Companion Approach for cross-sectoral Collaboration in health risks management in SEA

Reading and manipulating spatial data with R

Dr. Vladimir Grosbois
vladimir.grosbois@cirad.fr

CIRAD
UR AGIRs

This project is funded by the European Union

This projectis Implemented by cirad

Why using spatial statistics?

- Studying spatial processes (analysing movements, spatial diffusion of a disease, of a polutant)

- Studies in which we focus on the spatial distribution of statistical units (spatial distribution of the individuals of a species / spatial distribution of the cases of a disease, etc….)

- Accounting for spatial dependency in statistical analyses

Why using spatial statistics?

- Acces to spatialized data is increasingly easy (remote sensing data)

- Maps are excellent analysis and communication tools

CREATING SPATIAL DATA FROM SCRATCH

Object classes for spatialized data in R

- R package sp provides object classes and methods for spatial data
- Package sp provides object classes for spatial-only information
  - Points
  - Grids
  - Lines
  - Rings
  - Polygons

- In addition class extensions exist when attribute information stored in a data frame is associated with each spatial unit

Object classes for spatialized data in R

data type	class	attributes	contains
points	SpatialPoints	No	Spatial
points	SpatialPointsDataFrame	data.frame	SpatialPoints
multipoints	SpatialMultiPoints	No	Spatial
multipoints	SpatialMultiPointsDataFrame	data.frame	SpatialMultiPoints
pixels	SpatialPixels	No	SpatialPoints
pixels	SpatialPixelsDataFrame	data.frame	SpatialPixels SpatialPointsDataFrame
full grid	SpatialGrid	No	SpatialPixels
full grid	SpatialGridDataFrame	data.frame	SpatialGrid
line	Line	No	
lines	Lines	No	Line list
lines	SpatialLines	No	Spatial, Lines list
lines	SpatialLinesDataFrame	data.frame	SpatialLines
polygons	Polygon	No	Line
polygons	Polygons	No	Polygon list
polygons	SpatialPolygons	No	Spatial, Polygons list
polygons	SpatialPolygonsDataFrame	data.frame	SpatialPolygons

The Spatial class

- The Spatial class is the foundation class for spatial data
- It stores metadata included in all the other spatial classes
- It has two components (slots):
  - A boundary box (bbox)
  - A coordinate reference system (proj4string)
- But it doesn’t store any spatial elements or attribute information

Bounding box (bbox)

- The bbox object has two columns (min and max)
- And at least two lines:
  - Eastings (x-axis, longitude)
  - Northings (y-axis, latitude)

```
library(sp)
m <- matrix(c(0, 0, 1, 1), ncol = 2, dimnames = list(c("x-
axis", "y-axis"), c("min", "max")))

m
        min max
x-axis    0   1
y-axis    0   1
```

Coordinate Reference System (CRS)

- A geographic coordinate reference system refers the coordinates of a spatial element to it's position on the globe. It includes
  - Information on the units used to specify geographic coordinates
  - Information on the ellipsoïd model of the shape of the earth
  - Information on the datum (origin point in 3 dimensions)
  - Information on the origin in longitude

- A projected coordinate reference system refers the coordinates of a spatial element to it's position on a 2D map. It includes in addition
  - Information on the geometric model projecting 3D coordinates on a plane
  - Information on measures of length

- The coordinate reference system is stored in a PROJ.4 style
- It is declared with the CRS() function as a character string

The Ellipse: Describes the generalized shape of the Earth. All mapping and coordinate systems begin with this description.

There are lots of ways to do each step, resulting in lots of coordinate reference systems.

The Datum: Defines origin and orientation of the coordinate axes (as well the size/shape of Earth)

A Globe
A 3D ellipse with Lat/Long coordinates

The Projection: Project the globe onto a 2D surface

A Map
A 2D representation of the 3D Earth with Easting/Northing coordinates

The PROJ.4 style for declaring a coordinate reference system

The PROJ.4 style is composed of a series of tags for:

- Ellipsoïd model for the shape of the earth: **`+ellps=`**
- Datum (definition of origin point): **`+datum=`** and **`+towgs84=`**
- Projection system: **`+proj=`**
- Units in which coordinates are expressed: **`+units=`**
- ……

Partial list of tags used in the PROJ.4 style

+a Semimajor radius of the ellipsoid axis
+alpha ? Used with Oblique Mercator and possibly a few others
+axis Axis orientation (new in 4.8.0)
+b Semiminor radius of the ellipsoid axis
+datum Datum name (see `proj -ld`)
+ellps Ellipsoid name (see `proj -le`)
+k Scaling factor (old name)
+k_0 Scaling factor (new name)
+lat_0 Latitude of origin
+lat_1 Latitude of first standard parallel
+lat_2 Latitude of second standard parallel
+lat_ts Latitude of true scale
+lon_0 Central meridian
+lonc ? Longitude used with Oblique Mercator and possibly a few others
+lon_wrap Center longitude to use for wrapping (see below)
+nadgrids Filename of NTv2 grid file to use for datum transforms (see below)
+no_defs Don't use the /usr/share/proj/proj_def.dat defaults file
+over Allow longitude output outside -180 to 180 range, disables wrapping (see below)
+pm Alternate prime meridian (typically a city name, see below)
+proj Projection name (see `proj -l`)
+south Denotes southern hemisphere UTM zone
+to_meter Multiplier to convert map units to 1.0m
+towgs84 3 or 7 term datum transform parameters (see below)
+units meters, US survey feet, etc.
+vto_meter vertical conversion to meters.
+vunits vertical units.
+x_0 False easting
+y_0 False northing
+zone UTM zone

Examples of PROJ.4 strings

The coordinate reference system is declared using the CRS() function with a PROJ.4 style character string as an argument. Examples:

- `Unknowncrs<-CRS(as.character(NA))`

No, or unknown projection system. The coordinates describe positions on a simple x-axis y-axis system

- `Geocrs<-CRS("+proj=longlat +datum=WGS84 +ellps=WGS84")`

Note that whenever proj=longlat, we have a geographic coordinate reference system. The coordinates describe positions on a globe and not on a map.

- `Mapcrs<-CRS("+proj=utm +zone=48 +datum=WGS84 +units=m +ellps=WGS84")`

Here the coordinates describe positions on a map because proj=utm tells that the projection system used is the Universal Transverse Mercator system

Create a Spatial object from a bbox and a proj4string with the function Spatial()

```
s<-Spatial(bbox=m,proj4string=UnknownCRS)
s

An object of class "Spatial"
Slot "bbox":
       min max
x-axis  0  1
y-axis  0  1

Slot "proj4string":
CRS arguments: NA
```

Note that this object does not contain any coordinates of spatial data. It includes only a boundaries box and a coordinate reference system .

Create a SpatialPoints class object from a bbox and a proj4string with the function SpatialPoints()

```
x=c(0.2,0.3,0.6,0.8,0.9)
y=c(0.5,0.2,0.8,0.7,0.6)
p<-cbind(x,y)
spp<-SpatialPoints(p,proj4string=UnknownCRS,bbox=m)
summary(spp)
```

```
Object of class SpatialPoints
Coordinates:
       min max
x-axis   0   1
y-axis   0   1
Is projected: NA
proj4string : [NA]
Number of points: 5
```

Create a SpatialPoints object from a bbox and a proj4string with the function SpatialPoints()

```
x=c(0.2,0.3,0.6,0.8,0.9)
y=c(0.5,0.2,0.8,0.7,0.6)
p<-cbind(x,y)
spp<-SpatialPoints(p)
summary(spp)
```

```
Object of class SpatialPoints
Coordinates:
  min max
x 0.2 0.9
y 0.2 0.8
Is projected: NA
proj4string : [NA]
Number of points: 5
```

Create a SpatialPointsDataFrame object from a data frame and a SpatialPoints object with the function SpatialPointsDataFrame()

- A SpatialPointsDataFrame contains in addition to points coordinates, the values of one or several attributes associated with each point.

