Abstract

The so-called “map overlay” is not very well defined and does not have a simple equivalent in space-time. This paper will explain how the over method for combining two spatial features (and/or grids), defined in package sp and extended in package rgeos, is implemented for spatio-temporal objects in package spacetime. It may carry out the numerical spatio-temporal overlay, and can be used for aggregation of spatio-temporal data over space, time, or space-time.

Contents

1 Introduction 1

2 Overlay with method over 2

3 Spatio-temporal overlay with method over 3
   3.1 Time intervals or time instances? 4

4 Aggregating spatio-temporal data 4
   4.1 Example data: PM10 5
   4.2 Spatial aggregation 5
   4.3 Temporal aggregation 7
   4.4 Spatio-temporal aggregation 11
   4.5 Time intervals 11

1 Introduction

The so-called map overlay is a key GIS operation that does not seem to have a very sharp definition. The over vignette in package sp (Pebesma, 2012) comments on what paper (visual) overlays are, and discusses the over and aggregate methods for spatial data.

In the ESRI ArcGIS tutorial (ESRI, 2012), it can be read that
An overlay operation is much more than a simple merging of linework; all the attributes of the features taking part in the overlay are carried through, as shown in the example below, where parcels (polygons) and flood zones (polygons) are overlayed (using the Union tool) to create a new polygon layer. The parcels are split where they are crossed by the flood zone boundary, and new polygons created. The FID_flood value indicates whether polygons are outside (-1) or inside the flood zone, and all polygons retain their original land use category values.

It later on mentions raster overlays, such as the addition of two (matching) raster layers (so, potentially the whole of map algebra functions, where two layers are involved).

In the open source arena, with no budgets for English language editing, the Grass 7.0 documentation mentions the following:

\texttt{v.overlay} allows the user to overlay two vector area maps. The resulting output map has a merged attribute-table. The origin column-names have a prefix (\texttt{a} and \texttt{b}) which results from the ainput- and binput-map. [...] Operator defines features written to output vector map Feature is written to output if the result of operation ‘ainput operator binput’ is true. Input feature is considered to be true, if category of given layer is defined. Options: and, or, not, xor.

2 Overlay with method over

We loosely define map overlay as

- an operation involving at least two maps
- asymmetric – overlay is different from underlay
- either a visual or a numerical activity.

The method \texttt{over}, as defined in package \texttt{sp}, provides a way to numerically combine two maps. In particular,

\begin{verbatim}
R> over(x, geometry(y))
\end{verbatim}

returns an integer vector of length \texttt{length(x)} with \texttt{x[i]} the index of \texttt{y}, spatially corresponding to \texttt{x[i]}, so \texttt{x[i]=j} means that \texttt{x[i]} and \texttt{y[j]} match (have the same location, touch, or overlap/intersect etc.), or \texttt{x[i]=NA} if there is no match. If \texttt{y} has data values (attributes), then

\begin{verbatim}
R> over(x, y)
\end{verbatim}

retrieves a \texttt{data.frame} with \texttt{length(x)} rows, where row \texttt{i} contains the attributes of \texttt{y} at the spatial location of \texttt{x[i]}, and NA values if there is no match. If the relationship is more complex, e.g. a polygon or grid cell \texttt{x} containing more than one point of \texttt{y}, the command

\begin{verbatim}
R> over(x, y, returnList = TRUE)
\end{verbatim}

returns a list of length \texttt{length(x)}, with each list element a numeric vector with all indices if \texttt{y} is geometry only, or else a data frame with all attribute table rows of \texttt{y} that spatially matches \texttt{x[i]}.
3 Spatio-temporal overlay with method \texttt{over}

Package \texttt{spacetime} adds \texttt{over} methods to those defined for spatial data in package \texttt{sp}:

\begin{verbatim}
R> library(sp)
R> library(spacetime)
R> showMethods(over)

