Topic 1: Spatial Objects in package sf

This topic gives an overview of spatial data representations in the sf package. In particular, we consider

- Spatial Objects in **package sf**.
- Creating **sf objects**.
- Creating **a spatial object from a table** with coordinates.
- **Plotting sf objects**.
- Using the **mapview package** to visualise sf objects.
- **Converting between sf and sp** formats.

Spatial Objects in package sf.

A new standard way to use spatial data in R is provided by the **sf package**. The package can be used for most of SP PACKAGE functionality, but is more intuitive, flexible and provides a standardized way to encode and work with spatial vector data.

The packages stores spatial objects as data frames with a special column that contains information about geometry features. That special column is a list with the same length as the number of rows in the data frame.

Features describe where spatial objects are located on the Earth. There are also additional attributes, which describe other properties of spatial objects.

For example, the geometry of a city can be a point indicating its centre. Its attributes may include its diameter, population, etc.

There are seven basic simple geometric feature types: Point, MultiPoint, LineString, MultiLineString, Polygon, MultiPolygon, GeometryCollection.

To create these spatial sf objects from scratch one can use the following steps:

- Create a geometric feature object (called **sfg**) from a numeric vector, matrix or a list with the coordinates.
- Combine several individual feature objects into a simple feature collection (**sfc** object) using **st_sfc()**. The resulting SFC object has the bounding box and the projection information.
- Add attributes using the **st_sf()** function. The result is a data frame in R with a column that holds the simple feature geometry collection.

Example 1. Let us create two simple feature geometries (each consisting of two points) out of two coordinate matrices:

```
> library(sf)
> p1 <- st_multipoint(matrix(c(0,1/2,0,1/2),ncol=2))
> p2 <- st_multipoint(matrix(c(2/3,1,2/3,1),ncol=2))
```

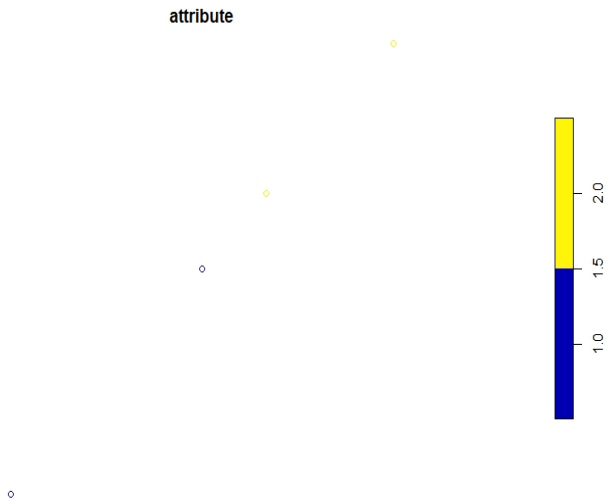
Then, let us combine them into a simple feature collection:

```
> sfc <- st_sfc(p1, p2)
> str(sfc)
sfc_MULTIPOINT of length 2; first list element:
'XY' num [1:2, 1:2] 0 0.5 0 0.5
```

Finally, let us assign attributes 1 and 2 to each of these simple features and plot them with different colours

```
> d_sf <- st_sf(sfc, attribute = 1:2)
> str(d_sf)
Classes sf and 'data.frame': 2 obs. of  2 variables:
$ attribute: int  1 2
$ sfc      :sfc_MULTIPOINT of length 2; first list element:
- attr(*, "sf_column")= chr "sfc"
- attr(*, "agr")= Factor w/ 3 levels "constant","aggregate"..
- attr(*, "names")= chr "attribute"
```

```
> plot(d_sf)
```

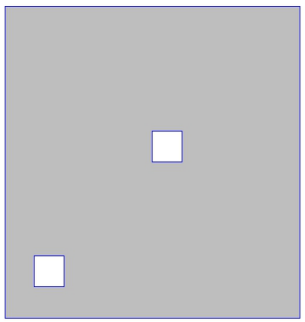


Example 2. In this example we create a polygon sf object with two holes. First, we define the corners of 3 polygons:

```
> outer <-  matrix(c(0,0,10,0,10,10,0,10,0,0),ncol=2,
+ byrow=TRUE)
> hole1 <-  matrix(c(1,1,1,2,2,2,2,1,1,1),ncol=2, byrow=TRUE)
> hole2 <-  matrix(c(5,5,5,6,6,6,6,5,5,5),ncol=2, byrow=TRUE)
```

Then we combine them and transform into the st polygon object:

```
> pts  <-  list(outer, hole1, hole2)
> pl  <-  st_polygon(pts)
> plot(pl, col="grey",border="blue")
```



Creating a spatial object from a table with coordinates

Example 3.

