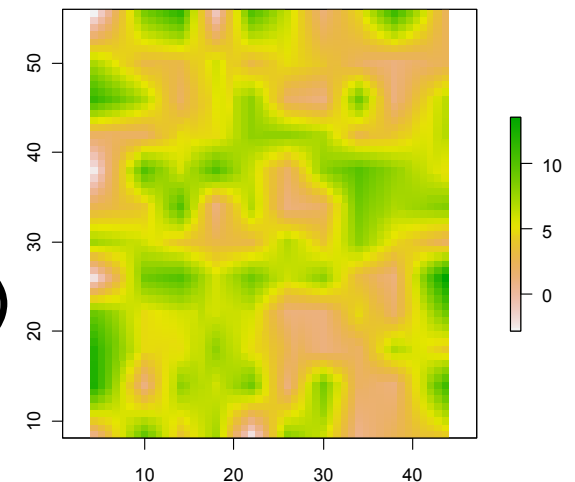
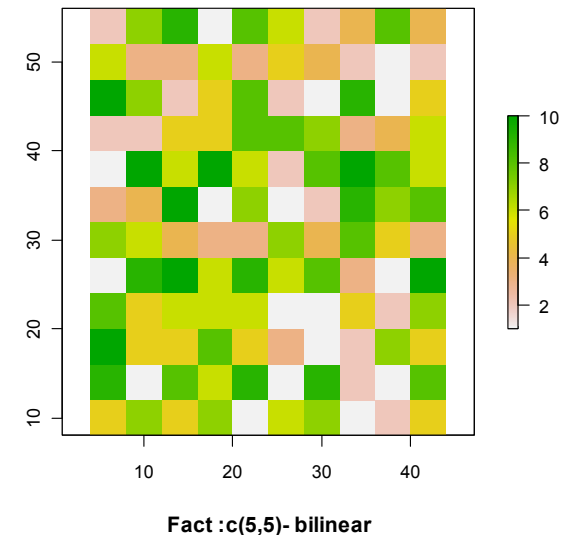
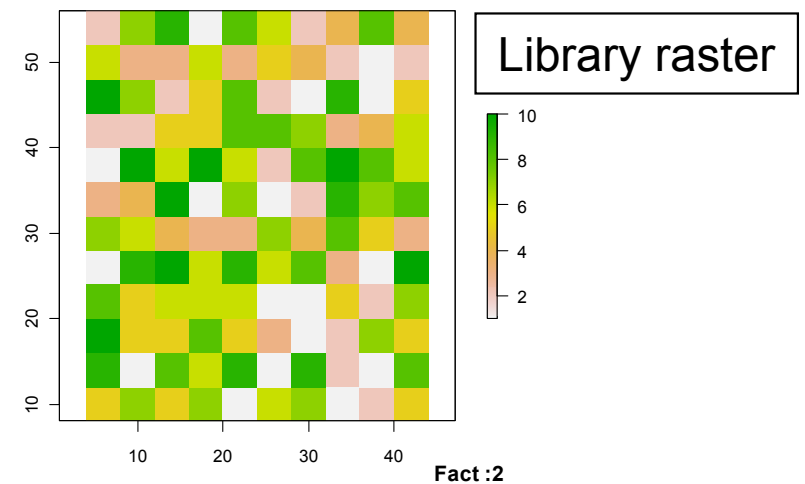
↑ raster source    ↑ Facteur : (entier)    ↑ Calcul des valeurs

	<b>rst1</b>	<b>rstD2</b>	
class	: RasterLayer	: RasterLayer	
dimensions	: 12, 10, 1	: 24, 20, 1	(nrow, ncol, nlayers)
resolution	: 4, 4	: 2, 2	(x, y)
extent	: 4, 44, 8, 56	: 4, 44, 8, 56	(xmin, xmax, ymin, ymax)
min value	: 1	: 1	
max value	: 10	: 10	

```
plot (rstD2, asp=1)
```

```
rstD2b <- disaggregate(rst1, fact=c(5,5), method='bilinear')
```

```
plot (rstD2b, asp=1, main="Fact :c(5,5)- bilinear")
```



CREER  
DES  
OBJETS  
CARTOGR  
APHIQUES



# Caler un raster image

## Les données :

- un fichier image (bmp)  
(carte communale de la végétation des Yvelines)

```
Veg <- raster ("Veg78.bmp")
```

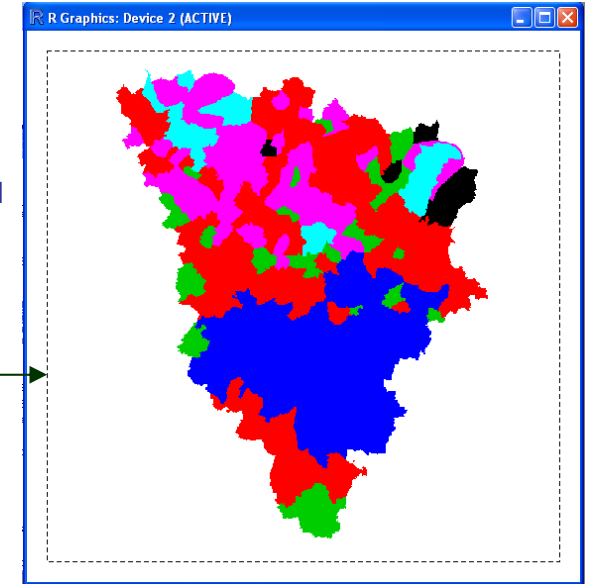
```
plot(Veg)
```

```
bVeg <- as.vector (bbox (Veg) ) # emprise de l'image
```

```
rect (bVeg[1], bVeg[2], bVeg[3], bVeg[4], border=1, lty=2)
```

### Veg

```
class      : RasterLayer  
dimensions : 671, 672, 1  
resolution : 1, 1 (x, y)  
extent     : 0, 672, 0, 671  
projection : NA  
values     : Veg78.bmp  
min value  : 0  
max value  : 255
```



## 1 - Isoler la carte du fond de l'image

```
hist (Veg)
```

```
Veg [ which (values (Veg) == 6)] <- NA
```

```
Veg <- trim (Veg) # suppression des NA
```

