Map overlay and spatial aggregation in sp

Edzer Pebesma*

Nov 2010

Contents

1 Introduction **1**

2 Geometry overlays **2**

3 Extraction, aggregation of attributes **5**
3.1 Extracting attribute values 5
3.2 Aggregation 7

4 More tests, and their output **7**

1 Introduction

According to the free e-book Reading topographic maps by Robert Davidson,

> *An overlay is a clear sheet of plastic or semi-transparent paper. It is used to display supplemental map and tactical information related to military operations. It is often used as a supplement to orders given in the field. Information is plotted on the overlay at the same scale as on the map, aerial photograph, or other graphic being used. When the overlay is placed over the graphic, the details plotted on the overlay are shown in their true position.*

This suggests that *map overlay* is concerned with combining two, or possibly more, map layers by putting them on top of each other. This kind of overlay can be obtained in R e.g. by plotting one map layer, and plotting a second map layer on top of it. If the second one contains polygons, transparent colours can be used to avoid hiding of the first layer. When using the `spplot` command, the `sp.layout` argument can be used to combine multiple layers.

*Institute for Geoinformatics, University of Muenster, Weseler Strasse 253, 48151 Münster, Germany. edzer.pebesma@uni-muenster.de

O'Sullivan and Unwin (2003) argue in chapter 10 (Putting maps together: map overlay) that map overlay has to do with the combination of two (or more) maps. They mainly focus on the combination of the selection criteria stemming from several map layers, e.g. finding the deciduous forest area that is less than 5 km from the nearest road. They call this *boolean overlays*.

One could look at this problem as a polygon-polygon overlay, where we are looking for the intersection of the polygons with the deciduous forest with the polygons delineating the area less than 5 km from a road. Other possibilities are to represent one or both coverages as grid maps, and find the grid cells for which both criteria are valid (grid-grid overlay). A third possibility would be that one of the criteria is represented by a polygon, and the other by a grid (polygon-grid overlay, or grid-polygon overlay). In the end, as O'Sullivan and Unwin argue, we can overlay any spatial type (points, lines, polygons, pixels/grids) with any other. In addition, we can address spatial attributes (as the case of grid data), or only the geometry (as in the case of the polygon-polygon intersection).

This vignette will explain how the `over` method in package `sp` can be used to compute map overlays, meaning that instead of to overlaying maps visually, the the information that comes from combining two map layers is retrieved.

2 Geometry overlays

We will use the word *geometry* to denote the purely spatial characteristics, meaning that attributes (properties of a particular geometry) are ignored. With *location* we denote a point, line, polygon or grid cell.

Given two geometries, `A` and `B`, the following equivalent commands

```
> A %over% B
> over(A, B)
```

will retrieve the geometry (location) indexes of B at the locations of A. More in particular, it returns an integer vector of length `length(A)` that has `NA` values for locations in `A` not matching with locations in `B` (e.g. those points outside a set of polygons). Selecting points *inside* some geometry `B` is done by

```
> A[B]
```

which is equivalent to

```
> A[!is.na(over(A, B))]
```

We will now illustrate this with toy data created by

```
> library(sp)
> x = c(0.5, 0.5, 1.2, 1.5)
> y = c(1.5, 0.5, 0.5, 0.5)
> xy = cbind(x,y)
> dimnames(xy)[[1]] = c("a", "b", "c", "d")
> pts = SpatialPoints(xy)
```

```
> plot(pol, xlim = c(-1.1, 2.1), ylim = c(-1.1, 1.6), border = 2:6,
+     axes = TRUE)
> points(pts, col = "red")
> text(c(-1, 0.1, 0.1, 1.1, 0.45), c(0, 0, -1, 0, 0.1), c("x1",
+     "x2", "x3", "x4", "x5"))
> text(coordinates(pts), pos = 1, row.names(pts))
```

