Topic 2: Spatial Objects in R

This topic gives overview of spatial data representation in R. In particular, we consider

- Some examples of **Spatial Objects in R**.
- R class **Spatial**.
- Subclasses **SpatialPoints** and **Spatial*DataFrame**.
- Several **Methods** for spatial subclasses.
- Example of **Meuse data**.

Spatial Objects in R.

One of standard ways to use spatial data in R is provided by the **sp package**.

This package uses new-style classes. The central advantage of new-style classes is that they explicitly define their components, called slots. Slots contain data, their names, formats, and are well-structured. This simplifies the writing, maintenance, and use of the classes, because their format is known from the definition.

New classes can be easily built by using hierarchical nested structures. One just need to add new slots to specify a subclass of spatial objects within a more general class.

The foundation class is the **Spatial** class, with just two slots:

- The first is a **bounding box,** a matrix of numerical coordinates with column names C('MIN', 'MAX'), and at least two rows, with the first row eastings (x-axis) and the second northings (y-axis). Most often the bounding box is generated automatically from the data in subclasses of Spatial.

- The second is a **CRS class object** defining the coordinate reference system, and may be set to 'missing', represented by NA in R, by CRS(AS.CHARACTER(NA)), its default value.

Operations on SPATIAL* objects should update or copy these values to the new SPATIAL* objects being created.

One can use **getClass** to return the complete definition of a class, including its slot names and the types of their contents:

```
> library(sp)
> getClass("Spatial")
      Class "Spatial" [package "sp"]
      Slots:
      Name:          bbox proj4string
      Class:       matrix         CRS
      Known Subclasses:
      Class "SpatialPoints", directly
      Class "SpatialMultiPoints", directly
      Class "SpatialGrid", directly
      Class "SpatialLines", directly
      Class "SpatialPolygons", directly
      ...
```

As we see, GETCLASS also returns known subclasses, showing the classes that include the SPATIAL class in their definitions.

The coordinate reference system (CRS) class:

```
> getClass("CRS")

Slots:
Name: projargs
Class: character
```

The class has a character string as its only slot value, which may be a missing value. If it is not missing, it should be a PROJ.4-format string describing the projection. For geographical coordinates, the simplest such string is "+proj=longlat", using "longlat".

Build a simple Spatial object from a bounding box matrix, and a missing coordinate reference system can be done as:

```
> m <- matrix(c(0, 0, 1, 1), ncol = 2, dimnames = list(NULL,
+ c("min", "max")))
> crs <- CRS(projargs = as.character(NA))
> crs

> S <- Spatial(bbox = m, proj4string = crs)
> S
    An object of class "Spatial"
    Slot "bbox":
    min max
    [1,]   0   1
    [2,]   0   1
    Slot "proj4string":
    CRS arguments: NA
```

Eastings always go before northings in **sp** classes. System checks the coordinate range.

For example, try the following incorrect range:

```
> Spatial(matrix(c(20, 45, 360, 15), ncol=2, dimnames=
    + list(NULL, c("min",  "max"))), proj4string=
    + CRS("+proj=longlat +datum=WGS84"))
```

and you will get the message "invalid bbox: max < min".

SpatialPoints

The **SpatialPoints** class is the first subclass of SPATIAL.

A two-dimensional point can be described by a pair of numbers (x, y), defined over a known region.

Using the standard READ.TABLE function, we read in a data file with the positions of CRAN mirrors across the world. We extract the two columns with the longitude and latitude values into a matrix, and use STR to view a digest:

```
> CRAN_df <- read.table("CRAN051001a.txt", header = TRUE)
> CRAN_mat <- cbind(CRAN_df$long, CRAN_df$lat)
> row.names(CRAN_mat) <- 1:nrow(CRAN_mat)
> str(CRAN_mat)
```

The SpatialPoints class extends the Spatial class by adding a **coords** slot, into which a matrix of point coordinates can be inserted.

```
> getClass("SpatialPoints")

> llCRS <- CRS("+proj=longlat +ellps=WGS84")

> CRAN_sp <- SpatialPoints(CRAN_mat, proj4string = llCRS)
> summary(CRAN_sp)
    Object of class SpatialPoints
    Coordinates:
    min       max
    coords.x1 -122.95000 153.0333
    coords.x2  -37.81667  57.0500
    Is projected: FALSE
    proj4string : [+proj=longlat +ellps=WGS84 +no_defs]
    Number of points: 54
```

Methods

Methods are available to access the values of the slots of Spatial objects.

The **bbox** method returns the bounding box of the object:

```
> bbox(CRAN_sp)
```

The first row reports the west-east range and the second the south north direction.

proj4string method reports the projection string contained as a CRS object in the proj4string slot of the object, but it also has an assignment form, allowing the user to alter the current value, which can also be a CRS object containing a character NA value:

```
> proj4string(CRAN_sp)
> proj4string(CRAN_sp) <- CRS(as.character(NA))
> proj4string(CRAN_sp)
> proj4string(CRAN_sp) <- llCRS
```

Extracting the coordinates from a SpatialPoints.