- The attribute values can be specified in a data frame

```
att<-data.frame(att1=c(56,12,23,7,32),att2=c(0.2,0.5,2,-1,-0.5))
```

- The SpatialPointsDataFrame object can then be created using the SpatialPointsDataFrame function with
  - a SpatialPoint object containing the coordinates as the first argument
  - and the data frame containing the attribute values as a second argument

```
sppdf<-SpatialPointsDataFrame(spp,att)
```

Create a SpatialPointsDataFrame object with the function SpatialPointsDataFrame()

```
sppdf
```

```
  coordinates att1 att2
1  (0.2, 0.5)   56  0.2
2  (0.3, 0.2)   12  0.5
3  (0.6, 0.8)   23  2.0
4  (0.8, 0.7)    7 -1.0
5  (0.9, 0.6)   32 -0.5
```

Create a SpatialPointsDataFrame object with the function SpatialPointsDataFrame()

```
summary(sppdf)
```

```
Object of class SpatialPointsDataFrame
Coordinates:
  min max
x 0.2 0.9
y 0.2 0.8
Is projected: NA
proj4string : [NA]
Number of points: 5
Data attributes:
      att1          att2
 Min.   : 7   Min.   :-1.00
 1st Qu.:12   1st Qu.:-0.50
 Median :23   Median : 0.20
 Mean   :26   Mean   : 0.24
 3rd Qu.:32   3rd Qu.: 0.50
```

Functions to select elements from a SpatialPointsDataFrame

```
coordinates(sppdf)
```

selects the coordinates only

```
      x   y
[1,] 0.2 0.5
[2,] 0.3 0.2
[3,] 0.6 0.8
[4,] 0.8 0.7
[5,] 0.9 0.6
```

```
sppdf[1]
```

selects the first attribute along with the coordinates

```
  coordinates  att1
1  (0.2, 0.5)   56
2  (0.3, 0.2)   12
3  (0.6, 0.8)   23
4  (0.8, 0.7)    7
5  (0.9, 0.6)   32
```

```
sppdf[1:2,"att2"]
```

selects the two first data points with only the values of attribute 2

```
  1  (0.2, 0.5)  0.2
  2  (0.3, 0.2)  0.5
```

Creating other types of spatial objects

- In the sp library, there are functions allowing to create from scratch SpatialLines, SpatialPolygons, SpatialGrid and SpatialPixel objects by directly entering coordinate values.

- In addition functions allow to associate attribute values to the elements of such objects.

- The resulting objects are of class SpatialLinesDataFrame, SpatialPolygonsDataFrame, SpatialGridDataFrame and SpatialPixelDataFrame.

The methods (functions) for Spatial classes

`dimensions(x)` returns the number of spatial dimensions

`spTransform(x,)` converts a spatial object from one coordinate reference system to another coordinate reference system

`bbox(x)` returns a matrix with the coordinates bounding box

`coordinates(x)` returns a matrix with the spatial coordinates

`spplot(x)` plots attributes in combination with the spatial information

READING SPATIAL DATA FROM EXTERNAL FILES

Create a SpatialPointsDataFrame from a data frame including point coordinates

```
setwd("D:/Mes donnees/Cours stats VG/comacross training/spatial data")
thai<-read.table("data_thailand.csv",sep="\t",header=T)
names(thai)
```

```
 [1] "NO_SUBDIST" "SUBDISTRIC" "infected"   "NBCHICK"    "NBDUCK"
 [6] "NBFGD"      "AREAKM2"    "CENTROIDX"  "CENTROIDY"  "ncropmean"
[11] "POPDENS"    "RIVERDENS"  "ROADDENS"
```

The thai data frame contains point coordinates (i.e. subdistrict centroïds)

Create a SpatialPointsDataFrame from a data frame including point coordinates

The thai data frame can be converted into a SpatialPointsDataFrame using the coordinates function to specify which variables are coordinates

```
coordinates(thai)<-c("CENTROIDX","CENTROIDY")
summary(thai)
```

```
Object of class SpatialPointsDataFrame
Coordinates:
             min     max
CENTROIDX 355002 1204736
CENTROIDY 634340 2258559
Is projected: NA
proj4string : [NA]
Number of points: 7366
```

Spatial objects in R

```
setwd("D:/Mes donnees/Cours stats VG/interrisk/Advanced Stat/scripts")
```

- Load the library for importing spatialized data analysis

```
library(rgdal)
```

- Import the vector data (shape files) with the readOGR function

```
villages<-readOGR("AI Cambodia data", "Donnees")
routes<-readOGR("AI Cambodia data","routes_principales")
frontieres<-readOGR("AI Cambodia data","Frontieres_des_provinces")
zone<-readOGR("AI Cambodia data","Zone")
```

Folder including the spatial data

Name of the layer

Importing vector data

- Set the working directory to the directory including the data folder

```
setwd("D:/Mes donnees/Cours stats VG/interrisk/Advanced Stat/scripts")
```

- Load the library required to import GIS data

```
library(rgdal)
```

- Import vector data (shape files) with the readOGR() function

```
villages<-readOGR("AI Cambodia data", "Donnees")
routes<-readOGR("AI Cambodia data","routes_principales")
frontieres<-readOGR("AI Cambodia data","Frontieres_des_provinces")
zone<-readOGR("AI Cambodia data","Zone")
```

Folder including the spatial data

Name of the layer

Content of vector type objects

- Information on the spatialized data objects created

```
summary(villages)
Object of class SpatialPointsDataFrame
Coordinates:
              min     max
coords.x1  213079  654500
coords.x2 1152800 1567023
Is projected: TRUE
proj4string :
[+proj=utm +zone=48 +datum=WGS84 +units=m +no_defs +ellps=WGS84
+towgs84=0,0,0]
Number of points: 4275
```

Class of the object

Boundary box

Information on the projection

Number of elements (here points)

Content of vector type objects

- Information on the attributes

```
Data attributes:
     ID_INF             NOM_INF          X_INF            Y_INF            INFECTE
 Min.   : 1020101   Thmei    :  46   Min.   :213079   Min.   :1152800   Min.   :0.00000
 1st Qu.: 3030902   Samraong:  24   1st Qu.:443150   1st Qu.:1229550   1st Qu.:0.00000
 Median : 3160801   Kandaal :  21   Median :504500   Median :1302854   Median :0.00000
 Mean   : 8735638   Doung   :  14   Mean   :475678   Mean   :1310845   Mean   :0.09965
 3rd Qu.:14090402   Pou     :  14   3rd Qu.:551850   3rd Qu.:1349139   3rd Qu.:0.00000
 Max.   :21101403   Chambak :  12   Max.   :654500   Max.   :1567023   Max.   :1.00000
                    (Other) :4144
   CULTIRMOY        CULTNOMOY         INONDMAX            POP
 Min.   :0.0000   Min.   :0.0000   Min.   :-9999.0   Min.   :    0
 1st Qu.:0.0333   1st Qu.:0.3618   1st Qu.:   13.0   1st Qu.:   31
 Median :0.1110   Median :0.5361   Median :   18.0   Median :   70
 Mean   :0.1669   Mean   :0.5554   Mean   : -294.2   Mean   :  458
 3rd Qu.:0.2375   3rd Qu.:0.7503   3rd Qu.:   24.0   3rd Qu.:  224
 Max.   :0.9816   Max.   :1.0000   Max.   :   42.0   Max.   :15095
```

Content of vector type objects

- Information on the spatialized data objects created

```
summary(roads)
Object of class SpatialLinesDataFrame
Coordinates:
      min     max
x  210865  782365
y 1145318 1595842
Is projected: TRUE
proj4string :
[+proj=utm +zone=48 +datum=WGS84 +units=m +no_defs +ellps=WGS84
+towgs84=0,0,0]
```