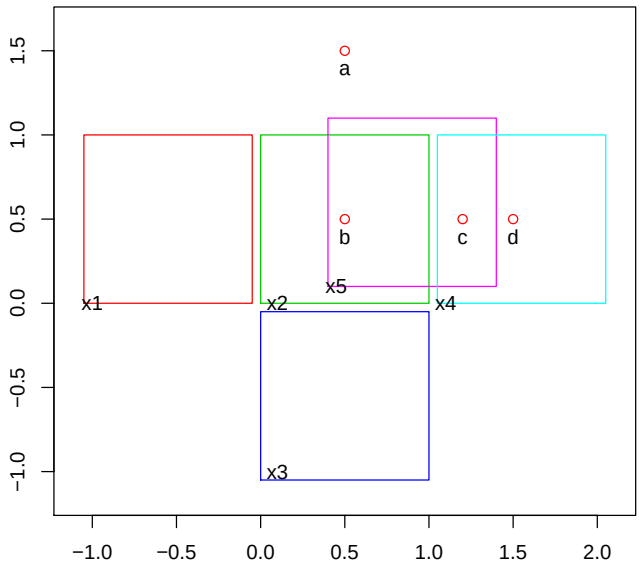

Figure 1: Toy data: points (a-d), and (overlapping) polygons (x1-x5)

```
> xpol = c(0,1,1,0,0)
> ypol = c(0,0,1,1,0)
> pol = SpatialPolygons(list(
+         Polygons(list(Polygon(cbind(xpol-1.05,ypol))), ID="x1"),
+         Polygons(list(Polygon(cbind(xpol,ypol))), ID="x2"),
+         Polygons(list(Polygon(cbind(xpol,ypol-1.05))), ID="x3"),
+         Polygons(list(Polygon(cbind(xpol+1.05,ypol))), ID="x4"),
+         Polygons(list(Polygon(cbind(xpol+.4, ypol+.1))), ID="x5")
+         ))
```

and shown in figure 1.

```
> over(pts, pol)
```

```
[1] NA  5  5  4
```

```
> over(pts, pol, returnList = TRUE)
```

```
[[1]]
integer(0)

[[2]]
[1] 2 5

[[3]]
[1] 4 5

[[4]]
[1] 4
```

```
> pts[pol]
```

```
SpatialPoints:
    x   y
b 0.5 0.5
c 1.2 0.5
d 1.5 0.5
Coordinate Reference System (CRS) arguments: NA
```

```
> over(pol, pts)
```

```
[1] NA  2 NA  3  2
```

```
> over(pol, pts, returnList = TRUE)
```

```
[[1]]
integer(0)

[[2]]
[1] 2

[[3]]
integer(0)

[[4]]
[1] 3 4

[[5]]
[1] 2 3
```

```
> row.names(pol[pts])
```

```
[1] "x2" "x4" "x5"
```

3 Extraction, aggregation of attributes

This section will demonstrate how, using `over`, attribute values of argument `y` at locations of `x` can be extracted, and either aggregated to a data frame, or reported as a list.

3.1 Extracting attribute values

This example creates a `SpatialPointsDataFrame` and a `SpatialPolygonsDataFrame` for the toy data created earlier:

```
> zdf = data.frame(z1 = 1:4, z2 = 4:1, f = c("a", "a", "b", "b"),
+     row.names = c("a", "b", "c", "d"))
> zdf
```

```
  z1 z2 f
a  1  4 a
b  2  3 a
c  3  2 b
d  4  1 b
```

```
> ptsdf = SpatialPointsDataFrame(pts, zdf)
> zpl = data.frame(z = c(10, 15, 25, 3, 0), zz = 1:5, f = c("z",
+     "q", "r", "z", "q"), row.names = c("x1", "x2", "x3", "x4",
+     "x5"))
> zpl
```

```
    z zz f
x1 10  1 z
x2 15  2 q
x3 25  3 r
x4  3  4 z
x5  0  5 q
```

```
> poldf = SpatialPolygonsDataFrame(pol, zpl)
```

The first example creates a data.frame, where for each element in `pts` the *first* element of the corresponding attribute table record of the second argument is taken:

```
> over(pts, poldf)
```

```
   z zz    f
a NA NA <NA>
b 15  2    q
c  3  4    z
d  3  4    z
```

As an alternative, we can pass a user-defined function to process the table:

```
> over(pts, poldf[1:2], fn = mean)

    z  zz
a  NA  NA
b 7.5 3.5
c 1.5 4.5
d 3.0 4.0
```

To obtain the list of table entries for each point, use the `returnList` argument:

```
> over(pts, poldf, returnList = TRUE)

[[1]]
[1] z  zz f
<0 rows> (or 0-length row.names)

[[2]]
    z zz f
x2 15  2 q
x5  0  5 q

[[3]]
   z zz f
x4 3  4 z
x5 0  5 q

[[4]]
   z zz f
x4 3  4 z
```

The same actions can be done when the arguments are reversed:

```
> over(pol, ptsdf)

   z1 z2    f
x1 NA NA <NA>
x2  2  3    a
x3 NA NA <NA>
x4  3  2    b
x5  2  3    a

> over(pol, ptsdf[1:2], fn = mean)

    z1  z2
x1  NA  NA
x2 2.0 3.0
x3  NA  NA
x4 3.5 1.5
x5 2.5 2.5
```

