Spatial and spatio-temporal data in R

ifgi
Institute for Geoinformatics
University of Münster

Edzer Pebesma

DailyMeteo, Belgrade, 24-27 Jun 2014

http://ifgi.uni-muenster.de/~epebe_01/R/

Overview

1. vector, matrix, array; lists, data.frame
2. selection on data.frame
3. spatial classes
4. selection on spatial objects
5. aggregation, in general
6. aggregation on spatial classes
7. spatio-temporal classes
8. selection, overlay, aggregation on spatio-temporal objects

All data are spatio-temporal

1. There are no pure-spatial data. Maps reflect either
   - a snapshot in time (remote sensing image)
   - an aggregate over a time period (e.g., interpolated *yearly average* temperature, or yearly aggregated daily interpolations)
   - something that is constant over a period of time (political boundary)
   - a seemingly non-changing phenomenon (geology)
2. There are no pure-temporal data. Time series reflect either
   - spatially aggregated values (global temperature curves)
   - a single spatial location (air quality sensor `DEUB032`, at 8.191934E,50.93033N)
   - vaguely located, or universal aggregates (world market prices, stock quotes)

Vector, matrix, array

```
> a = vector(3, mode = "numeric")
> a

[1] 0 0 0

> length(a)

[1] 3
```

or simply by initialisation

```
> c = 1:3
> c

[1] 1 2 3

> typeof(c)

[1] "integer"

> d = 1.5:3.5
> d

[1] 1.5 2.5 3.5

> typeof(c)
```

```
> m = matrix(rnorm(6), 2, 3)
> print(m, digits=3)

       [,1]   [,2]   [,3]
[1,]  0.415  0.735 -0.171
[2,] -1.115 -0.540 -1.029

> dim(m)

[1] 2 3

> a = array(1:(5*7*9), c(5,7,9))
> dim(a)

[1] 5 7 9
```

lists, data.frame

Lists can contain anything:

```
> a = list(1:3, x = c("foo", "bar"), c(TRUE, FALSE))
> a

[[1]]
[1] 1 2 3

$x
[1] "foo" "bar"

[[3]]
[1]  TRUE FALSE

> a[1]

[[1]]
[1] 1 2 3

> a[[1]]

[1] 1 2 3

> a$x

[1] "foo" "bar"
```

data.frame is a column store, but mimics records of mixed type

```
> a[[1]] = 1:2
> b = as.data.frame(a)
> names(b) = c("NR", "what", "cond")
> b

  NR what  cond
1  1  foo  TRUE
2  2  bar FALSE

> is.list(b)

[1] TRUE
```

Selection on data.frame

```
> b

  NR what  cond
1  1  foo  TRUE
2  2  bar FALSE

> b[[1]]

[1] 1 2

> b[["NR"]]

[1] 1 2

> b$NR

[1] 1 2

> b[1]

  NR
1  1
2  2

> b[1,]

  NR what cond
1  1  foo TRUE

> b[,1:2]

  NR what
1  1  foo
2  2  bar

> b[,1]

[1] 1 2

> b[,1,drop=FALSE]

  NR
1  1
2  2
```

Deletion, negative selection, replacement

```
> b

  NR what  cond
1  1  foo  TRUE
2  2  bar FALSE

> b$NR = NULL
> b

  what  cond
1  foo  TRUE
2  bar FALSE

> b[-1,]

  what  cond
2  bar FALSE
```

```
> b$cond2 = ! b$cond
> b

  what  cond cond2
1  foo  TRUE FALSE
2  bar FALSE  TRUE

> b[1,1] = NA
> b

  what  cond cond2
1 <NA>  TRUE FALSE
2  bar FALSE  TRUE

> class(b$what)

[1] "factor"

> b$what

[1] <NA> bar
Levels: bar foo

> as.numeric(b$what)

[1] NA  1
```

Spatial classes (sp)

Class `Spatial` provides a coordinate reference system and a bounding box.