### extent (Veg)

```
class  : Extent  
xmin   : 89  
xmax   : 577  
ymin   : 31  
ymax   : 647
```

**Nouvelle  
emprise**

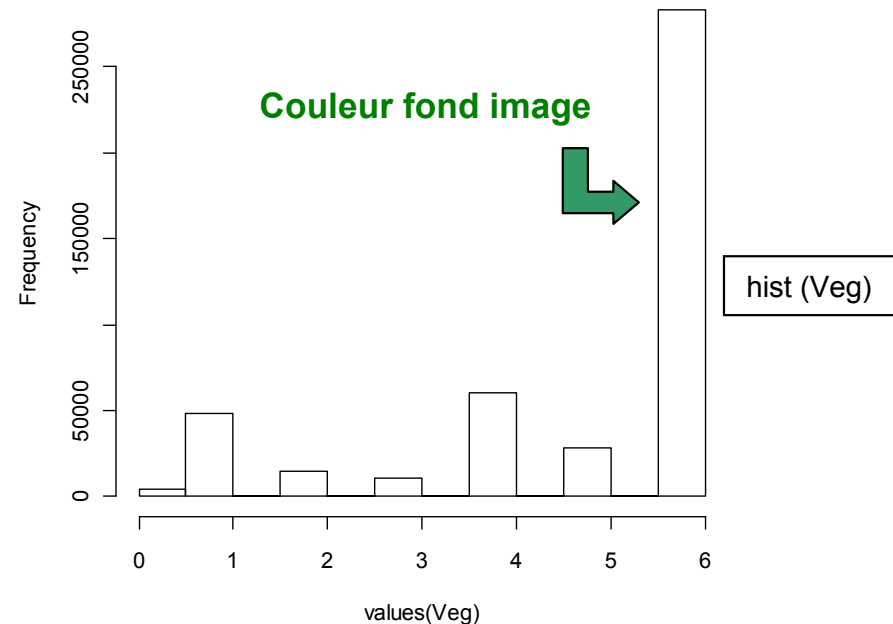
### res (Veg)

```
[1] 1 1
```

### origin (Veg)

```
[1] 0 0
```

Histogram of values(Veg)



# Caler un raster image

## 2 - Ajuster le raster sur la référence (raster)

```
d78Veg10 <- raster("d78Veg10.asc", crs="+init=epsg:27572") # raster source
```

```
d78Veg10  
dimensions : 241, 191, 1  
resolution : 300, 300  
extent : 534895, 592195, 2382200, 2454500
```

**Création d'un raster de référence sans valeur** (reprenant ici les caractéristiques du raster source)

```
r <- raster (nrows= nrow(d78Veg10), ncols= ncol(d78Veg10),  
            xmn= xmin(d78Veg10), xmx= xmax(d78Veg10), ymn= ymin(d78Veg10), ymx= ymax(d78Veg10),  
            crs= "+init=epsg:27572" )
```

**Ajuster les emprises (étendues)**

```
extent (Veg) <- extent (r)
```

**Création du raster par échantillonnage**

fonction resample ()

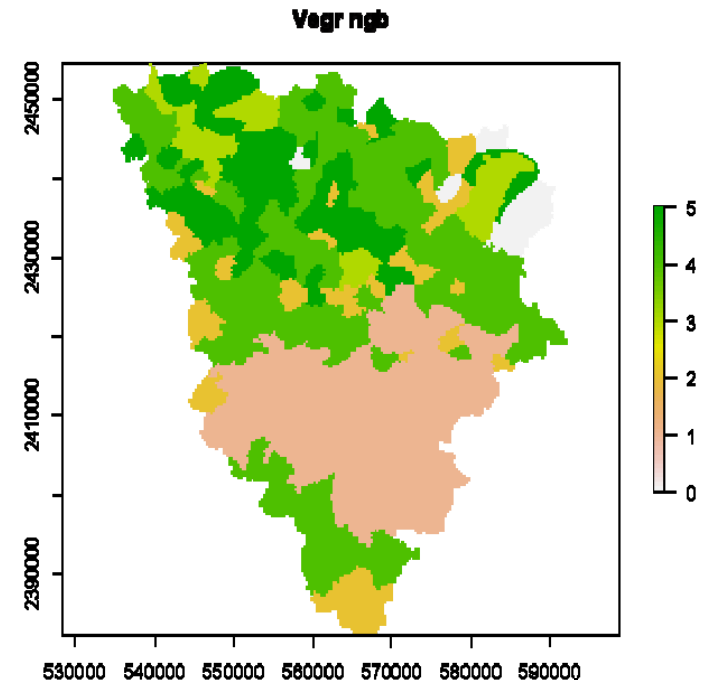
```
Vegr <- resample (Veg, r, method='ngb')
```

objet échantillonné  
Cellules et valeurs recalculées

raster de  
référence

méthode de calcul  
des valeurs

```
Vegr  
class : RasterLayer  
dimensions : 241, 191, 1 (nrow, ncol, nlayers)  
resolution : 300, 300 (x, y)  
extent : 534895, 592195, 2382200, 2454500 (xmin, xmax, ymin, ymax)  
projection : +init=epsg:27572 +proj=lcc +lat_1=46.8 +lat_0=46.8 +lon_0=0 +k_0=0.99987742 +x_0=600000  
+y_0=2200000 +a=6378249.2 +b=6356515 +towgs84=-168,-60,320,0,0,0,0 +pm=paris +units=m +no_defs  
values : in memory  
min value : 0  
max value : 5
```



```
plot( Vegr, main="Vegr ngb")
```

# Créer un raster à partir de données existantes

## Les données :

- une carte (objet `SpatialGridDataFrame`) des communes (n polygones)
- un fichier de valeurs par commune
- un raster de référence

### Lecture de la carte des communes

```
c78 <- readOGR ("Com78.MIF", layer="Com78")
```

### Lecture des données caractérisant les communes

```
nbi <- read.table ("nbi_CAH.txt")
```

### Introduction dans les données attributaires

```
c78@data <- cbind (c78@data, nbi$classe)
```

```
str (c78@data)
```