Class of the object is not points but lines

Content of vector type objects

- Information on the attributes

```
Data attributes:
     FNODE_          TNODE_          LPOLY_            RPOLY_
 Min.   :    1   Min.   :    3   Min.   :-1.0000   Min.   :-1.0000
 1st Qu.:11172   1st Qu.:11176   1st Qu.:-1.0000   1st Qu.:-1.0000
 Median :21923   Median :21950   Median :-1.0000   Median :-1.0000
 Mean   :20320   Mean   :20328   Mean   :-0.9999   Mean   :-0.9999
 3rd Qu.:29499   3rd Qu.:29507   3rd Qu.:-1.0000   3rd Qu.:-1.0000
 Max.   :36558   Max.   :36555   Max.   : 0.0000   Max.   : 0.0000
     LENGTH             RD_LIN_        RD_LIN_ID         CODE          Poids
 Min.   :    0.006   Min.   :    1   Min.   :   0   Min.   :1.00   Min.   :4.000
 1st Qu.:  400.149   1st Qu.:13021   1st Qu.: 342   1st Qu.:3.00   1st Qu.:4.000
 Median :  916.362   Median :26319   Median :1214   Median :5.00   Median :4.000
 Mean   : 1458.262   Mean   :24657   Mean   :1519   Mean   :3.99   Mean   :4.721
 3rd Qu.: 1839.536   3rd Qu.:36092   3rd Qu.:2459   3rd Qu.:5.00   3rd Qu.:6.000
 Max.   :27481.910   Max.   :45502   Max.   :5627   Max.   :5.00   Max.   :6.000
```

Content of vector type objects

- Information on the spatialized data objects created

```
summary(zone)
Object of class SpatialPolygonsDataFrame
Coordinates:
        min       max
x  211823.1  660177.1
y 1149860.6 1575975.1
Is projected: TRUE
proj4string :
[+proj=utm +zone=48 +datum=WGS84 +units=m +no_defs +ellps=WGS84
+towgs84=0,0,0]
Data attributes:
 Zone
 NO:1
 SE:1
 SO:1
```

Class of the object is not points but polygons

Importing raster data

- Import the raster data with the readGDAL function

```
population<-readGDAL("AI Cambodia data/population")
```

For raster data generated with ARCINFO, name of the folder containing the raster files

```
summary(population)
Object of class SpatialGridDataFrame
Coordinates:
        min       max
x 102.32419 107.64919
y  10.34073  14.69906
Is projected: FALSE
proj4string :
[+proj=longlat +ellps=WGS84 +towgs84=0,0,0,0,0,0,0 +no_defs]
Grid attributes:
  cellcentre.offset    cellsize cells.dim
x         102.32836 0.008333333       639
y          10.34489 0.008333333       523
```

The class of the object is spatial grid

No projection because proj=longlat

smallest coordinate for each dimension

cell size for each dimension

number of cells in each dimension

Table of the values of the raster variable

```
Data attributes:
     band1
 Min.   :    0.00
 1st Qu.:    1.00
 Median :    3.00
 Mean   :   62.25
 3rd Qu.:   19.00
 Max.   :56223.00
 NA's   :119532
```

Checking the structure of the object with str()

str(villages)

Allows you to see how to extract components (with @ and $)

```
Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
  ..@ data       :'data.frame':         4275 obs. of  9 variables:
  .. ..$ ID_INF   : num [1:4275] 1020101 1020103 1020117 1020119 1020205 ...
  .. ..$ NOM_INF  : Factor w/ 3228 levels "Aa Kreach","Aa Roung",..: 1497 211 1436 1137 192 191 31
1130 1765 2498 ...
  .. ..$ X_INF    : num [1:4275] 287800 285900 288292 291700 282900 ...
  .. ..$ Y_INF    : num [1:4275] 1494700 1494300 1489214 1489700 1495700 ...
  .. ..$ INFECTE  : num [1:4275] 0 0 1 1 0 0 0 0 1 1 ...
  .. ..$ CULTIRMOY: num [1:4275] 0.206 0.171 0.264 0.292 0.117 ...
  .. ..$ CULTNOMOY: num [1:4275] 0.418 0.479 0.329 0.244 0.357 ...
  .. ..$ INONDMAX : num [1:4275] 19 13 13 13 14 14 9 19 19 18 ...
  .. ..$ POP      : int [1:4275] 383 147 16 15 15 15 16 15 15 15 ...
  ..@ coords.nrs : num(0)
  ..@ coords     : num [1:4275, 1:2] 287800 285900 288292 291700 282900 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:2] "coords.x1" "coords.x2"
  ..@ bbox       : num [1:2, 1:2] 213079 1152800 654500 1567023
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "coords.x1" "coords.x2"
  .. .. ..$ : chr [1:2] "min" "max"
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
  .. .. ..@ projargs: chr "+proj=utm +zone=48 +datum=WGS84 +units=m +no_defs +ellps=WGS84
+towgs84=0,0,0"
```

Checking projections

- In order to represent several layers on a map it is important to check that the projections are the same

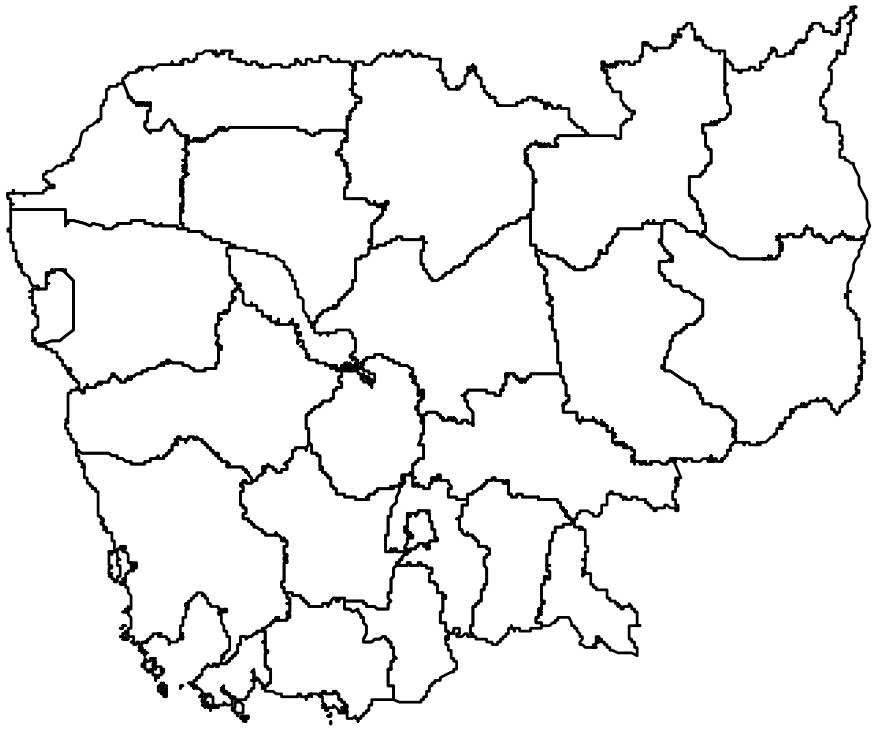
```
villages@proj4string
roads@proj4string
frontiers@proj4string
zone@proj4string
population@proj4string
```

```
proj=utm +zone=48 +datum=WGS84 +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0
proj=utm +zone=48 +datum=WGS84 +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0
proj=utm +zone=48 +datum=WGS84 +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0
proj=utm +zone=48 +datum=WGS84 +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0
proj=longlat +ellps=WGS84 +towgs84=0,0,0,0,0,0,0 +no_defs
```