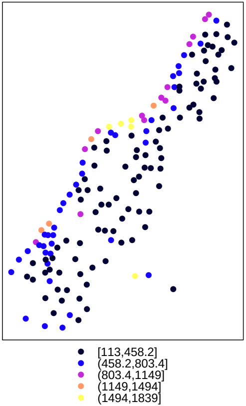
- Objects deriving from `Spatial` consist of a geometry:
  - `SpatialPoints`
  - `SpatialLines`
  - `SpatialPolygons`
  - `SpatialPixels`
  - `SpatialGrid`
- from these, `Spatial*DataFrame` objects derive, and have attributes (a data slot, of class `data.frame`)

```
> library(sp)
> library(rgdal)
> p = SpatialPoints(cbind(lon = 8, lat = 52), CRS("+init=epsg:4326"))
> p

SpatialPoints:
     lon lat
[1,]   8  52
Coordinate Reference System (CRS) arguments: +init=epsg:4326
+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84
+towgs84=0,0,0
```

Meuse data set

```
> library(sp)
> data("meuse")
> coordinates(meuse) <- ~x+y
> spplot(meuse["zinc"],
+         col.regions = bpy.colors())
```



Getting spatial data in R

Usually, we don't create spatial objects from scratch, but from external files (`readGDAL`, `readOGR`), or from `data.frame` objects:

```
> library(sp)
> data(meuse)
> class(meuse)

[1] "data.frame"

> dim(meuse)

[1] 155  14

> names(meuse)[1:6]

[1] "x"       "y"       "cadmium" "copper"  "lead"    "zinc"

> coordinates(meuse) = c("x", "y")
> # which is short for:
> data(meuse)
> pts = SpatialPoints(meuse[c("x", "y")])
> m = SpatialPointsDataFrame(pts, meuse)
> class(m)

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

Selection on spatial objects

Selecting the `data.frame` metaphor:

- selection of records (features), attributes (columns):

```
> meuse[1:3,1:6]
```

```
       x      y cadmium copper lead zinc
1 181072 333611    11.7     85  299 1022
2 181025 333558     8.6     81  277 1141
3 181165 333537     6.5     68  199  640
```

- extraction of variables:

```
> meuse$zinc[1:3]
```

```
[1] 1022 1141  640
```

- replacement:

```
> meuse$zinc[1:2] = NA
> meuse[1:3,1:6]
```

```
       x      y cadmium copper lead zinc
1 181072 333611    11.7     85  299   NA
2 181025 333558     8.6     81  277   NA
3 181165 333537     6.5     68  199  640
```

Aggregation, in general

```
> d = data.frame(x = 1:6, grp1 = c(rep("A",3), rep("B",3)))
> d$grp2 = rep(c("P","Q","R"), each = 2)
> d

  x grp1 grp2
1 1    A    P
2 2    A    P
3 3    A    Q
4 4    B    Q
5 5    B    R
6 6    B    R

> aggregate(d[1], list(d$grp1), mean)

  Group.1 x
1       A 2
2       B 5

> aggregate(d[1], list(d$grp1, d$grp2), mean)

  Group.1 Group.2   x
1       A       P 1.5
2       A       Q 3.0
3       B       Q 4.0
4       B       R 5.5
```

Aggregation, needs:

```
## S3 method for class 'data.frame'
aggregate(x, by, FUN, ..., simplify = TRUE)
```

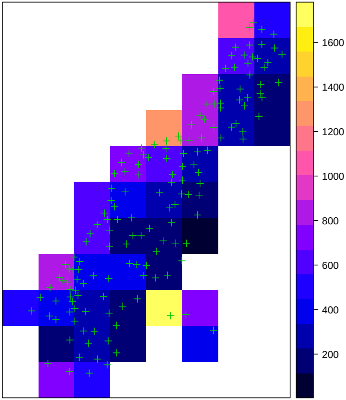
- an object to aggregate (`x`)
- an aggregation predicate (`by`)
- an aggregation function (`FUN`)

NOTE that

- we pass functions as arguments → R is a functional programming language
- we can write our own function, and pass it to `FUN`
- `...` is passed on to this function

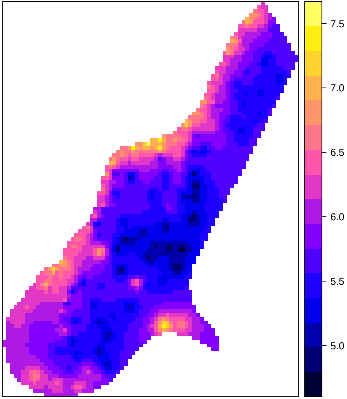
Aggregation of spatial classes