### Lecture du raster de référence

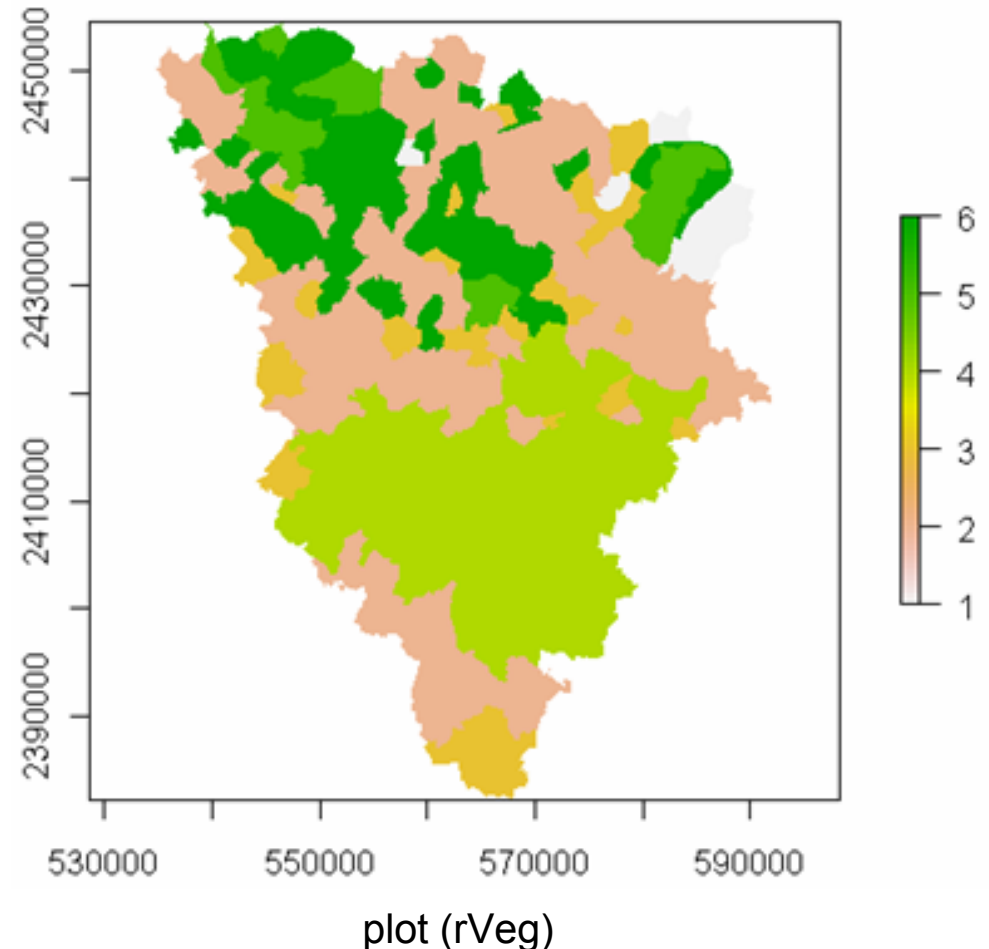
```
d78alt <- raster ("d78alt.asc")
```

## Rasterisation du `SpatialGridDataFrame`

```
rVeg <- rasterize (c78, d78alt, field=10)
```

Objet Spatial...	Raster de référence	Champ de la valeur
---------------------	------------------------	-----------------------

```
writeRaster (rVeg, filename="d78Veg.asc")
```



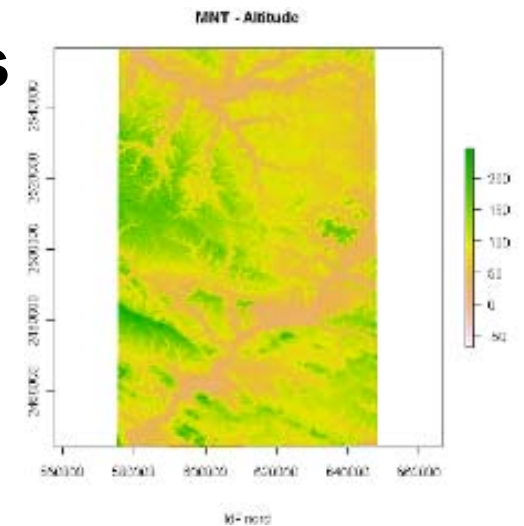
```
origin (rVeg)  
[1] -5.026339 -9.726538  
origin (d78alt)  
[1] -5.026339 -9.726538
```

# Créer un raster à partir de données existantes

## Les données :

- contour des Yvelines (1 polygone : SpatialPolygonsDataFrame)
- 4 rasters (MNT - altitude \*) couvrant ce département

```
d78 <- readOGR ("Dept78.TAB", layer="Dept78")
rn  <- raster ("aster_idf_nord.txt", crs= "+init=epsg:27572")
rnw <- raster ("aster_idf_nw.txt", crs= "+init=epsg:27572")
rwy <- raster ("aster_idf_ouest_yvelines.asc", crs= "+init=epsg:27572")
rs  <- raster ("aster_idf_sud.txt", crs= "+init=epsg:27572")
```

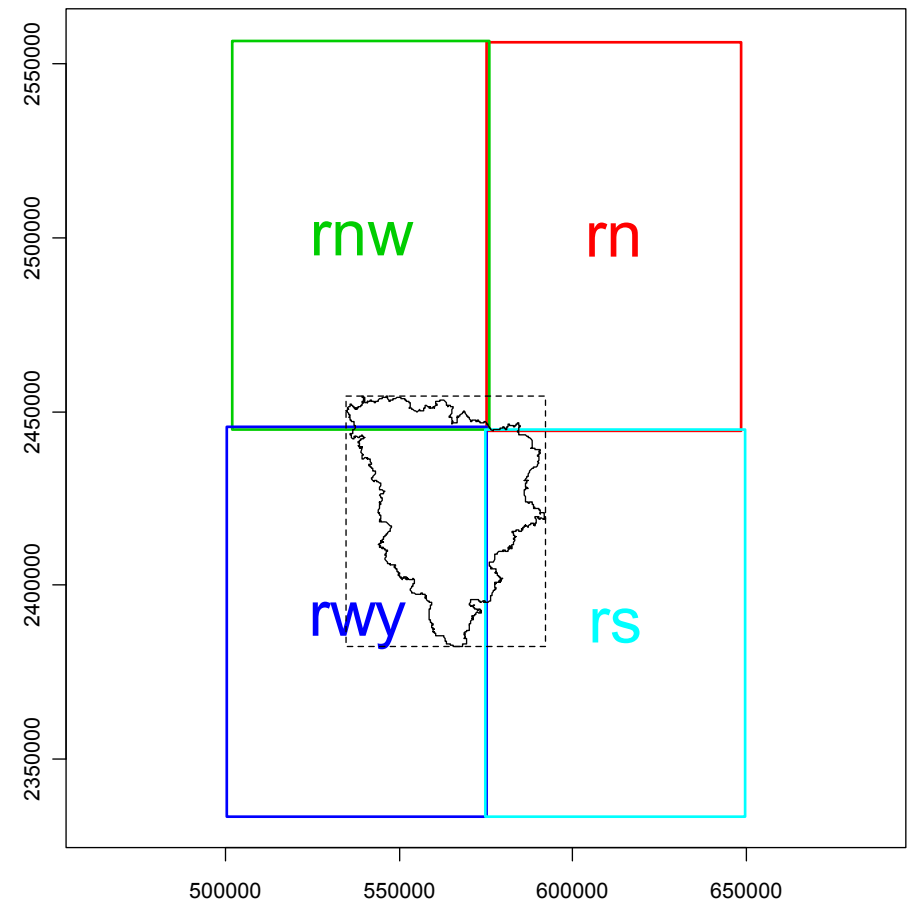


plot (rn, main = "MNT - Altitude", sub="IdF nord")