REPRESENTING SPATIAL DATA WITH R

Ploting vector data with the plot function

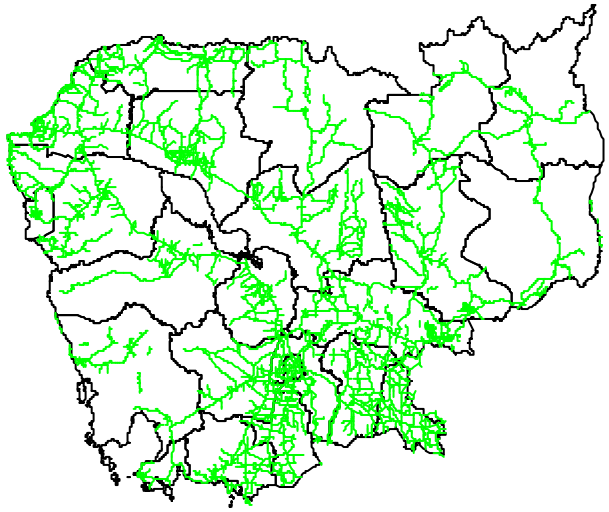
- The plot function is used to produce the map for points, polygons



Representing vector data with the plot function

- The add=TRUE option can be used to overlay several vector data

```
plot(roads,col="green",add=T)
```

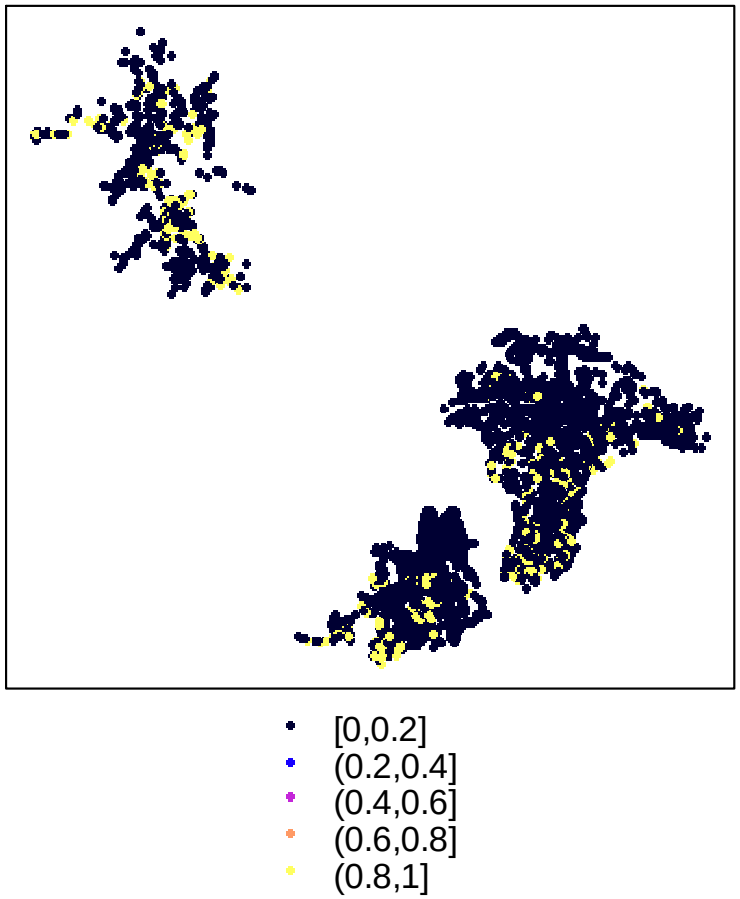


Be carefull that the CRS of the represented objects must match.
Otherwise you can use the spTransform function from library sp to change the CRS of the objects

Representing point objects with attribute values using the spplot function

```
spplot(villages,"INFECTE", cex=0.5)
```

Argument for symbol size

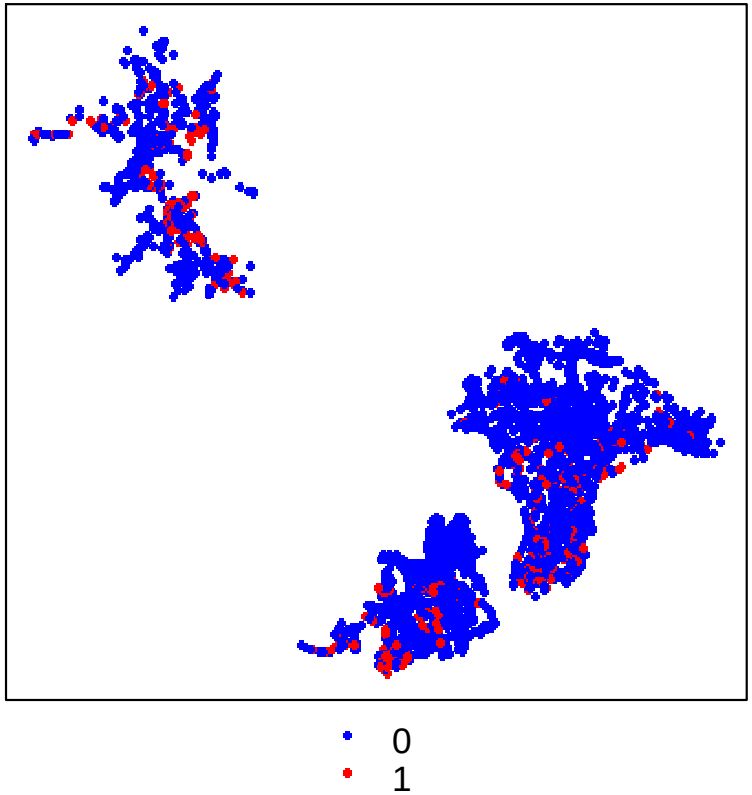


Not a very good representation because the attribute INFECTE is binary and not continuous

Representing points objects with attribute values using the spplot function

We change the INFECTE attribute from numeric to factor

```
villages$INFFAC<-as.factor(villages$INFECTE)
spplot(villages,"INFFAC",col.regions=c("blue","red"),cex=0.5)
```

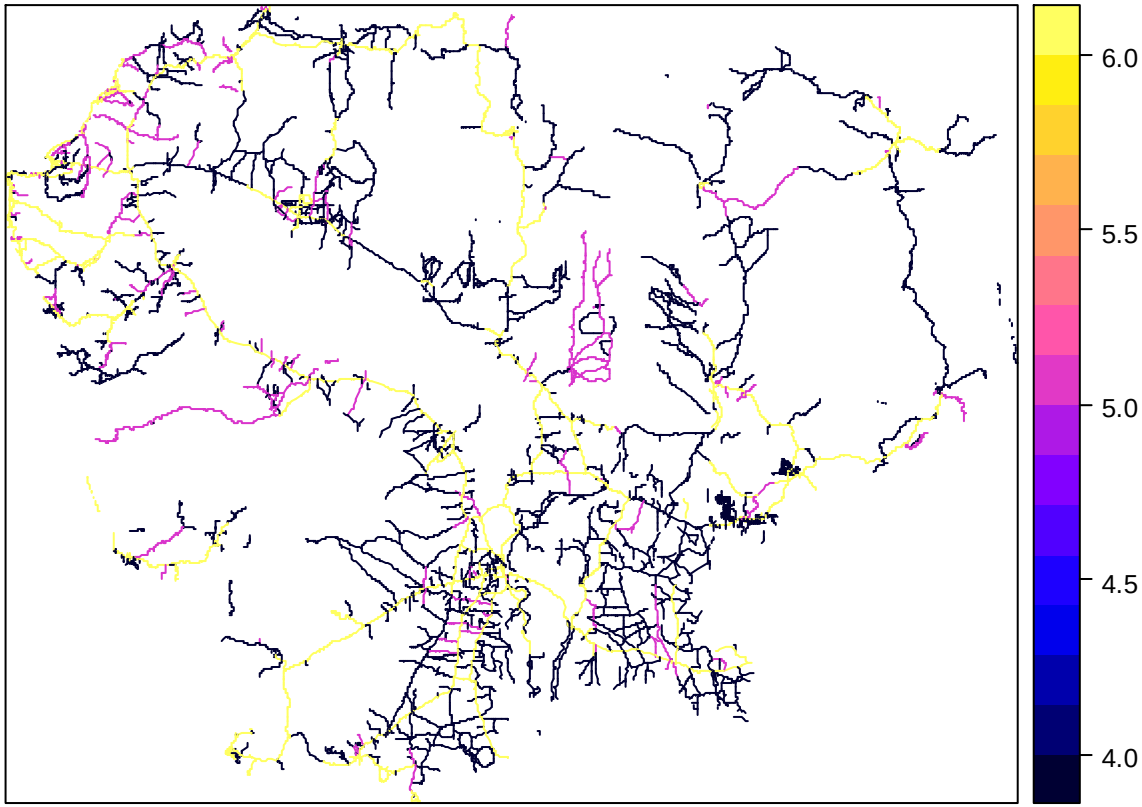


Argument for symbol colour

Note that the spplot function will also allow representing polygons and lines along with their attribute values