Consider the case when one has a txt or a similar file that contains latitudes, longitudes and some attribute values of several locations.

First let us read it into an R data frame with READ.TABLE or READ.CSV.

```
> CRAN_df <- read.table("CRAN051001a.txt", header = TRUE)
> str(CRAN_df)
'data.frame': 54 obs. of  6 variables:
$ place: chr  "Brisbane" "Melbourne" "Wien" "Curitiba" ...
$ north: chr  "27d28'S" "37d49'S" "48d13'N" "25d25'S" ...
$ east : chr  "153d02'E" "144d58'E" "16d20'E" "49d16'W" ...
$ loc  : chr  "Australia" "Australia" "Austria" "Brazil" ...
$ long : num  153 145 16.3 -49.3 -42.9 ...
$ lat  : num  -27.5 -37.8 48.2 -25.4 -20.8 ...
```

Then, use the function **st_as_sf()** to convert it into a spatial object in R:

```
> CRAN_sf <- st_as_sf(CRAN_df, coords = c("long", "lat"))
> str(CRAN_sf)
Classes sf and 'data.frame': 54 obs. of  5 variables:
$ place   : chr  "Brisbane" "Melbourne" "Wien" "Curitiba" ...
$ north   : chr  "27d28'S" "37d49'S" "48d13'N" "25d25'S" ...
$ east    : chr  "153d02'E" "144d58'E" "16d20'E"  ...
$ loc     : chr  "Australia" "Australia" "Austria"  ...
$ geometry:sfc_POINT of length 54 ...
```

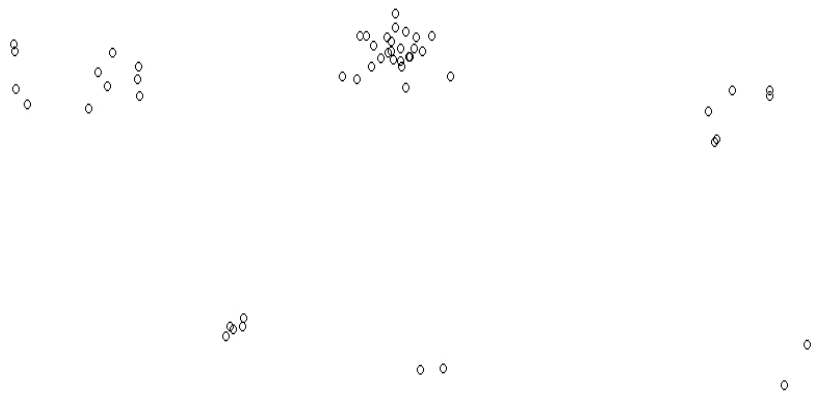
Finally, we assign the WGS84 projection, which has the EPSG code 4326:

```
> st_crs(CRAN_sf)
Coordinate Reference System: NA
> st_crs(CRAN_sf) <- 4326
> st_crs(CRAN_sf)
Coordinate Reference System: User input: EPSG:4326 ...
```

Plotting sf objects

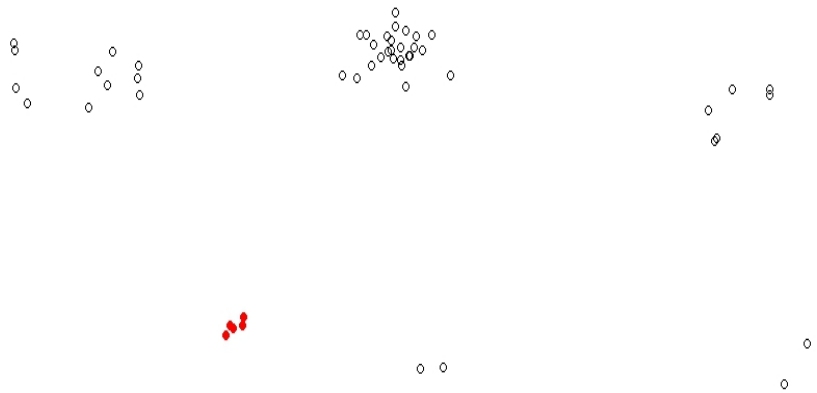
To plot an sf object we directly use the geometry column by extracting it with the function **st_geometry()**

```
> plot(st_geometry(CRAN_sf))
```