```
> library(sp)
> data("meuse")
> coordinates(meuse) <- ~x+y
> offset = c(178460, 329620)+20
> gt = GridTopology(offset, c(400,400),
+          c(8,11))
> SG = SpatialGrid(gt)
> agg = aggregate(meuse["zinc"], SG)
> spplot(agg["zinc"],
+          col.regions=bpy.colors(),
+          sp.layout = list("sp.points",
+          meuse, col=3))
```



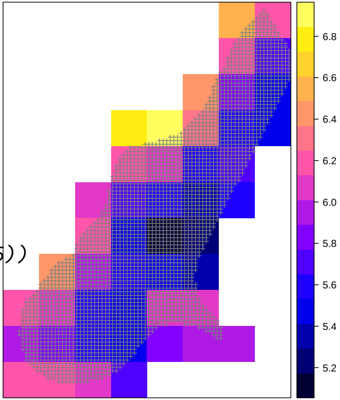
Aggregating interpolated values

```
> library(sp)
> data("meuse")
> coordinates(meuse) <- ~x+y
> data("meuse.grid")
> coordinates(meuse.grid) <- ~x+y
> gridded(meuse.grid) <- TRUE
> library(gstat)
> x = idw(log(zinc)~1, meuse,
+   meuse.grid, debug.level=0)[1]
> spplot(x[1],col.regions=bpy.colors())
```



Aggregating interpolated values, spatially

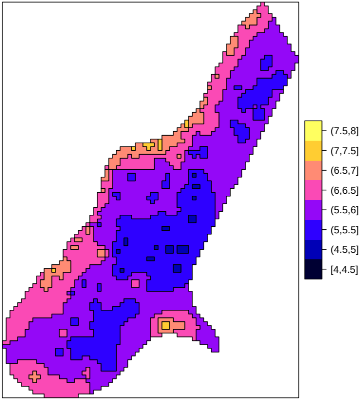
```
> agg = aggregate(x["var1.pred"], SG)
> spplot(agg, col.regions=bpy.colors(),
+         sp.layout = list("sp.points",
+         meuse.grid, col=grey(.5), cex=.5))
```



NOTE: the aggregation predicate (SG) can be of any type: points, lines, polygons, grid.

Aggregating by attribute value

```
> i = cut(x$var1.pred, seq(4, 8, by=.5),
+  include.lowest = TRUE)
> xa = aggregate(x["var1.pred"],
+  list(i=i))
> spplot(xa[1],col.regions=bpy.colors(8))
```



(NOTE: this is still in sp on r-forge, not on CRAN)

Spatio-temporal classes

Package `spacetime` tries to combine all cleverness of spatial data in `sp`, of temporal data in `zoo` and `xts`, and then add some. It mainly solves:

- object creation (e.g. from tables, sp and/or xts objects),
- some I/O (RasterStack with time z; TGRASS, PostGIS)
- selection (space, time, attributes)
- aggregation (over space, over time, over space-time)
- plotting

meteo: precipitation and stations

```
> library(meteo)
> data(dprec); head(dprec, 2)

        staid       time prec
1        2707 2011-07-01  0.2
2 67240-99999 2011-07-01  0.0

> data(stations); head(stations, 2)

     staid   lon      lat elev_1m data_source station_name
9602     1 14.80 56.86667     166         ECA      VAEXJOE
9512    10 18.05 59.35000      44         ECA    STOCKHOLM

> mtch = match(dprec$staid, stations$staid)
> dprec = data.frame(dprec, stations[mtch, c("lon", "lat")])
> head(dprec, 2)

        staid       time prec      lon      lat
1        2707 2011-07-01  0.2 7.241389 62.24778
2 67240-99999 2011-07-01  0.0 7.467000 46.30000
```

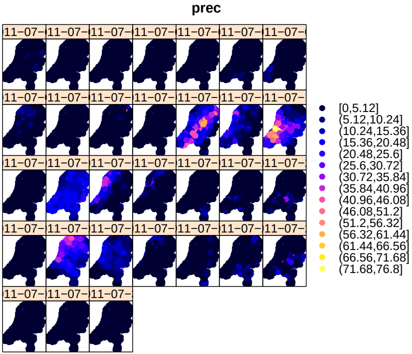
meteo: precipitation and stations