Representing lines objects with attribute values using the spplot function

```
spplot(roads,"Poids")
```

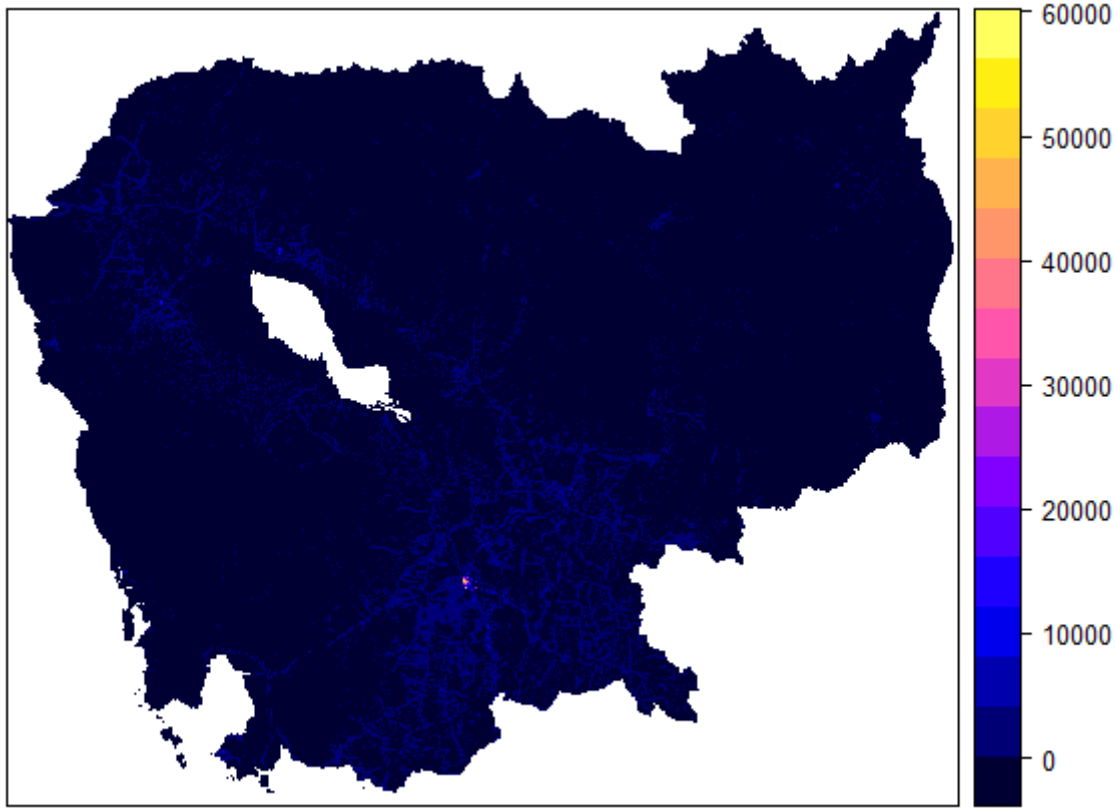


Representing object of type grid or pixel with the spplot function

- For objects of type SpatialGridDataFrame or SpatialPixelDataFrame, the plot function does not work, you have to use the spplot function

```
spplot(population)
```

Not a nice representation because the population data is too contrasted with very low population density except in very local areas where density can be very high

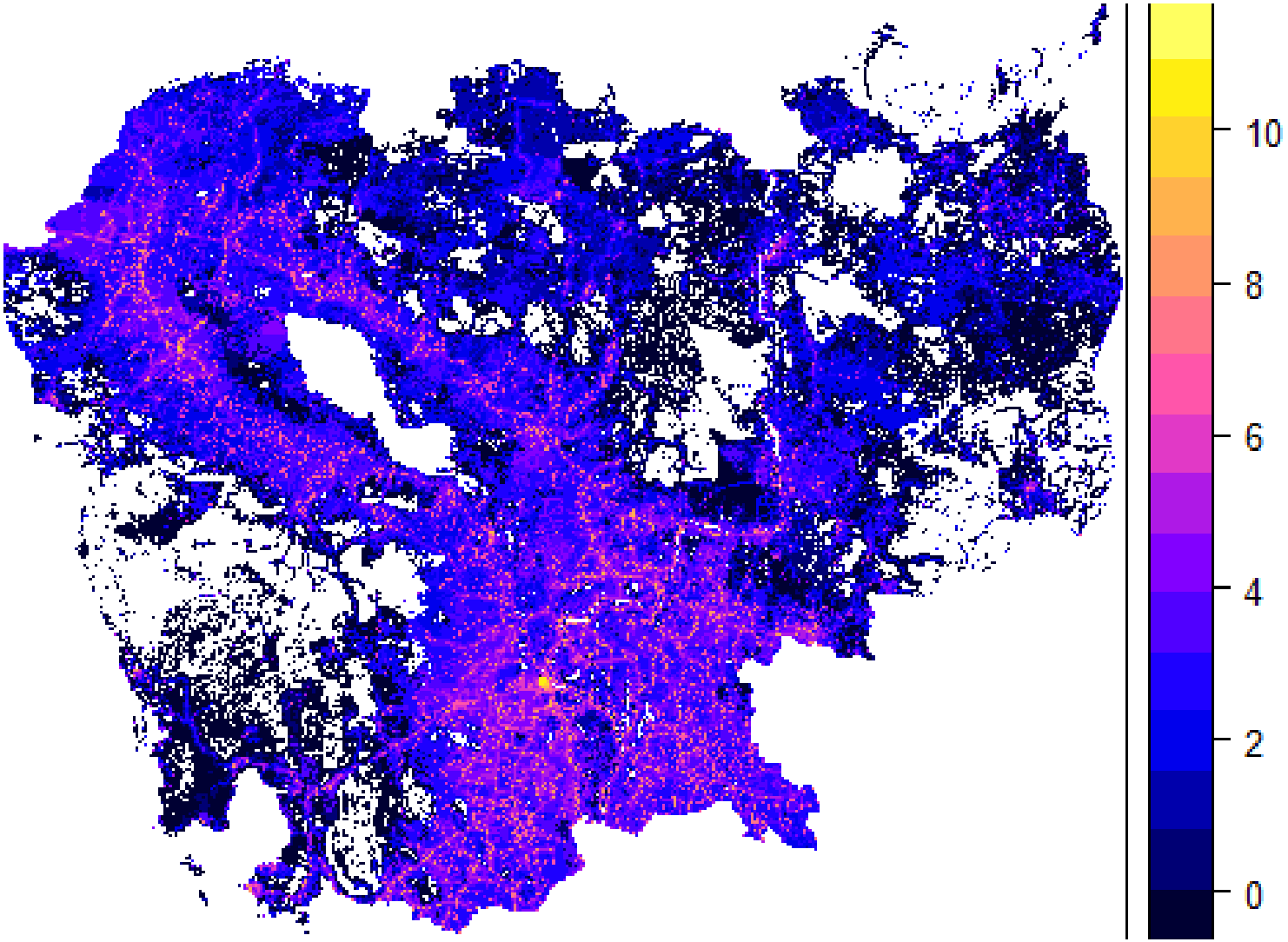


Representing object of type grid or pixel with the spplot function

- We need to transform the population data because the variance is too large for the representation on a map

```
logpopulation<-population
logpopulation@data<-log(logpopulation@data)
spplot(logpopulation)
```