Function: over (package sp)
  x="ST", y="STS"
  x="STF", y="STF"
  x="STF", y="STFDF"
  x="STF", y="STI"
  x="STF", y="STIDF"
  x="STF", y="STSDF"
  x="STI", y="STF"
  x="STI", y="STFDF"
  x="STI", y="STI"
  x="STI", y="STIDF"
  x="STI", y="STSDF"
  x="STS", y="STF"
  x="STS", y="STFDF"
  x="STS", y="STI"
  x="STS", y="STIDF"
  x="STS", y="STSDF"
  x="Spatial", y="Spatial"
  x="SpatialGrid", y="SpatialGrid"
  x="SpatialGrid", y="SpatialGridDataFrame"
  x="SpatialGrid", y="SpatialPixels"
  x="SpatialGrid", y="SpatialPixelsDataFrame"
  x="SpatialGrid", y="SpatialPoints"
  x="SpatialGrid", y="SpatialPolygons"
  x="SpatialGrid", y="SpatialPolygonsDataFrame"
  x="SpatialPoints", y="SpatialGrid"
  x="SpatialPoints", y="SpatialGridDataFrame"
  x="SpatialPoints", y="SpatialPixels"
  x="SpatialPoints", y="SpatialPixelsDataFrame"
  x="SpatialPoints", y="SpatialPolygons"
  x="SpatialPoints", y="SpatialPolygonsDataFrame"
  x="SpatialPolygons", y="SpatialGrid"
  x="SpatialPolygons", y="SpatialGridDataFrame"
  x="SpatialPolygons", y="SpatialPoints"
  x="SpatialPolygons", y="SpatialPointsDataFrame"
  x="SpatialPolygonsDataFrame", y="SpatialPoints"
    (inherited from: x="SpatialPolygons", y="SpatialPoints")

\end{verbatim}

3
3.1 Time intervals or time instances?

When computing the overlay

\[
R > \text{over}(x, y)
\]

A space-time feature matches another space-time feature when their spatial locations match (coincide, touch, intersect or overlap), and when their temporal properties match. For temporal properties, it is crucial whether time is a time interval, or a time instance. When all \texttt{endTime} values are equal to the \texttt{time} times, time is considered instance. When one or more \texttt{endTime} values are larger than \texttt{time}, time is considered to reflect intervals.

Suppose we have two time sequences, \( T : t_1, t_2, \ldots, t_n \) and \( U : u_1, u_2, \ldots, u_m \). Both are ordered: \( t_i \leq t_{i+1} \).

Both \( T \) and \( U \) can reflect time \textit{instances} or time \textit{intervals}. In case they reflect time \textit{instances}, an observation at \( t_i \) takes place at the time instance \( t_i \), and has an unregistered (possibly ignorable) duration. In case they reflect time \textit{intervals}, an observation “at” \( t_i \) takes place during, or is representative for, the time interval \( t_i \leq t < t_{i+1} \). (The last time interval \( t_n \) is obtained by adopting the one-but-last time interval duration: \( t_n \leq t < t_n + (t_n - t_{n-1}) \)).

We define the time (instance or interval) pair \( \{t_i, u_j\} \) to match if

for \( T \) instance, \( U \) instance:

\[
t_i = u_j
\]

for \( T \) interval, \( U \) instance

\[
t_i \leq u_j < t_{i+1}
\]

for \( T \) instance, \( U \) interval

\[
u_j \leq t_i < u_{j+1}
\]

for \( T \) interval, \( U \) interval

\[
\exists t : t_i \leq t < t_{i+1} \land u_j \leq t < u_{j+1}
\]

which can be rephrased as the negation of \( t_{i+1} \leq u_j \lor t_i \geq u_{j+1} \) (where \( \lor \) denotes ‘or’), or alternatively expressed as

\[
t_{i+1} > u_j \land t_i < u_{j+1}
\]

where \( \land \) denotes ‘and’.

All these conditions fail for intervals having zero width (empty intervals), i.e. the case where \( T \) is interval and for some \( i, t_{i+1} - t_i = 0 \) or the case where \( U \) is interval and for some \( j, u_{j+1} - u_j = 0 \).

