Spatio-temporal overlay and aggregation

ifgi
Institute for Geoinformatics
University of Münster

Edzer Pebesma

March 13, 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`

Being loosely defined, we characterize *map overlay* by

- involving at least two maps
- being asymmetric – *overlay* is not *underlay*
- being 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))
```

retrieves an array of `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 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 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.

The `over` methods in package `spacetime` have a boolean argument `timeInterval` which is by default `TRUE` for the cases where `y` derives from class `STF` or `STS`, and `FALSE` otherwise. When `TRUE`, the times of `y` are considered as time intervals, meaning that the times of `x[i,j]` and `y[k,l]` match if the time instant of `x[i,j]` is larger than or equal to the time instant of `y[k,l]`, but less than the next time instant. The time interval length of the last time step is taken to be identical to the last time interval of an object.

Spatio-temporal objects with only one time step cannot be used to determine time intervals.

4 Aggregating spatio-temporal data

The `aggregate` method for a `data.frame` is defined as

```
R> aggregate(x, by, FUN, ..., 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> data(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> x = as(rural[,"2008"], "xts")
R> apply(x, 1, mean, na.rm=TRUE)[1:5]
```

```
2008-01-01 2008-01-02 2008-01-03 2008-01-04 2008-01-05
  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> dim(rural[,"2008"])
```

```
[1]  70 366   1
```

```
R> x = aggregate(rural[,"2008"], DE_NUTS1, mean, na.rm=TRUE)
R> dim(x)
```

```
[1]  13 366   1
```

```
R> summary(x)
```

```
Object of class STFDF
 with Dimensions (s, t, attr): (13, 366, 1)
[[Spatial:]]
                  Length                    Class
                      13 SpatialPolygonsDataFrame
                    Mode
                      S4
[[Temporal:]]
     Index                  ..1
 Min.   :2008-01-01   Min.   :3653
 1st Qu.:2008-04-01   1st Qu.:3744
 Median :2008-07-01   Median :3836
 Mean   :2008-07-01   Mean   :3836
 3rd Qu.:2008-09-30   3rd Qu.:3927
 Max.   :2008-12-31   Max.   :4018
[[Data attributes:]]
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  2.181   9.933  13.750  15.020  18.370  68.750 366.000
```

```
R> stplot(x, mode = "tp")
```

the result of which is shown in figure 1, which was created by

```
R> stplot(x, mode = "tp", par.strip.text = list(cex=.5))
```

An aggregation for all stations selected within a single area is obtained by merging all states:

```
R> library(rgeos)
R> DE = gUnionCascaded(DE_NUTS1)
```

and then aggregating the observations within Germany for each moment in time:

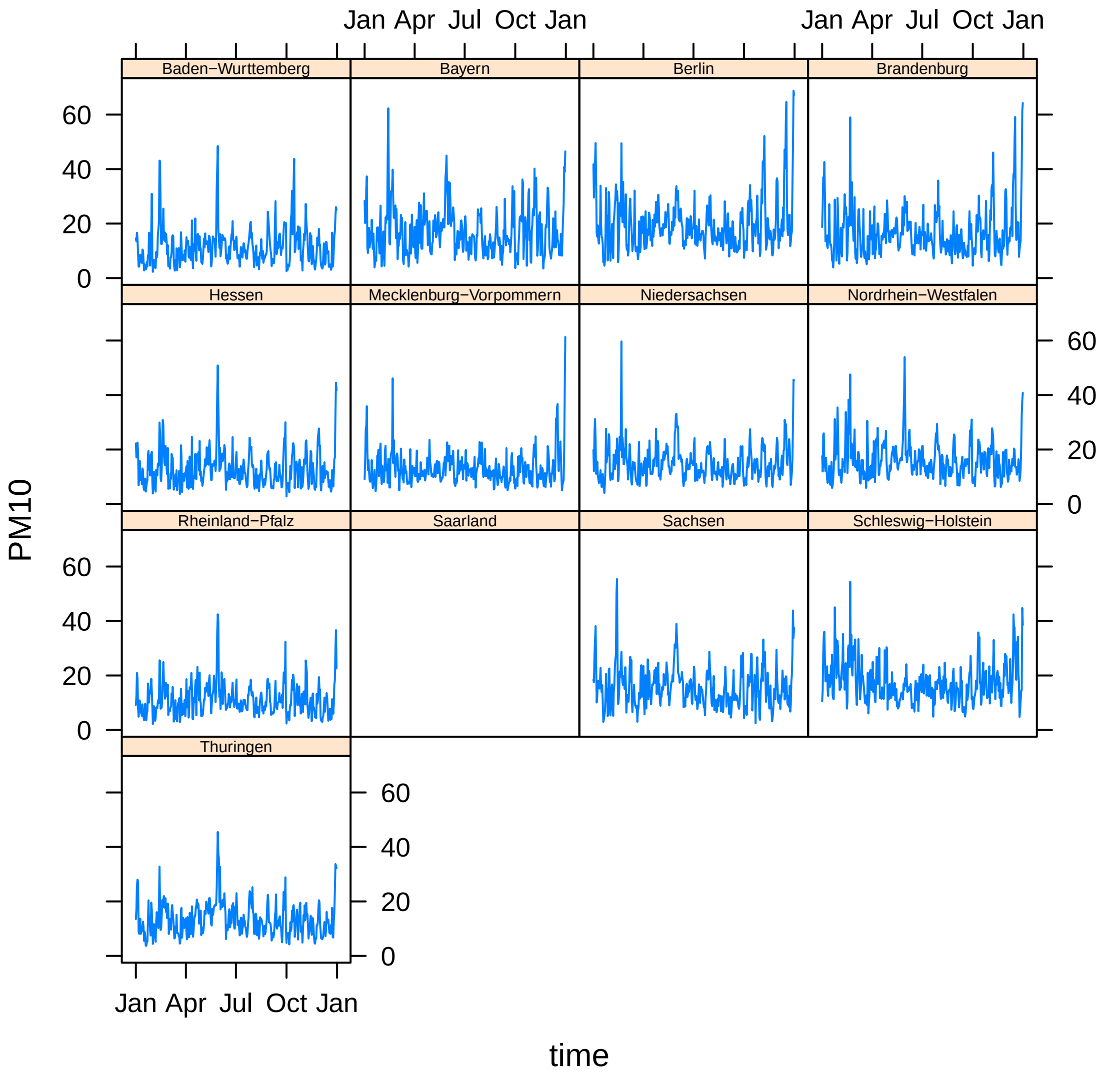

Figure 1: Daily PM10 values, aggregated (averaged) over states

```
R> x = aggregate(rural[,"2008"], DE, mean, na.rm=TRUE)
R> class(x)
```

```
[1] "xts" "zoo"
```

```
R> plot(x)
```

shown in figure 2.

4.3 Temporal aggregation

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

```
R> x = as(rural[,"2008"], "xts")
R> apply(x, 2, mean, na.rm=TRUE)[1:5]
```

```
 DESH001  DENI063  DEUB038  DEBE056  DEBE062
     NaN 18.41594      NaN 20.76446      NaN
```

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,

```
R> sel = which(!apply(as(rural[,"2008"], "xts"), 2, function(x) all(is.na(x))))
R> x = aggregate(rural[sel,"2008"], "month", mean, na.rm=TRUE)
R> stplot(x, mode = "tp")
```

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:

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

shown in figure 4. 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> DE.years = STF(DE, xts(1:2, as.POSIXct(as.Date(c("2008-01-01", "2009-01-01")))))
R> aggregate(rural[,"2008::2009"], DE.years, mean, na.rm=TRUE)
```

```
                    PM10 as.matrix(by@time)
2008-01-01 01:00:00   NA                  1
2009-01-01 01:00:00   NA                  2
```