```
> library(spacetime)
> m = stConstruct(dprec, c("lon", "lat"), "time")
>         #, crs = CRS("+init=epsg:4326"))
> m2 = as(m, "STFDF")
> summary(m2[,,"prec"])
```

```
Object of class STFDF
 with Dimensions (s, t, attr): (12923, 31, 1)
[[Spatial:]]
Object of class SpatialPoints
Coordinates:
         min    max
lon -179.633 179.75
lat  -90.000  83.65
Is projected: NA
proj4string : [NA]
Number of points: 12923
[[Temporal:]]
     Index                 timeIndex
 Min.   :2011-07-01   Min.   : 1.0
 1st Qu.:2011-07-08   1st Qu.: 8.5
 Median :2011-07-16   Median :16.0
 Mean   :2011-07-16   Mean   :16.0
```

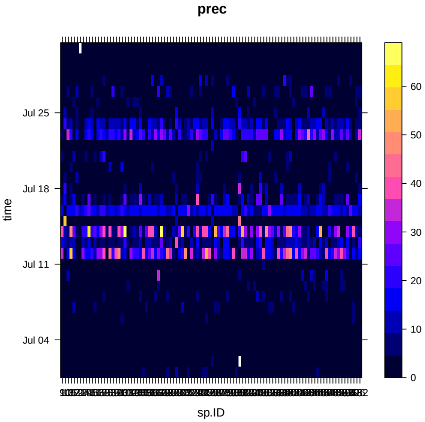
plot: map-panel

```
> data(NLpol) # in meteo!
> proj4string(m2) = proj4string(NLpol)
> m2.NL = m2[NLpol,]
> stplot(m2.NL[,,"prec"],
+         col.regions = bpy.colors())
```



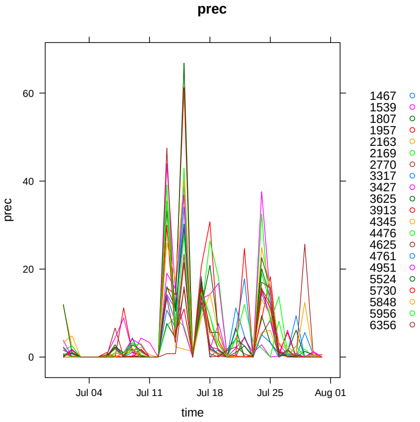
xt: space-time (Hovmöller)

```
> proj4string(m2) = proj4string(NLpol)
> m2.NL = m2[NLpol,]
> stplot(m2.NL[1:100,,"prec"],
+         col.regions=bpy.colors(),
+         mode="xt")
```



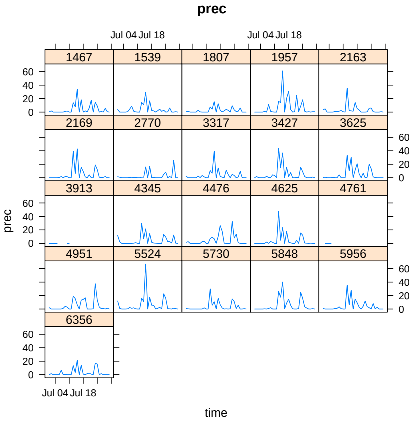
ts: time series

```
> proj4string(m2) = proj4string(NLpol)
> m2.NL = m2[NLpol,]
> stplot(m2.NL[120:140,,"prec"],
+         mode="ts")
```



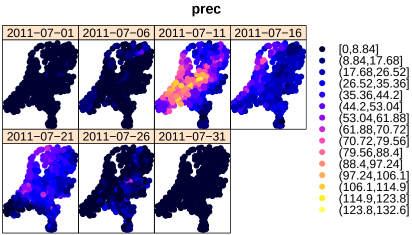
time-panel

```
> proj4string(m2) = proj4string(NLpol)
> m2.NL = m2[NLpol,]
> stplot(m2.NL[120:140,,"prec"],
+          mode="tp")
```



Aggregation on spatio-temporal objects

```
> m2.agg = aggregate(m2.NL[,,"prec"],
+          "5 days", sum)
> stplot(m2.agg[,,"prec"],
+          col.regions=bpy.colors())
```



Aggregation on spatio-temporal objects

```
> m2.aggNL = aggregate(m2.NL[,,"prec"],
+           NLpol, mean, na.rm=TRUE)
> plot(m2.aggNL)
```