Representing object of type grid or pixel with the spplot function



Representing spatial objects with attribute values with a layout defined with other spatial objects

- The maptools library allows to add information on other spatial objects on a spplot for a spatial object with attribute

- For doing you have to define a layout with the spatial objects to be added to the spplot

- The layout is defined with lists including spatial objects, functions, and options

Representing spatial objects with attribute values with a layout defined with other spatial objects

- The function depends on the class of the spatial object to be added to the spplot

sp layout function	Object class	Useful arguments[a]
`sp.points`	`SpatialPoints`	`pch`, `cex`, `col`
`sp.polygons`	`SpatialPolygons`	`lty`, `lwd`, `col`
`sp.lines`	`SpatialLines`	`lty`, `lwd`, `col`
`sp.text`	text	(see `panel.text`)

[a]For help, see `?par`

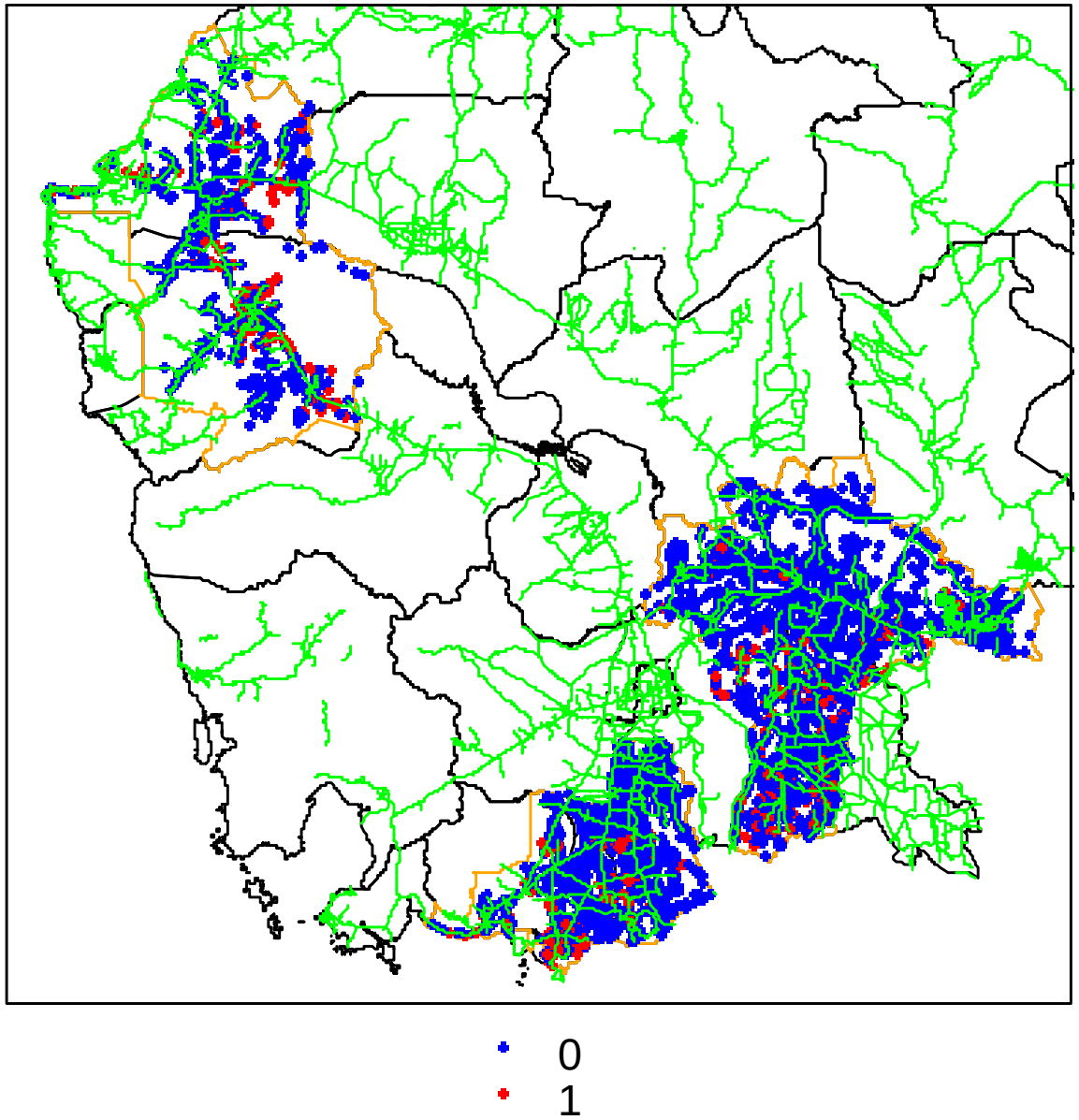
Representing object of type grid or pixel with the spplot function

Definition of the layout

```
fr<-list("sp.polygons", frontiers)
ro<-list("sp.lines", roads,col="green")
zo<-list("sp.polygons", zone, col="orange")
lo<-list(fr,ro,zo)
```

Producing the spplot

```
spplot(villages,"INFFAC",col.regions=c("blue","red"),cex=0.5,sp.layout=
lo)
```


0
1

Representing spatial grid objects with a layout defined with other spatial objects

- This method can also be used to represent to overlay geometry of spatial points, lines or polygons on a spatial grid object
- Here we want to represent the population density and the frontiers