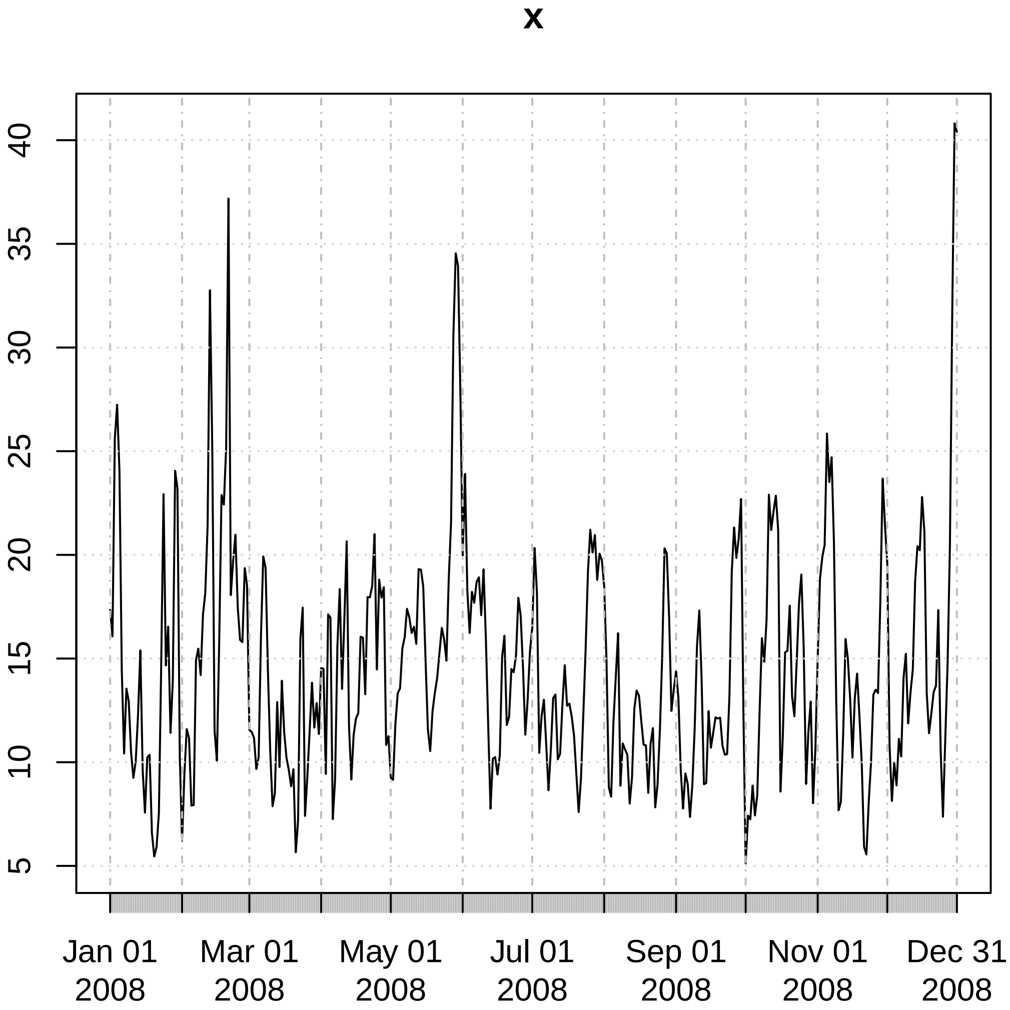

Figure 2: Time series plot of daily rural background PM10, averaged over Germany