The indices can be used to choose subsets, for example CRAN mirrors located in Brazil in 2005:

```
> brazil <- which(CRAN_df$loc == "Brazil")

> brazil

> coordinates(CRAN_sp)[brazil, ]
      coords.x1 coords.x2
      4 -49.26667 -25.41667
      5 -42.86667 -20.75000
      6 -43.20000 -22.90000
      7 -47.63333 -22.71667
      8 -46.63333 -23.53333
```

In addition, a SpatialPoints object can also be accessed by index, using the "[" operator, here on the coordinate values treated as an entity. The object returned is of the same class, and retains the projection information, but has a new bounding box:

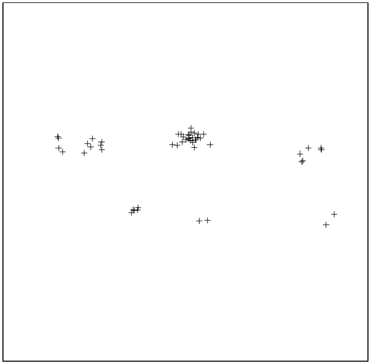
```
> summary(CRAN_sp[brazil, ])
```

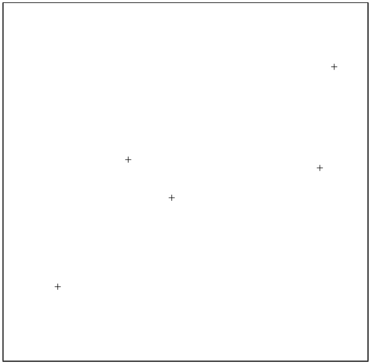
summary reports the number of spatial entities, the projection information, and the bounding box, and print gives a view of the data in the object:

```
> print(CRAN_sp[brazil,])
```

We can use plot to produce an image with our data locations:

```
> plot(CRAN_sp)
> plot(CRAN_sp[brazil,])
```





Data Frames for Spatial Point Data.

We use numbers in sequence to index the points and the rows of our data frame, because neither the place names nor the countries are unique.

```
> str(row.names(CRAN_df))
```

We would like to associate the correct rows of our data frame object with 'their' point coordinates as it often happens that data are collected from different sources, and the two need to be merged.

The **SpatialPointsDataFrame** class is the container for this kind of spatial point information.

If the matrix of point coordinates has row names and the match.ID argument is set to its default value of TRUE, then the matrix row names are checked against the row names of the data frame.

- If they match, but are not in the same order, the data frame rows are re-ordered to suit the points.
- If they do not match, no SpatialPointsDataFrame is constructed.

Note that the new object takes two indices, the first for the spatial object, the second, if given, for the column.

Giving a single index number, or range of numbers, or column name or names returns a new SpatialPointsDataFrame with the requested columns. Using other extraction operators, especially the $ operator, returns the data frame column referred to:

```
> CRAN_spdf1 <- SpatialPointsDataFrame(CRAN_mat, CRAN_df,
+ proj4string = llCRS, match.ID = TRUE)
> CRAN_spdf1[4, ]
```

Because the SpatialPointsDataFrame class extends SpatialPoints, it also inherits the information contained in the Spatial class object.

The data slot is where the information from the data frame is kept, in a data.frame object.

```
> getClass("SpatialPointsDataFrame")
```

The Spatial*DataFrame classes have been designed to behave as far as possible like data frames.

Plot Spatial* with spplot

Apart from the traditional plot methods provided by R, a method, called **spplot**, provides plotting of spatial data with attributes. The advantage it offers is that many maps can be composed into single (sets of) graphs, easily and efficiently. Other cases in which multiple sub-maps are useful are, for example when different moments of time or different modelling scenarios are used to split the data over sub-plots (panels).

Function SPPLOT plots spatial objects using colour (or grey tone) to denote attribute values. The first argument therefore has to be a spatial object with attributes.

The first argument to SPPLOT is a SPATIAL*DATAFRAME object with points, lines, polygons, or a grid. The second argument tells which attributes (column names or numbers) should be used; if omitted, all attributes are plotted. Further attributes control the plotting: colours, symbols, legend classes, size, axes, and geographical reference items to be added.

We use meuse.grid data. The meuse.grid data frame has 3103 rows and 7 columns; a grid with 40m x 40m spacing that covers the Meuse River Study area. This data frame contains the following columns:

- **x** a numeric vector; x-coordinate
- **y** a numeric vector; y-coordinate
- **dist** distance to the Meuse river; obtained by a spread (spatial distance) GIS operation, from border of river; normalized to $[0, 1]$
- **ffreq** flood frequency; the lower the value, the larger the flood frequency
- **part.a** arbitrary division of the area in two areas, a and b
- **part.b** see part.a
- **soil** soil type.

To load data, run the commands

```
> library(lattice)
> data(meuse.grid)
```

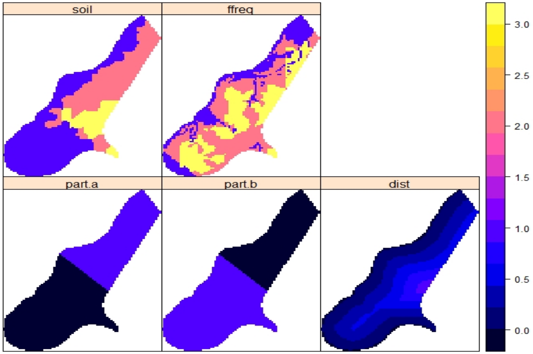
Now let us use **coordinates** to create Spatial*DataFrame

```
> coordinates(meuse.grid) <- ~ x + y

> gridded(meuse.grid) = TRUE
```

Finally we produce an image with all attributes of our data:

```
> spplot(meuse.grid)
```


soil
ffreq
part.a
part.b
dist
3.0
2.5
2.0
1.5
1.0
0.5
0.0

Key R commands

getClass(x)	*gets the definition of a class*
CRS(x)	*returns an object of class CRS*
Spatial(bbox, proj4string)	*returns an object of class Spatial*
SpatialPoints(x)	*creates an object of class SpatialPoints*
bbox(x)	*retrieves spatial bounding box from spatial data*
proj4string(x)	*returns a character vector of projection*
coordinates(x)	*sets/returns spatial coordinates*
SpatialPointsDataFrame(x)	*create an object of class SpatialPointsDataFrames*
gridded (x)	*adds grid topology to an object*
spplot(x)	*returns a lattice plot of spatial attributes*