Definition of the layout

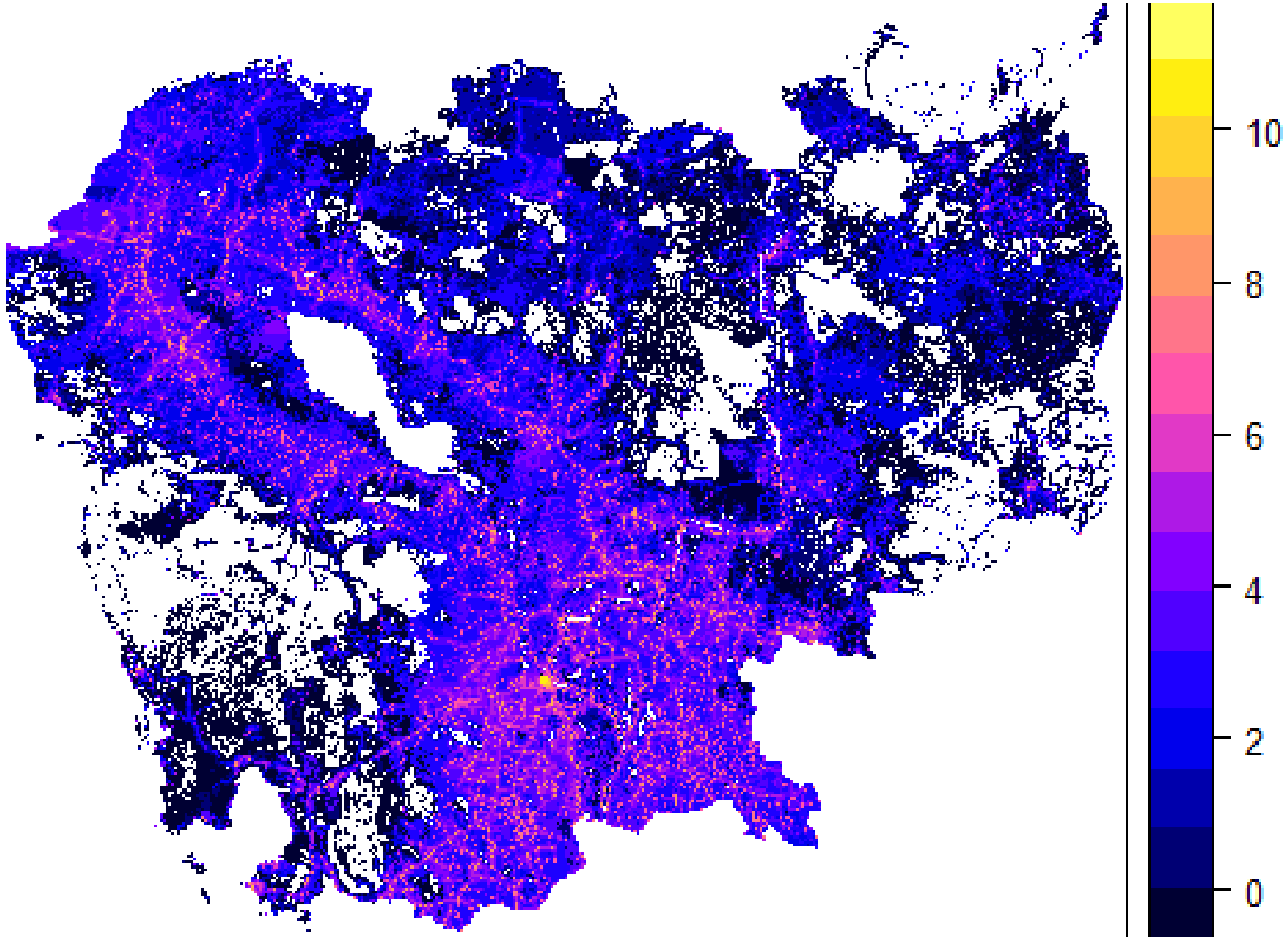
```
lo1<-list("sp.polygons", frontier,lwd=4,first=F)
```

This layer is not drawn first

Producing the spplot

```
spplot(logpopulation,sp.layout=lo1)
```

No frontiers because the CRS of the two layers are different



Representing spatial grid objects with a layout defined with other spatial objects

- Check the CRS

```
proj4string(logpopulation)
"+proj=longlat +ellps=WGS84 +towgs84=0,0,0,0,0,0,0 +no_defs"
proj4string(frontiers)
"+proj=utm +zone=48 +datum=WGS84 +units=m +no_defs +ellps=WGS84
+towgs84=0,0,0"
```

- Change the CRS of frontiers

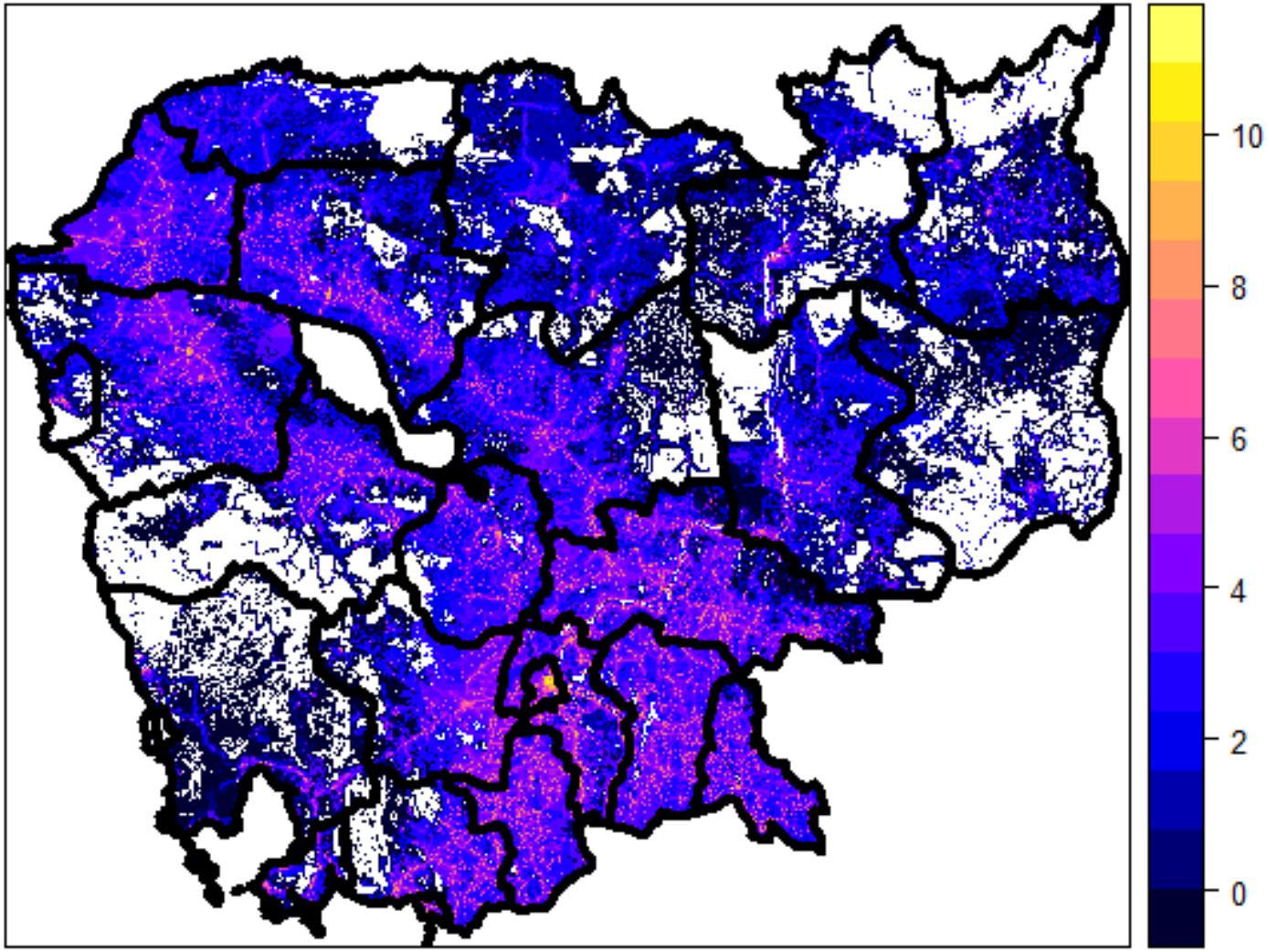
```
projcomp<-CRS("+proj=longlat +ellps=WGS84 +towgs84=0,0,0,0,0,0,0
+no_defs")
frontierscomp<-spTransform(frontiers, projcomp)
```

- Define the layout

```
lo2<-list("sp.polygons", frontierscomp,lwd=4,first=F)
```

- Produce the plot

```
spplot(logpopulation,sp.layout=lo2)
```


10
8
6
4
2
0

CONVERTING SPATIAL DATA WITH R

Converting SpatialGridDataFrames into SpatialPolygonsDataFrames with function aggegate()

- We want to get for each area defined by frontiers the median of the log of population density

