# Spatio-temporal overlay and aggregation

ifgi
Institute for Geoinformatics
University of Münster

Edzer Pebesma

September 25, 2012

**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

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` comments on what paper (visual) overlays are, and discusses the `over` and `aggregate` methods for spatial data.

In the ESRI ArcGIS tutorial, 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:

> `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 (a_ and 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 – *over*lay is different from *under*lay
- either a *visual* or a *numerical* activity.

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

```
R> over(x, geometry(y))
```

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

```
R> over(x, y)
```

retrieves a `data.frame` with `length(x)` rows, where row `i` contains the attributes of `y` at the spatial location of `x[i]`, and NA values if there is no match.

If the relationship is more complex, e.g. a polygon or grid cell `x` containing more than one point of `y`, the command

```
R> over(x, y, returnList = TRUE)
```

returns a list of length `length(x)`, with each list element a numeric vector with all indices if `y` is geometry only, or else a data frame with all attribute table rows of `y` that spatially matches `x[i]`.

# 3 Spatio-temporal overlay with method `over`

Package `spacetime` adds `over` methods to those defined for spatial data in package `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="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="SpatialPoints"
x="SpatialPoints", y="SpatialPointsDataFrame"
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="xts", y="xts"
```

## 3.1 Time intervals or time instances?

When computing the overlay

```
R> 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 considered to be a time interval, or a time instance. Matching time instance is always considered.

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

Both $T$ and $U$ can reflect time *instances* or time *intervals*. In case they reflect time *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 *intervals*, an observation "at" $t_i$ takes place during, or is representatitive 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**

$$\exists t : t_i \leq t < t_{i+1} \wedge u_j \leq t < u_{j+1}$$

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

$$t_{i+1} > u_j \wedge t_i < u_{j+1}$$

where $\wedge$ denotes 'and'.

All these conditions fail for intervals having zero width, 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 $j$, $u_{j+1} - u_j = 0$. For that reason, specifying objects with

```
R> timeIsInterval(x) = TRUE
```

results in a warning if some time intervals have zero width.