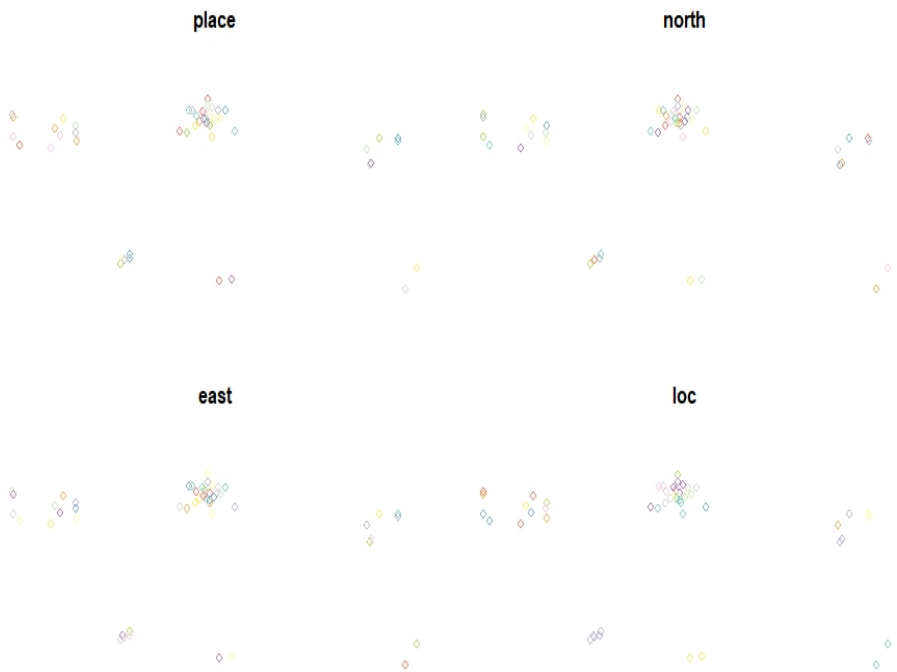
One can extract elements from a sf object using the usual methods of subsetting data frames:

```
> CRAN_sf_Brazil <- CRAN_sf[CRAN_sf$loc == "Brazil", ]
> plot(st_geometry(CRAN_sf_Brazil),add=T,col="red",pch = 16)
```

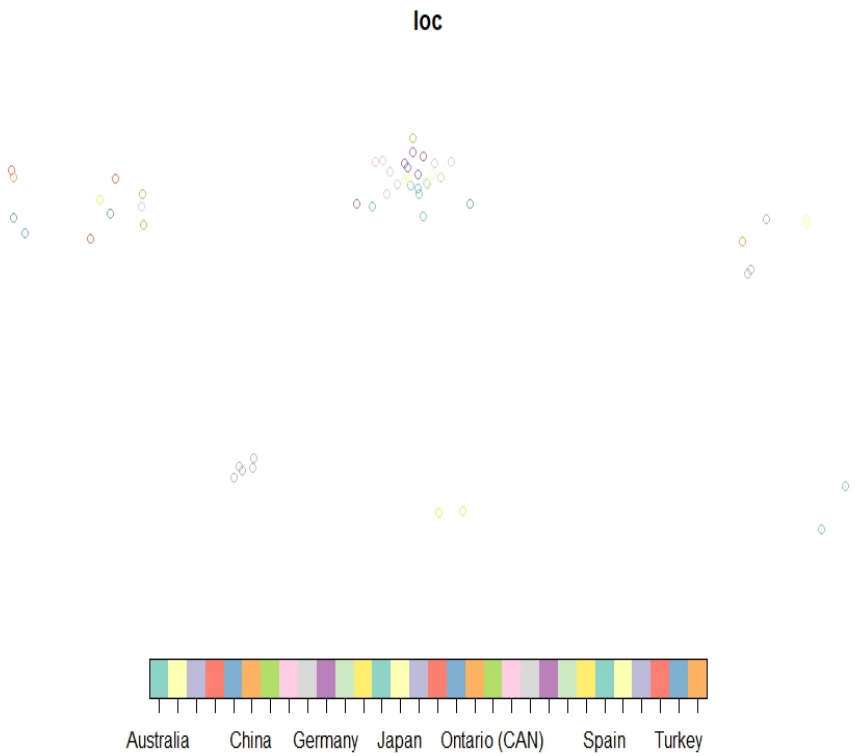


Using the generic command PLOT one can visualise all attributes in separate plots or specify an attribute for visualisation:

```
> plot(CRAN_sf)
```



```
> plot((CRAN_sf)[4])
```



Package MAPVIEW can be used to create interactive html pages with maps.