## Objectif : avoir le MNT des Yvelines

```
bd78 <- as.vector (bbox(d78))      emprise des Yvelines
```

```
n <- c("rn","rnw","rwy","rs")
rec <- rbind (as.vector(bbox(m))
             ,as.vector(bbox(rnw))
             ,as.vector(bbox(rwy))
             ,as.vector(bbox(rs)))
par (mar=c(2.2,2.2,.2,.2))
plot (c(min(rec[,1]),max(rec[,3]),c(min(rec[,2]), max(rec[,4])), type="n", asp=1)
for (i in 1:nrow(rec)) {
  rect (rec[i,1],rec[i,2],rec[i,3],rec[i,4], border=i+1, lwd=2)
  text (mean(rec[i,c(1,3)]), mean (rec[i,c(2,4)]), labels=n[i], col=i+1, cex=3) }
plot (d78,add=TRUE)
rect (bd78[1], bd78[2], bd78[3], bd78[4], border=1, lty=2)
```



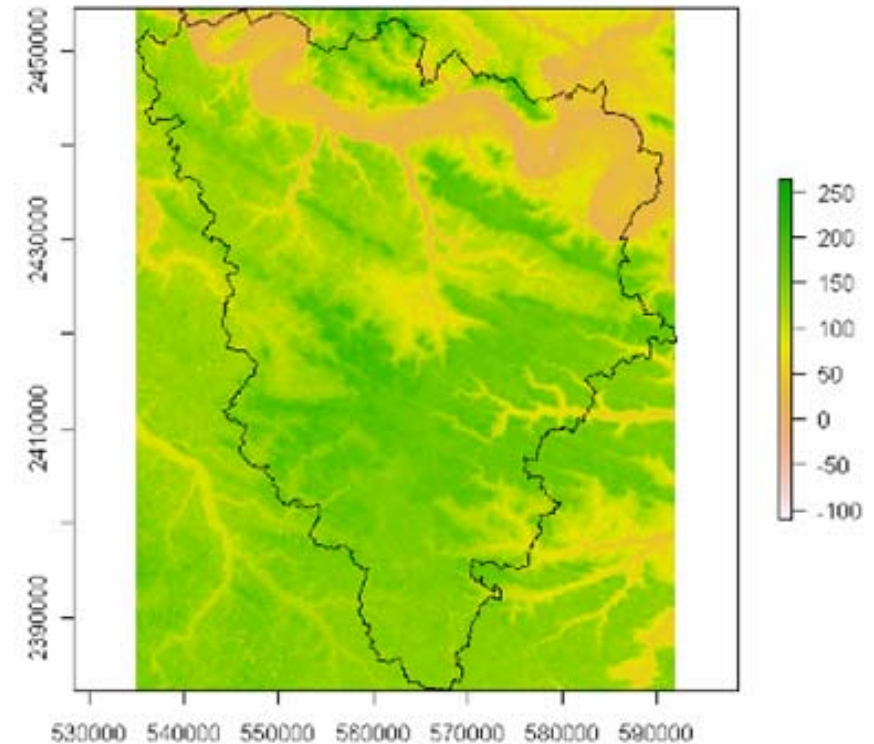
(\*) ASTER Global Digital Elevation Model (ASTER GDEM), modèle numérique de terrain (MNT) disponible sur le monde entier  
<http://www.ersdac.or.jp/GDEM/E/index.html>

# 1 - Découper les rasters par l'emprise du polygone

## Fonction crop ()

```
rd78n  <- crop (rn, extent (bd78[1],bd78[3],bd78[2],bd78[4]))
rd78nw <- crop (rnw, extent (bd78[1],bd78[3],bd78[2],bd78[4]))
rd78wy <- crop (rwy, extent (bd78[1],bd78[3],bd78[2],bd78[4]))
rd78s  <- crop (rs, extent (bd78[1],bd78[3],bd78[2],bd78[4]))
```

↑ raster source    ↑ emprise  
xmin    xmax    ymin    ymax



# 2 - Fusionner les rasters

## Fonction merge ()

```
rd78M <- merge (rd78n, rd78nw, rd78wy, rd78s, tolerance=1)
```

} rasters à fusionner    Tolérance % taille cellule ↑ admissible

```
plot (rd78M, asp=1 )
plot (d78, add=TRUE)
```

Les rasters devraient avoir même origine et même résolution

Ici même résolution

resolution : 30, 30 (x, y)

mais origines différentes

cellsize = (rec[1,3]-rec[1,1]) / ncol(rn)

origin (rn)	#	10.517 -9.020
origin (rnw)	#	-2.671439 7.268528
origin (rwy)	#	-5.026339 -9.726538
origin (rs)	#	13.678634 -7.611634

Créer un raster à partir de données existantes

### 3 - Pixellisation d'un objet spatial

#### Fonction rasterize ()

`class (d78)`

`[1] "SpatialPolygonsDataFrame"`

```
d78alt <- rasterize (d78, rd78M , mask=TRUE)
```

↑ objet Spatial  
↑ raster à découper  
↑ Option Si TRUE : Utilise valeurs du raster

`d78alt`

`class` : RasterLayer

`dimensions` : 2408, 1907, 1 (nrow, ncol, nlayers)

`resolution` : 30, 30 (x, y)

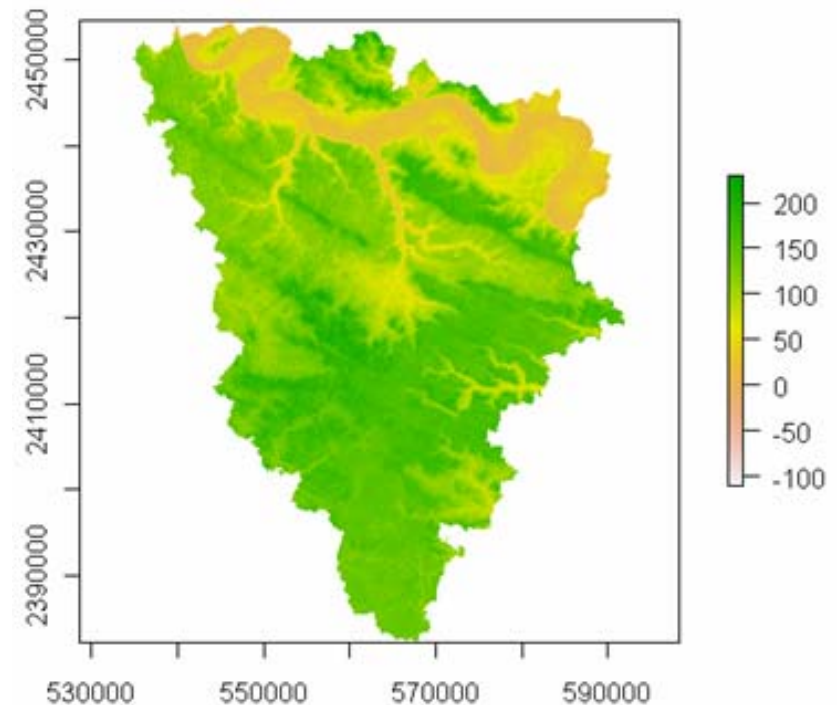
`extent` : 534895, 592105, 2382260, 2454500 (xmin, xmax, ymin, ymax)