4 Aggregating spatio-temporal data

The \texttt{aggregate} method for a \texttt{data.frame} is defined as

\[
R > \text{aggregate}(x, by, FUN, \ldots, simplify = TRUE)
\]
where \( x \) is the data.frame to be aggregated, by indicates how groups of \( x \) are formed, FUN is applied to each group, and simplify indicates whether the output should be simplified (to vector), or remain a data.frame. The ... are passed to FUN, e.g. passing na.rm=TRUE is useful when FUN is mean and missing values need to be ignored.

For spatio-temporal data, the \( x \) argument needs to be of class STFDF, STSDF or STIDF. The by argument needs to specify an aggregation medium: time, space, or space-time.

### 4.1 Example data: PM10

Air quality example data are loaded by

```r
R> data(air)
R> rural = STFDF(stations, dates, data.frame(PM10 = as.vector(air)))
R> class(rural)
[1] "STFDF"
attr(,"package")
[1] "spacetime"
R> class(DE_NUTS1)
[1] "SpatialPolygonsDataFrame"
attr(,"package")
[1] "sp"
```

it provides PM10 daily mean values (taken from AirBase - the European Air quality database), for Germany, 1998-2009, where only stations classified as rural background were selected. The object DE_NUTS1 contains NUTS-1 level state boundaries for Germany, downloaded from GADM.

### 4.2 Spatial aggregation

To aggregate completely over space, we can coerce the data to a matrix and apply a function to the rows:

```r
R> x = as(rural[,"2008"], "xts")
R> apply(x, 1, mean, na.rm=TRUE)[1:5]
17.34950 16.06945 25.60065 27.24141 24.03417
```

A more refined spatial aggregation of time series can be obtained by grouping them to the state (“Bundesland”) level. Here, states are passed as a SpatialPolygons object:

```r
R> dim(rural[,"2008"])
space time variables
70 366 1
R> x = aggregate(rural[,"2008"], DE_NUTS1, mean, na.rm=TRUE)
R> dim(x)
```
space time variables

13 366 1

R> summary(x)

Object of class STFDF
with Dimensions (s, t, attr): (13, 366, 1)

[[Spatial:]]
Object of class SpatialPolygonsDataFrame

Coordinates:

min  max

x  5.871619 15.03811
y  47.269858 55.05653
Is projected: FALSE
proj4string:
[
+init=epsg:4326 +proj=longlat +ellps=WGS84 +datum=WGS84
+no_defs +towgs84=0,0,0]

Data attributes:

<table>
<thead>
<tr>
<th>ID_0</th>
<th>ISO</th>
<th>NAME_0</th>
<th>ID_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min.</td>
<td>:60</td>
<td>DEU:13 Germany:13 Min. :753.0</td>
<td></td>
</tr>
<tr>
<td>1st Qu.:60</td>
<td>1st Qu.:756.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Median :60</td>
<td>Median :761.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean  :60</td>
<td>Mean  :760.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3rd Qu.:60</td>
<td>3rd Qu.:764.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max.  :60</td>
<td>Max.  :768.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NAME_1</th>
<th>VARNAME_1</th>
<th>NL_NAME_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length:13</td>
<td>Bavaria</td>
<td>:1 NA's:13</td>
</tr>
<tr>
<td>Class :character</td>
<td>Hesse</td>
<td>:1</td>
</tr>
<tr>
<td>Mode :character</td>
<td>Lower Saxony</td>
<td>:1</td>
</tr>
<tr>
<td>Mecklenburg-West Pomerania:1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>North Rhine-Westphalia:1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Other):7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NA's</td>
<td>:5</td>
<td></td>
</tr>
<tr>
<td>HASC_1</td>
<td>CC_1</td>
<td>TYPE_1</td>
</tr>
<tr>
<td>DE.BR</td>
<td>:1</td>
<td></td>
</tr>
<tr>
<td>DE.BW</td>
<td>:1</td>
<td></td>
</tr>
<tr>
<td>DE.BY</td>
<td>:1</td>
<td></td>
</tr>
<tr>
<td>DE.HE</td>
<td>:1</td>
<td></td>
</tr>
<tr>
<td>DE.MV</td>
<td>:1</td>
<td></td>
</tr>
<tr>
<td>(Other):7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REMARKS_1</td>
<td>Shape_Leng</td>
<td>Shape_Area</td>
</tr>
<tr>
<td>NA's:13</td>
<td>Min. : 2.631</td>
<td>Min. :0.1172</td>
</tr>
<tr>
<td>1st Qu.:14.529</td>
<td>1st Qu.:2.1541</td>
<td></td>
</tr>
<tr>
<td>Median :16.891</td>
<td>Median :2.6645</td>
<td></td>
</tr>
<tr>
<td>Mean  :18.068</td>
<td>Mean  :3.3126</td>
<td></td>
</tr>
<tr>
<td>3rd Qu.:24.519</td>
<td>3rd Qu.:4.3832</td>
<td></td>
</tr>
<tr>
<td>Max.  :32.255</td>
<td>Max.  :8.6561</td>
<td></td>
</tr>
</tbody>
</table>
### Temporal aggregation