The function **mapview** produces an interactive map of a sf spatial object on top of a base map.

```
> library(mapview)
> mapview(CRAN_sf, col.regions = "red", fgb = FALSE)
```

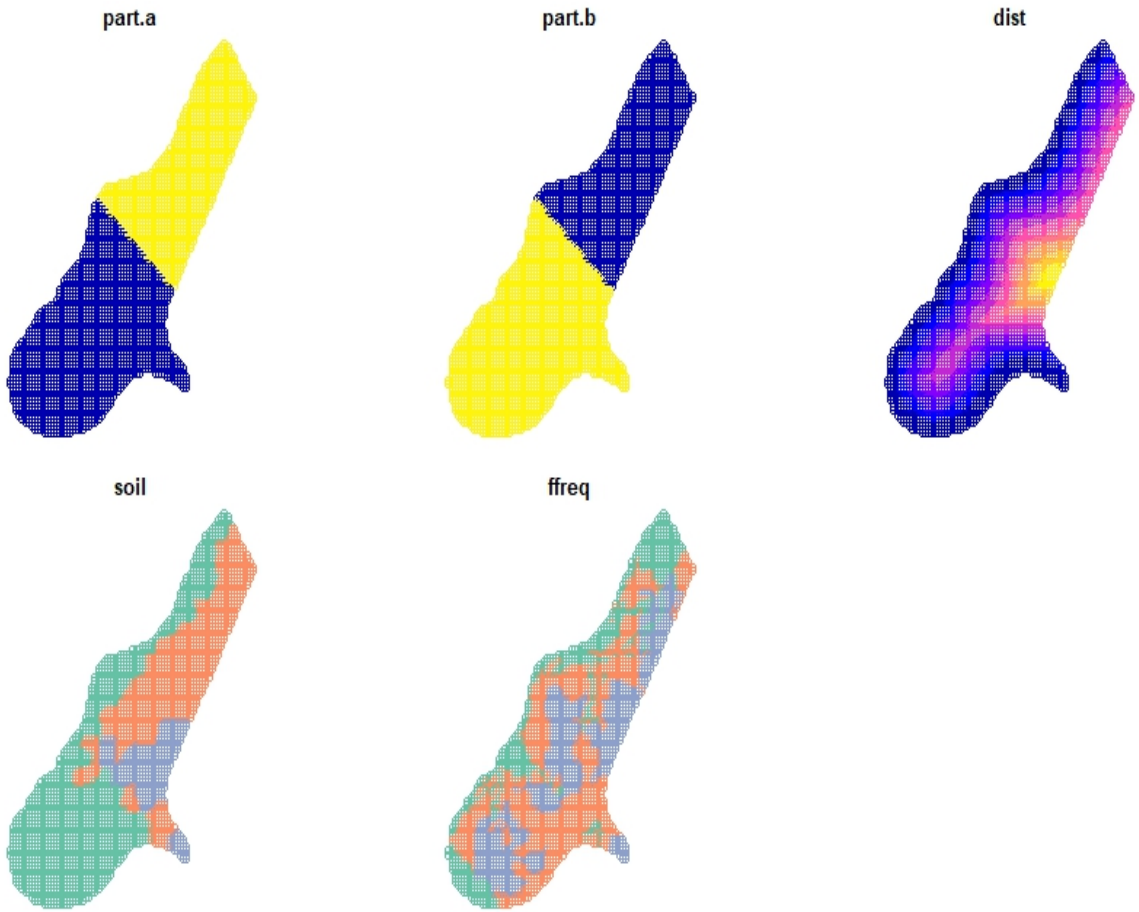
To combine several maps use "+", i.e. mapview(...)+mapview(...)+...

Conversion between sf and sp

Spatial objects from SP can be converted into objects in SF by using the function **st_as_sf**:

```
> library(sp)
> library(lattice)
> data(meuse.grid)
> coordinates(meuse.grid) <- ~ x + y
> m.sf <- st_as_sf(meuse.grid)
> str(m.sf)
Classes sf and 'data.frame': 3103 obs. of  6 variables:
$ part.a   : num  1 1 1 1 1 1 1 1 1 1 ...
$ part.b   : num  0 0 0 0 0 0 0 0 0 0 ...
$ dist     : num  0 0 0.0122 0.0435 0 ...
$ soil     : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1  ...
$ ffreq    : Factor w/ 3 levels "1","2","3": 1 1 1 1 1  ...
$ geometry:sfc_POINT of length 3103; first list element:  ..
```

```
> plot(m.sf)
```



Vice versa, the conversion of simple feature objects from SF or SFC into corresponding Spatial* objects can be done by using the function AS:

```
> CRAN_sp <- as(CRAN_sf, "Spatial")
> str(CRAN_sp)
Formal class 'SpatialPointsDataFrame' [package "sp"] ...
..@ data       :'data.frame': 54 obs. of  4 variables:
.. ..$ place: chr [1:54] "Brisbane" "Melbourne" "Wien"  ...
.. ..$ north: chr [1:54] "27d28'S" "37d49'S" "48d13'N" ...
.. ..$ east : chr [1:54] "153d02'E" "144d58'E" "16d20'E" ...
```