3.2 Aggregation

4 More tests, and their output

```
> gt = GridTopology(c(0.5, 0.5), c(1, 1), c(3, 2))
> sg = SpatialGrid(gt)
> df6 = data.frame(z = 6:1, f = c("a", "a", "b", "b", "c", "c"))
> sgdf = SpatialGridDataFrame(gt, df6)
> over(sg, pol)

[1] NA NA NA  5  4 NA

> over(sg, poldf)

   z zz    f
1 NA NA <NA>
2 NA NA <NA>
3 NA NA <NA>
4 15  2    q
5  3  4    z
6 NA NA <NA>

> over(sg, poldf[1:2])

   z zz
1 NA NA
2 NA NA
3 NA NA
4 15  2
5  3  4
6 NA NA

> spix = as(sg, "SpatialPixels")
> spixdf = as(sgdf, "SpatialPixelsDataFrame")
> over(spix, pol)

[1] NA NA NA  5  4 NA

> over(spix, poldf)

   z zz    f
1 NA NA <NA>
2 NA NA <NA>
3 NA NA <NA>
4 15  2    q
5  3  4    z
6 NA NA <NA>
```

```
> data(meuse.grid)
> gridded(meuse.grid) = ~x + y
> gt = GridTopology(gridparameters(meuse.grid)$cellcentre.offset,
+     c(400, 400), c(8, 11))
> SG = SpatialGrid(gt)
> if (as.numeric(version$minor) < 12) {
+     image(sp:::aggregate.Spatial(meuse.grid[3], SG))
+ } else {
+     image(aggregate(meuse.grid[3], SG))
+ }
```

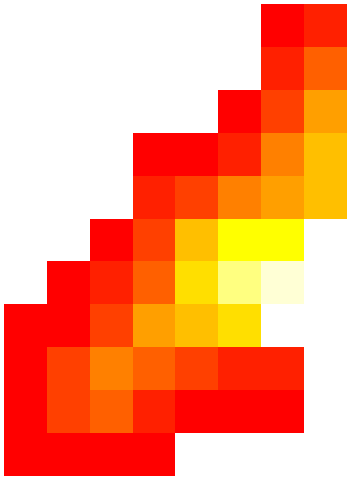

Figure 2: aggregation over meuse.grid distance values

```
> over(spix, poldf[1:2])

   z zz
1 NA NA
2 NA NA
3 NA NA
4 15  2
5  3  4
6 NA NA

> over(pol, sg)

[1] NA  4 NA  5  4

> over(pol, sgdf)

    z    f
x1 NA <NA>
x2  3    b
x3 NA <NA>
x4  2    c
x5  3    b

> over(pol, sgdf[1], fn = mean)

    z
x1 NA
x2  3
x3 NA
x4  2
x5  3

> over(pol, spix)

[1] NA  4 NA  5  4

> over(pol, spixdf)

    z    f
x1 NA <NA>
x2  3    b
x3 NA <NA>
x4  2    c
x5  3    b

> over(pol, spixdf[1], fn = mean)
```

```
    z
x1 NA
x2  3
x3 NA
x4  2
x5  3

> over(pts, sg)

[1] 1 4 5 5

> over(pts, spix)

[1] 1 4 5 5

> over(pts, sgdf)

  z f
a 6 a
b 3 b
c 2 c
d 2 c

> over(pts, spixdf)

  z f
a 6 a
b 3 b
c 2 c
d 2 c

> over(sg, sg)

[1] 1 2 3 4 5 6

> over(sg, spix)

[1] 1 2 3 4 5 6

> over(sg, sgdf)

  z f
1 6 a
2 5 a
3 4 b
4 3 b
5 2 c
6 1 c
```

```
> over(sg, spixdf)

  z f
1 6 a
2 5 a
3 4 b
4 3 b
5 2 c
6 1 c

> over(spix, sg)

[1] 1 2 3 4 5 6

> over(spix, spix)

[1] 1 2 3 4 5 6

> over(spix, sgdf)

  z f
1 6 a
2 5 a
3 4 b
4 3 b
5 2 c
6 1 c

> over(spix, spixdf)

  z f
1 6 a
2 5 a
3 4 b
4 3 b
5 2 c
6 1 c
```

References

- O'Sullivan, D., Unwin, D. (2003) Geographical Information Analysis. Wiley, NJ.