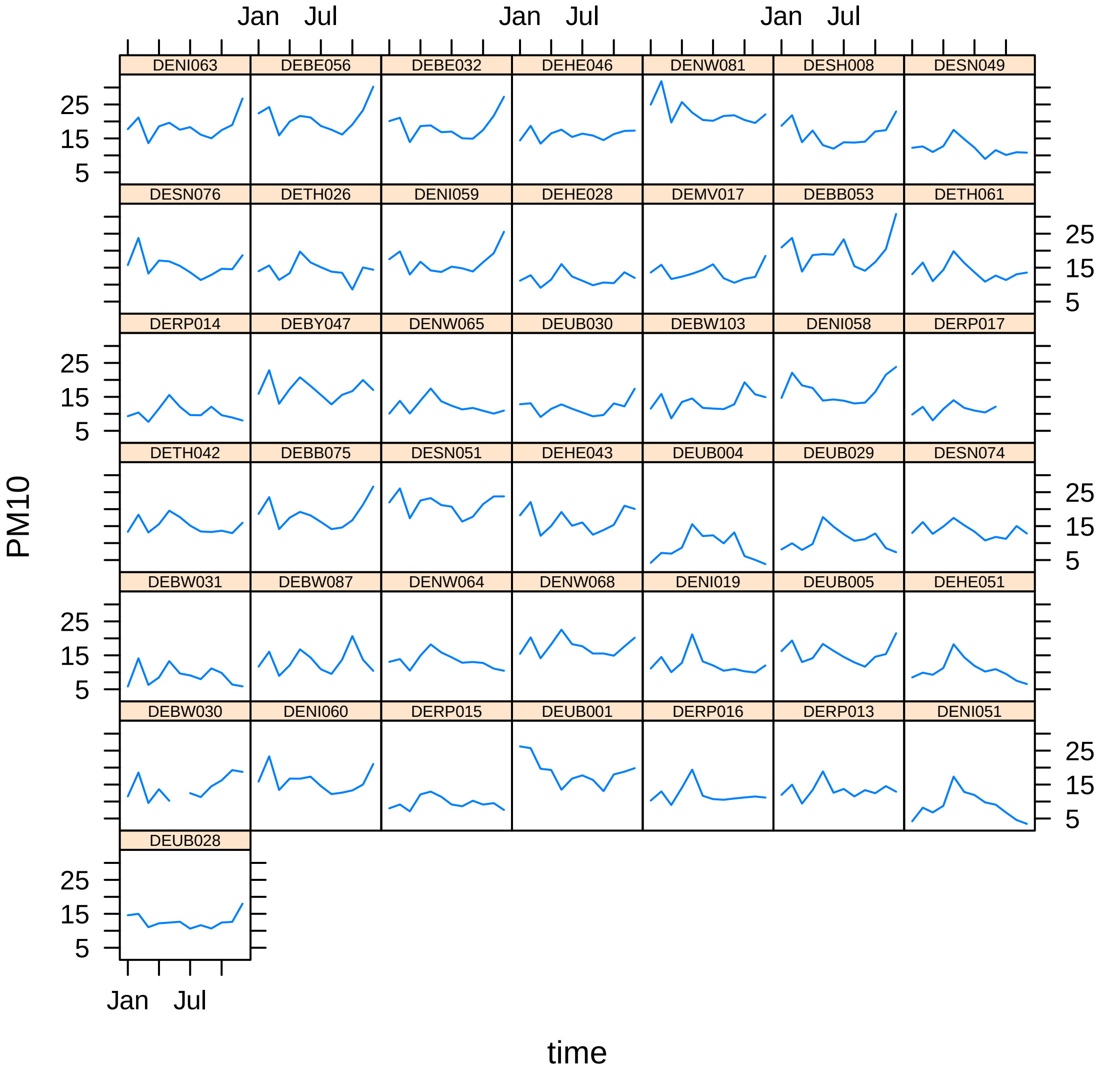

Figure 3: Monthly averaged PM10 values, for those rural background stations in Germany having measured values