To aggregate **completely** over time, we can coerce the data to a matrix and apply a function to the columns:

\[
\begin{align*}
R &> x = \text{as(rural[,"2008"], "xts")} \\
R &> \text{apply}(x, 2, \text{mean}, \text{na.rm=}\text{TRUE})[1:5] \\
\text{DESH001} &\quad \text{DENI063} \quad \text{DEUB038} \quad \text{DEBE056} \quad \text{DEBE062} \\
\text{NaN} &\quad 18.41594 \quad \text{NaN} \quad 20.76446 \quad \text{NaN}
\end{align*}
\]

Aggregating values **temporally** is done by passing a character string or a function to the by argument. For monthly data, we will first select those stations that have measured (non-NA) values in 2008,

\[
\begin{align*}
R &> \text{sel} = \text{which(!apply(as(rural[,"2008"], "xts"), 2, \text{function}(x) \text{all(is.na(x))})}) \\
R &> x = \text{aggregate(rural[sel,"2008"], "month", mean, na.rm=}\text{TRUE}) \\
R &> \text{stplot}(x, \text{mode} = \text{"tp"})
\end{align*}
\]
Figure 1: Daily PM10 values, aggregated (averaged) over states
Figure 2: Time series plot of daily rural background PM10, averaged over Germany
Figure 3: Monthly averaged PM10 values, for those rural background stations in Germany having measured values.
shown in figure 3

The strings that can be passed are e.g. "year", but also "3 days". See ?cut.Date for possible values. Aggregation using this way is only possible if the time index is of class Date or POSIXct.

An alternative is to provide a function for temporal aggregation. The function as.yearqtr from package zoo transforms dates to quarters, and hence allows aggregation to quarterly values, in this example medians:

```r
R> library(zoo)
R> x = aggregate(rural[sel,"2005::2011"], as.yearqtr, median, na.rm=TRUE)
R> stplot(x, mode = "tp")
```

shown in figure 4. Aggregating to monthly values is obtained by function as.yearmon, aggregating to years by creating the function

```r
R> as.year <- function(x) as.numeric(floor(as.yearmon(x)))
```

Further information can be found in ?aggregate.zoo, which is the function used to do the processing.

### 4.4 Spatio-temporal aggregation

Aggregation over spatio-temporal volumes can be done by passing an object inheriting from ST to the by argument:

```r
R> DE.years = STF(DE, as.Date(c("2008-01-01", "2009-01-01")))
R> aggregate(rural[,"2008::2009"], DE.years, mean, na.rm=TRUE)
```

<table>
<thead>
<tr>
<th>PM10 timeIndex</th>
<th>2008-01-01</th>
<th>2009-01-01</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.56871</td>
<td>15.23961</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

### 4.5 Time intervals

If all data concern time instances (endTime equals time), then time instances are being matched, else overlapping time intervals are being matched. In case intervals are being matched, empty intervals are never matched.

References

- Pebesma, 2012. Map overlay and spatial aggregation in sp. Vignette in package sp, [http://cran.r-project.org/web/packages/sp/vignettes/over.pdf](http://cran.r-project.org/web/packages/sp/vignettes/over.pdf)
Figure 4: PM10 values, averaged to quarterly medians of daily averages