`projection` : +proj=lcc +lat\_1=45.898918964419 +lat\_2=47.696014502038 +lat\_0=46.8 +lon\_0=0 +x\_0=600000  
+y\_0=2200000 +a=6378249.2 +b=6356515.000000472 +towgs84=-168,-60,320,-0,-0,-0,0  
+pm=2.337229166667 +units=m +no\_defs

`values` : in memory

`min value` : -111

`max value` : 231



`plot (d78alt, asp=1)`

Ecriture du raster au format asc

```
writeRaster (d78alt, filename="d78alt.asc")
```

# Créer des objets : POINTS

library (rgdal)

plot (d78WGS84, axes = TRUE)

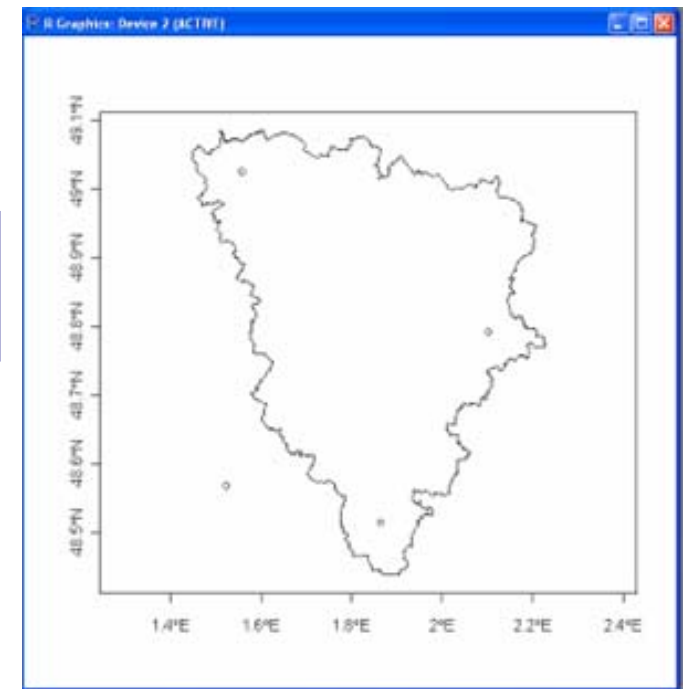
(pt <- locator (type = "p"))

Clic droit pour arrêter

```
$x  
[1] 1.556412 2.103122 1.862998 1.522108  
$y  
[1] 49.02610 48.79291 48.51591 48.56961
```

pt <- data.frame (x = pt\$x, y = pt\$y)  
(pt <- rbind (pt, c(1.84, 48.8)))  
coordonnées d'un 5<sup>ème</sup> point

	x	y
1	1.556412	49.02610
2	2.103122	48.79291
3	1.862998	48.51591
4	1.522108	48.56961
5	1.840000	48.80000



## ➤ 2 méthodes

pt <- **SpatialPoints** (pt, proj4string = CRS (proj4string (d78WGS84)))

Data.frame des coordonnées

Système de référence  
du fond cartographique

# ou

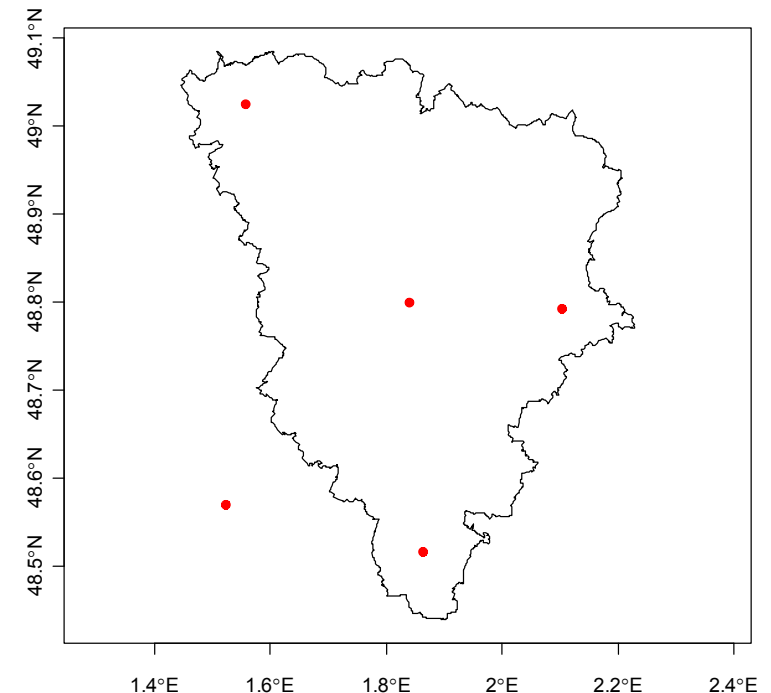
# transformation en objet de classe SpatialPoints

**coordinates** (pt) <- ~ x + y

# affectation du BON système de projection

**proj4string** (pt) <- CRS (proj4string (d78WGS84))

plot (pt, col=2, pch=19, add=TRUE)



SpatialPointsDataFrame() permet d'associer des données attributaires

# Objet de classe SpatialPoints (librairie sp)

library (rgdal)

str (pt)

```
Formal class 'SpatialPoints' [package "sp"] with 3 slots
..@ coords      : num [1:5, 1:2] 1.56 2.1 1.86 1.52 1.84 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:2] "x" "y"
..@ bbox        : num [1:2, 1:2] 1.52 48.52 2.1 49.03
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:2] "x" "y"
.. .. ..$ : chr [1:2] "min" "max"
..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
.. .. ..@ projargs: chr " +init=epsg:4326 +proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs
+towgs84=0,0,0"
```

slot (pt, "bbox")