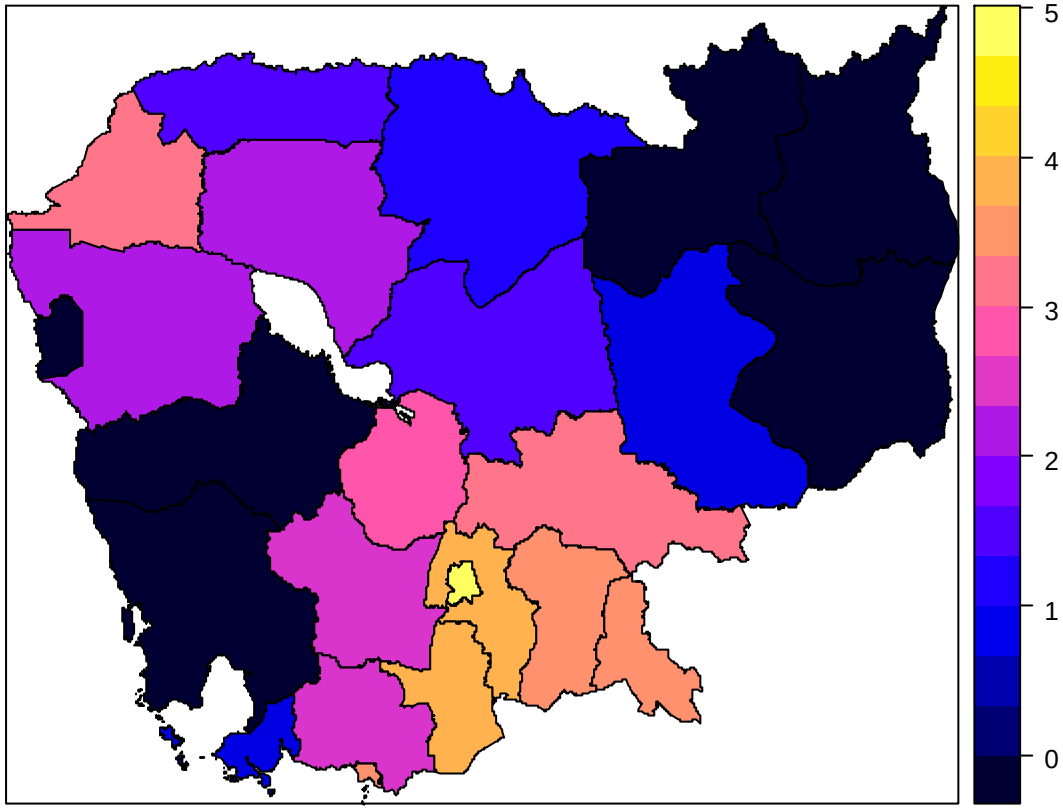
```
logpoparea<-aggregate(logpopulation, by = frontierscomp,median)
summary(logpoparea)
Object of class SpatialPolygonsDataFrame
Coordinates:
        min      max
x 102.33759 107.6315
y  10.35076  14.6873
Is projected: FALSE
proj4string :
[+proj=longlat +ellps=WGS84 +towgs84=0,0,0,0,0,0,0 +no_defs]
Data attributes:
     band1
 Min.   :0.0000
 1st Qu.:0.5199
 Median :2.2499
 Mean   :1.9518
 3rd Qu.:3.3021
 Max.   :4.6821
```

Note that we have to use the frontiers object with CRS compatible with logpopulation

Converting SpatialGridDataFrames into SpatialPolygonsDataFrames with function aggegate()

- Producing the plot

```
spplot(logpoparea,"band1")
```



Converting SpatialPolygonsDataFrames into SpatialPointsDataFrames with function gcentroid()

- We want to transform the area defined by frontiers into points that are at the centroids of the areas with the log of population density as an attribute

```
library(rgeos)
areapoint<-gCentroid(logpoparea, byid=TRUE)
summary(areapoint)
Object of class SpatialPoints
Coordinates:
        min       max
x 102.63451 107.09080
y  10.51871  14.16357
Is projected: FALSE
proj4string :
[+proj=longlat +ellps=WGS84 +towgs84=0,0,0,0,0,0,0 +no_defs]
Number of points: 24
```

We get a SpatialPoints object without attribute

Converting SpatialPolygonsDataFrames into SpatialPointsDataFrames with function gcentroid()

- We need to associate median log(pop density) to the SpatialPoints object to create a SpatialPointsDataFrame

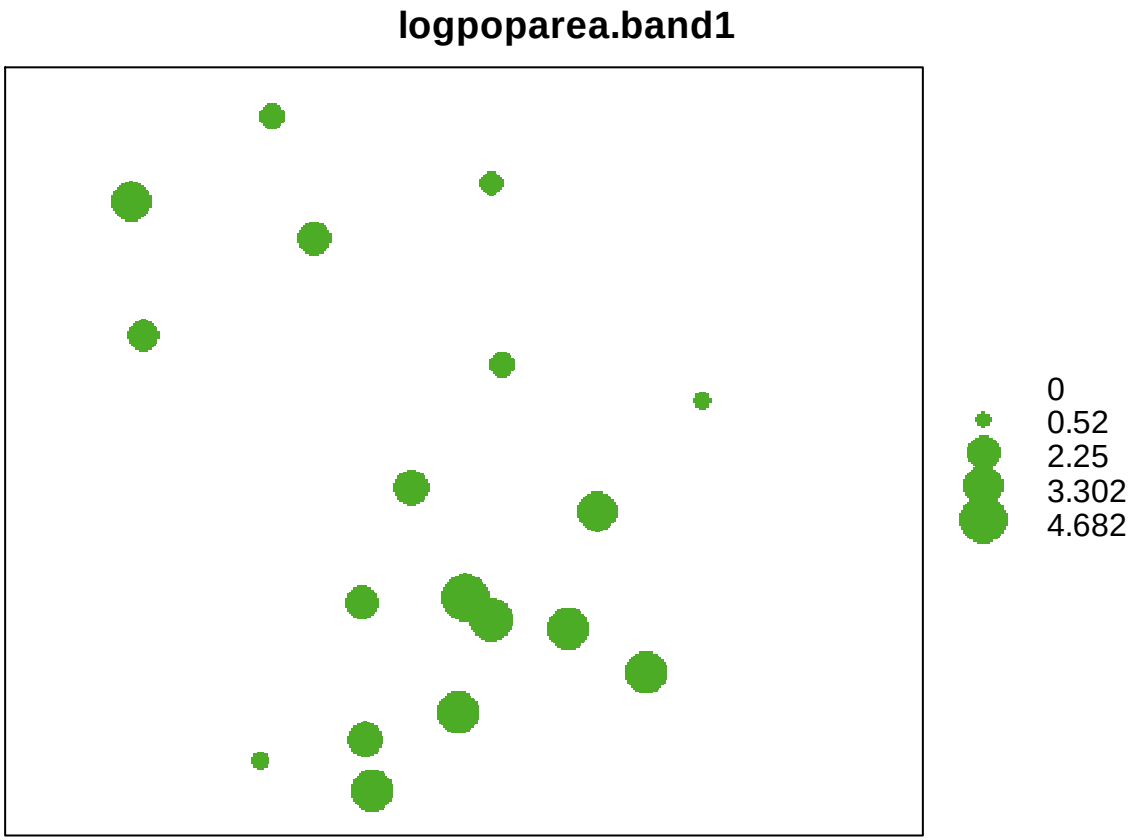
```
logpoppoint<-
SpatialPointsDataFrame(areapoint,data.frame(logpoparea$band1)
```

Converting SpatialPolygonsDataFrames into SpatialPointsDataFrames with function gcentroid()

```
summary(logpoppoint)
Object of class SpatialPointsDataFrame
Coordinates:
        min       max
x 102.63451 107.09080
y  10.51871  14.16357
Is projected: FALSE
proj4string :
[+proj=longlat +ellps=WGS84 +towgs84=0,0,0,0,0,0,0 +no_defs]
Number of points: 24
Data attributes:
 logpoparea.band1
 Min.   :0.0000
 1st Qu.:0.5199
 Median :2.2499
 Mean   :1.9518
 3rd Qu.:3.3021
 Max.   :4.6821
```

Converting SpatialPolygonsDataFrames into SpatialPointsDataFrames with function gcentroid()

```
bubble(logpoppoint,"logpoparea.band1")
```



The bubble function is an alternartive to spplot where attribute value is represented through symbol size