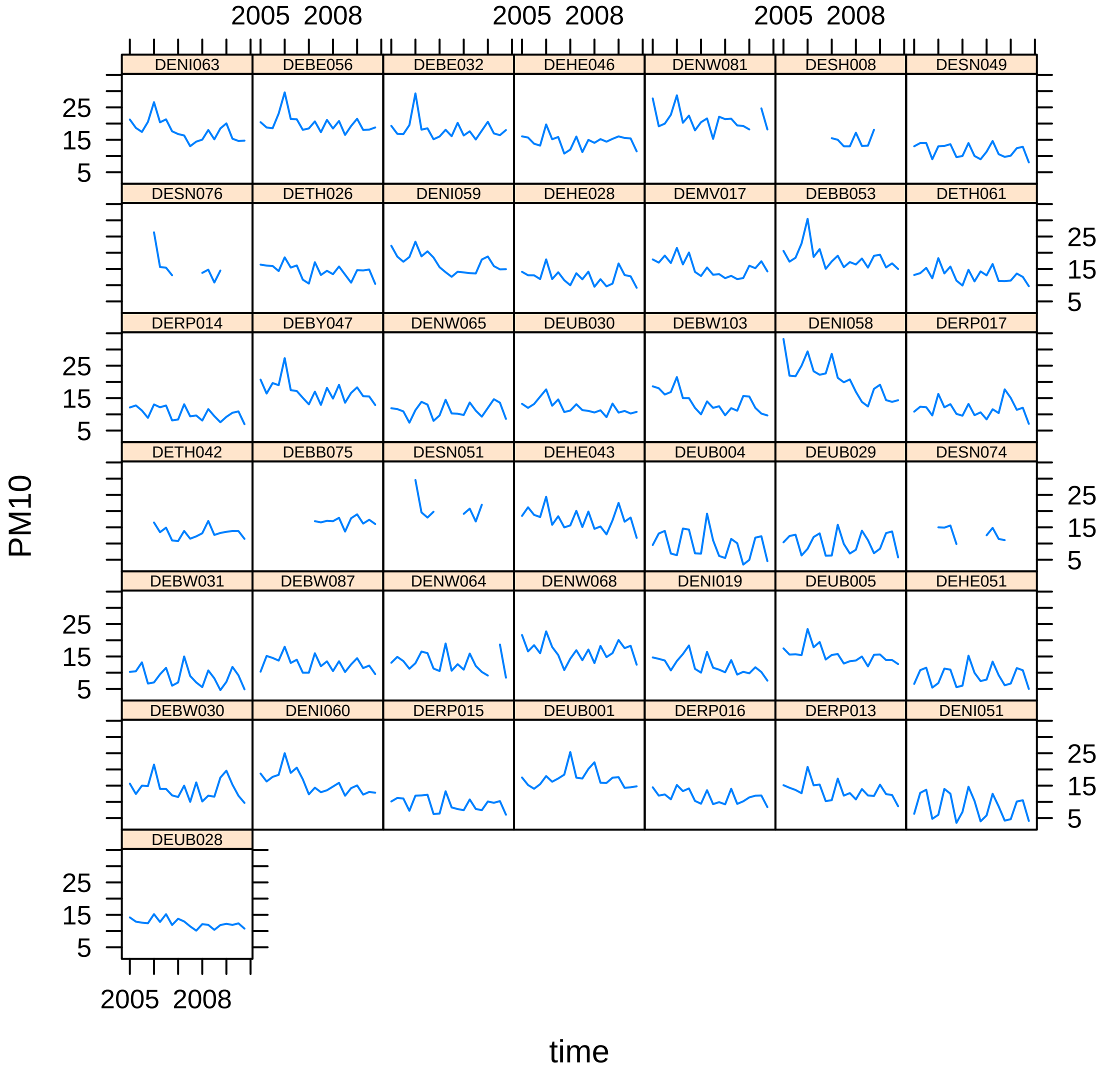

Figure 4: PM10 values, averaged to quarterly medians of daily averages

4.5 Time intervals

Spatial, temporal and spatio-temporal aggregation is all based on the `over` methods. Whether time is considered to be time intervals (for establising whether a space-time point falls, time-wise, inside an interval or coincides with the time point), depends on the defaults for the `over` methods.