# identique à pt@bbox

```
      min      max
x 1.522108 2.103122
y 48.515905 49.026098
```

pt@coords

# identique à slot (pt, "coords")

```
      x      y
[1,] 1.556412 49.02610
[2,] 2.103122 48.79291
[3,] 1.862998 48.51591
[4,] 1.522108 48.56961
[5,] 1.840000 48.80000
```



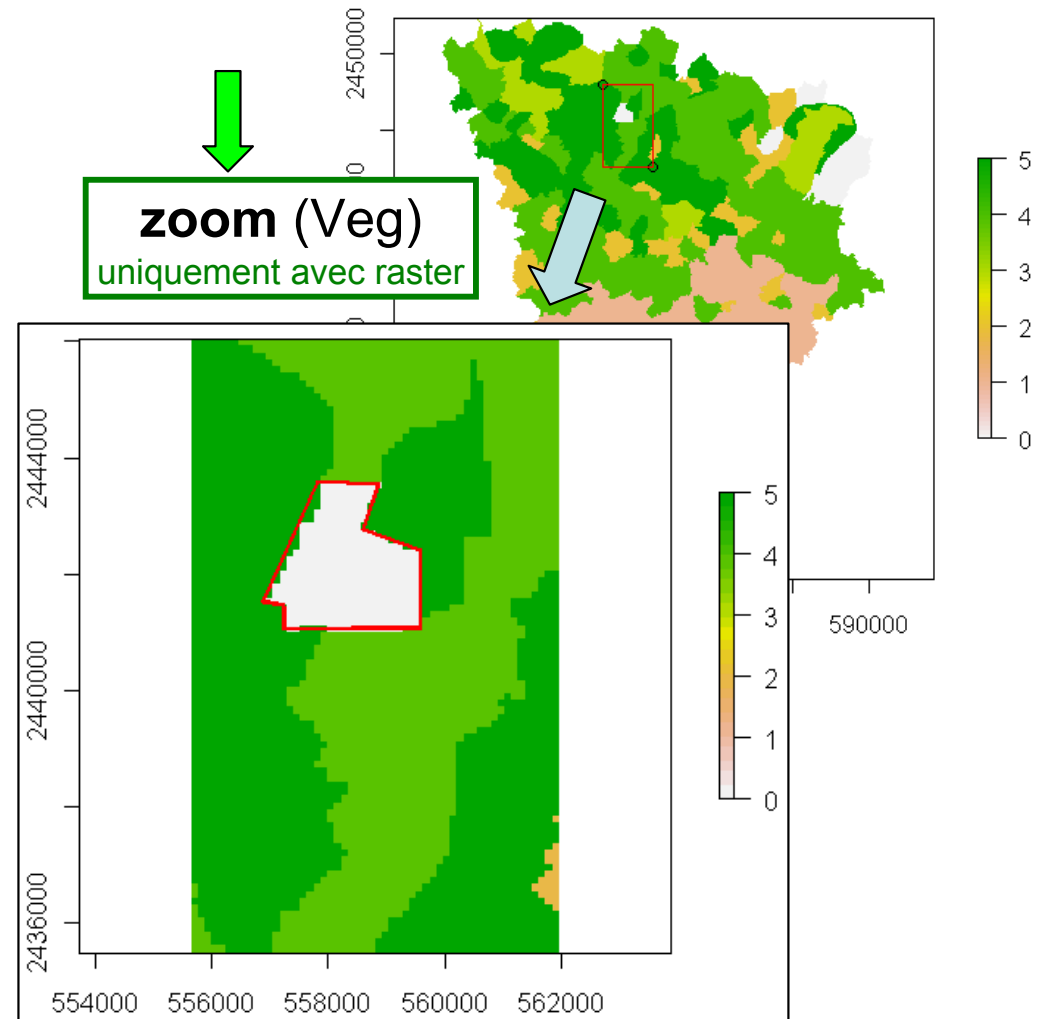
# Créer un POLYGONE

library (raster)

```
d78 <- readOGR ("Dept78.TAB", layer="Dept78")  
Veg <- raster ("VegImg.asc")  
projection (Veg) <- proj4string (d78)  
plot (Veg)  
zoom (Veg)
```

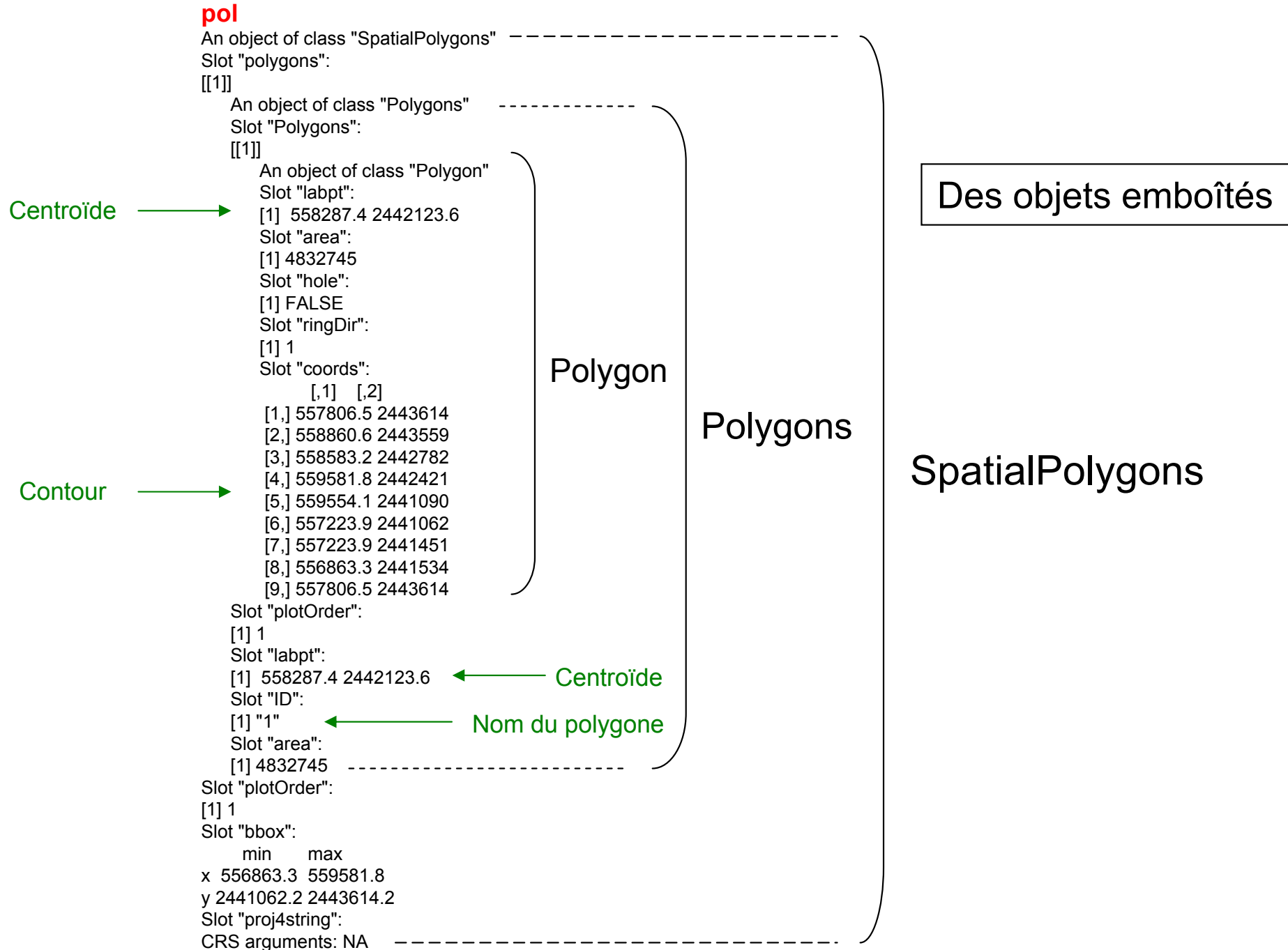
```
pol <- drawPoly (sp = TRUE)
```

Si TRUE : création d'un  
objet SpatialPolygons  
(bibliothèque sp)



```
plot(d78,lwd=2)  
plot(pol, add=TRUE,border=2,col="grey")
```

# Objet de classe SpatialPolygons (librairie sp)



# Créer des objets : POLYGONES

library (raster)

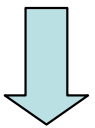
`mpol <- drawPoly (sp = FALSE)` ←

Si FALSE : création d'une matrice des coordonnées

```
mpol
      [,1] [,2]
[1,] 557806.5 2443587
[2,] 558832.8 2443559
[3,] 558583.2 2442782
[4,] 559554.1 2442394
[5,] 559581.8 2441118
[6,] 557251.7 2441034
[7,] 557251.7 2441478
[8,] 556835.6 2441562
[9,] 557806.5 2443587
```

## Création d'un objet classe Polygon

library (rgdal)



`P <- Polygon (mpol)` ←

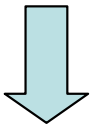
Transforme la matrice de coordonnées en objet de classe Polygon

## Création d'un objet de classe Polygons

Pourquoi cette classe?  
Permettre la gestion des polygones disjoints

`Ps <- Polygons (list (P), "mpol")`

Liste des objets de classe Polygon  
Nom du polygone (ID)



## Création d'un objet spatial de classe SpatialPolygons

`pol <- SpatialPolygons (list (Ps), proj4string=CRS("+init=epsg:27572"))`

Liste des objets de classe Polygons  
Le BON système de référence celui de la carte du tracé !

# Créer des objets : POLYGONES

library(rgdal)

```
plot(d78,lwd=2)
mpol1 <- drawPoly (sp = FALSE, col=1)
mpol2 <- drawPoly (sp = FALSE, col=2)
mpol3 <- drawPoly (sp = FALSE, col=3)
```

## Objet classe Polygon

```
P1 <- Polygon (mpol1)
P2 <- Polygon (mpol2)
P3 <- Polygon (mpol3)
```

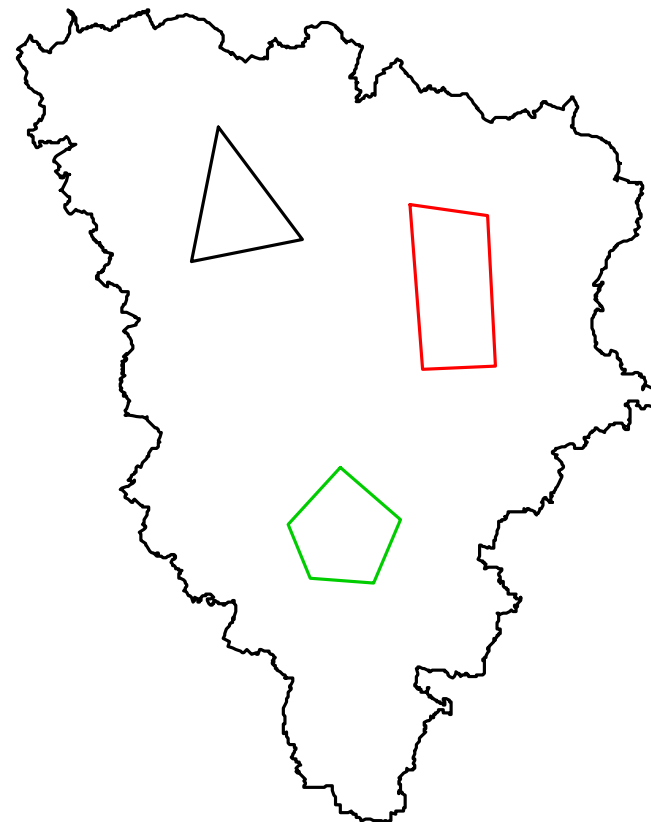
## Objet de classe Polygons

```
Ps12 <- Polygons (list (P1, P2), "Pm") # polygone disjoint composé de P1, P2
Ps3 <- Polygons (list (P3) , "P3")
```

## Objet spatial de classe SpatialPolygons

```
polSp <- SpatialPolygons (list (Ps12, Ps3), proj4string = CRS(proj4string (d78)))
```

↑  
Système de référence du fond cartographique



# Objet de classe SpatialPolygons [librairie sp]

`str (polSp)`

## Nom des polygones

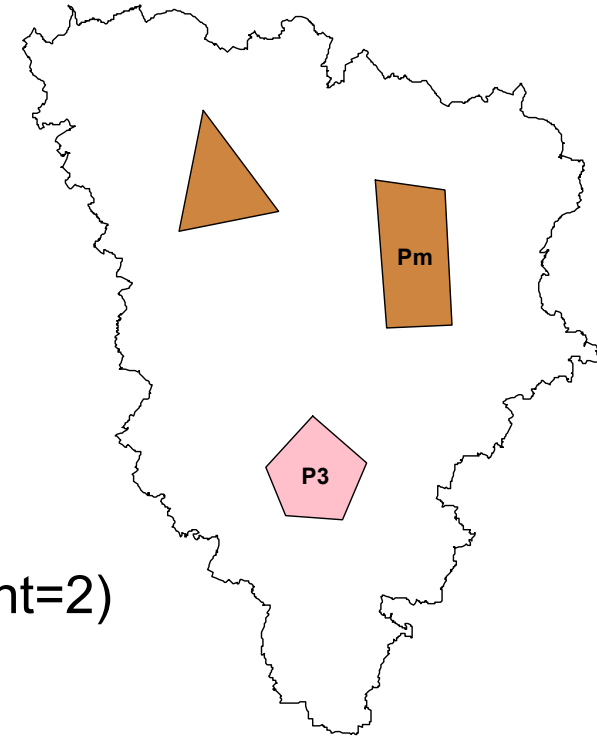
`row.names (polSp)`

```
[1] "Pm" "P3"
```

`plot (d78)`

`plot (polSp, add=TRUE, col = c("peru","pink"))`

`text (coordinates (polSp), labels = row.names (polSp), font=2)`



## Accès aux polygones par leur nom (ID)

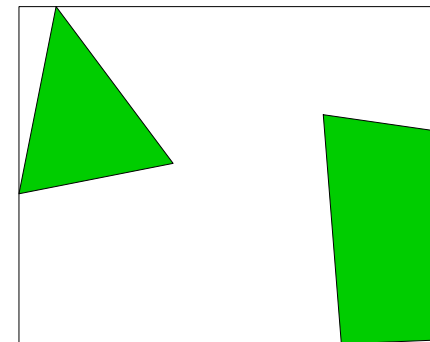
`polSp["P3"]@bbox`

```
      min      max  
x 558980 568988.7  
y 2403501 2413791.5
```

`plot (polSp ["Pm"], col=3)`

`y <- as.vector (polSp ["Pm"]@bbox)`

`rect (y[1], y[2], y[3], y[4])`



Remarque : pour un objet de classe SpatialPolygonsDataFrame, la sélection est réalisée comme pour un data.frame :  
`plot ( c78 [ "196", ] )`

# Créer des polygones à partir d'un fichier de coordonnées

```
MatCoord <- read.table ("MatCoord.txt")
```

```
# création du fichier de coordonnées
MatCoord <- rbind (
  cbind(as.data.frame(mpol1),pol="P1"),
  cbind(as.data.frame(mpol2),pol="P2"),
  cbind(as.data.frame(mpol3),pol="P3") )
write.table (MatCoord,"MatCoord.txt")
```

	V1	V2	pol
1	552777.3	2444100	P1
2	560248.7	2434091	P1
3	550380.9	2432117	P1
4	552777.3	2444100	P1
5	569834.5	2437192	P2
6	576742.0	2436206	P2
7	577446.8	2422814	P2
8	570962.3	2422532	P2
9	569834.5	2437192	P2
10	563631.9	2413792	P3
11	568988.7	2409140	P3
12	566592.3	2403501	P3
13	560953.5	2403924	P3
14	558980.0	2408717	P3
15	563631.9	2413792	P3

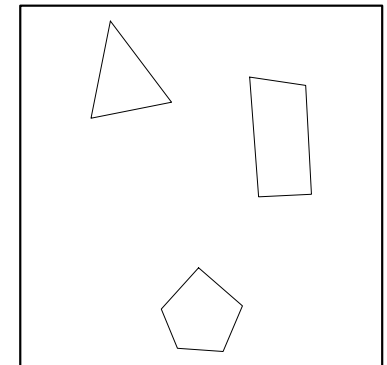
```
str (MatCoord)
```

```
'data.frame': 15 obs. of 3 variables:
 $ V1 : num 552777 560249 550381 552777 569835 ...
 $ V2 : num 2444100 2434091 2432117 2444100 2437192 ...
 $ pol: Factor w/ 3 levels "P1","P2","P3": 1 1 1 1 2 2 2 2 2 3 ...
```

## ► library (ade4)

```
area.plot (MatCoord [, c(3,1:2)] )
```

Fonction graphique  
Identificateur du polygone en  
1ère colonne et de classe "factor"

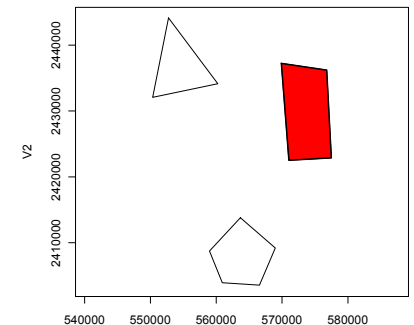


## ► library (adehabitat)

```
map <- as.area (MatCoord [, c(3,1:2)] )
```

Création d'un objet de classe :  
**class**(map)  
[1] "area" "data.frame"

```
plot.area (map, colpol=NA, lwd=1)
```



```
polmap <- area2spol (map)
```

```
class (polmap)
```

Création d'un objet de  
classe :  
[1] "SpatialPolygons"  
attr(,"package")  
[1] "sp"

```
plot.area (map, colpol=NA, lwd=1)
plot.area (map, which = "P2",
  colpol="red", add=TRUE)
```

# Fonctions et objets décrits

<b>Fonction</b>	<b>librairie</b>	<b>pages</b>
aggregate ()	raster	28, 29
area.plot ()	ade4	45
area2spol ()	adehabitat	45
as.area ()	adehabitat	45
bbox ()	sp	26, 32, 35
coordinates ()	rgdal	38
crop ()	raster	36
disaggregate ()	raster	30
drawPoly ()	raster	40, 42, 43
extent ()	raster	32, 33
import.asc ()	ade4habitat	12
make_EPSG ()	rgdal	24
merge ()	raster	36
origin ()	raster	36
plot.area ()	adehabitat	45
Polygon ()	rgdal	42, 43
Polygons ()	rgdal	42, 43
proj4string ()	rgdal	25, 38
projection ()	raster	27
projectRaster ()	raster	27
raster ()	raster	12, 20, 27, 28, 29, 30, 32, 33, 34, 35
rasterize ()	raster	34, 37
read.asc ()	SDMTools	12
readGDAL()	rgdal	12, 18
readOGR ()	rgdal	12, 13, 15, 16, 17, 25, 34, 35
readShapeSpatial ()	maptools	12, 16
resample ()	raster	33
slot ()	rgdal	39
SpatialPoints ()	rgdal	38
SpatialPolygons ()	rgdal	42, 43
spTransform ()	rgdal	25
trim ()	raster	32
writeRaster ()	raster	29, 34, 37
zoom ()	raster	40

<b>Objet de classe</b>	<b>librairie</b>	<b>pages</b>
RasterLayer	raster	21, 37
SpatialGridDataFrame	sp	19
SpatialPoint	sp	39
SpatialPolygons	sp	41, 44
SpatialPolygonsDataFrame	sp	14