uncertainty-enabled model web

Proposal/Contract no.: 248488
Project start: February 1, 2010
Project end: January 31, 2013

UncertWeb

The *Uncert*ainty Enabled Model *Web*

SEVENTH FRAMEWORK PROGRAMME

THEME FP7-ICT-2009-4

ICT for Environmental Services and Climate Change Adaptation

Deliverable3.2

Spatio-temporal Aggregation of Uncertainties

Title of Deliverable	Spatio-temporal aggregation of uncertainties
Deliverable reference number	D3.2
Related WP and Tasks	WP3, Task 3.2
Type of Document	Public
Authors	Christoph Stasch, Phuong Truong, Edzer Pebesma, Gerard Heuvelink
Date	31/10/2011
Version	0.8

Project coordinator

Dr. Dan Cornford

Aston University, United Kingdom

E-mail: d.cornford@aston.ac.uk

http://www.uncertweb.org

RevisionHistory

Version	Date	Changes	Authors
0.0	28/09/2011	Initial template	Christoph Stasch
0.1	06/10/2011	Section 2 and 4 added	Christoph Stasch
0.2	07/10/2011	Section 2.2 added	Phuong Truong
0.3	14/10/2011	Section 5 added	Phuong Truong
0.4	14/10/2011	Section 3.2 added	Christoph Stasch
0.5	20/10/2011	Secion 3 added, corrections	Edzer Pebesma
0.6	21/10/2011	Formatting; references corrected	Christoph Stasch
0.7	26/10/2011	Adding texts, corrections	Gerard Heuvelink
0.8	28/10/2011	Review	Dan Cornford
0.9	03/11/2011	Corrections according to review	Christoph Stasch, Edzer Pebesma, Phuong Truong
1.0	11/11/2011	Corrections and Executive Summary	Gerard Heuvelink

Related task(s):

Task 3.2 Spatio-temporal aggregation of uncertainties[1]

This task develops and implements the tools for aggregation of data, encoded as UncertML and GML, overspace and time. Aggregation over space and/or time is often needed because models produce outputs atmuch finer scales than is of interest to end users and decision makers. For instance, terrestrial greenhouse gasemission or groundwater quality may be predicted for small grid cells or 'points' over instants in time, whereasinternational agreements and European legislation are based on annual averages over entire countries or NUTSregions. This service can also be used in composing (uncertain) model chains where the output of one modelis required at a larger spatial or temporal resolution for input to another model. We will focus on the "upscaling"issue (aggregation). However, we also consider how simple methods for "downscaling" (disaggregation) couldbe included in the service. Aggregation is done numerically using Monte Carlo simulation (e.g. Heuvelink andPebesma 1999), thus allowing maximum flexibility with respect to aggregation domain (regular as well asirregular) and type of aggregation (linear as well as non-linear). The tool will take the form of an UncertWebservice.

Active partners: WU, UOM

[1]Task description is taken from the UncertWeb Description of Work document [UW-DOW 2009]

Legal Notices

Executive Summary

This report documents open sourcetools to support space-time aggregationand disaggregation of environmental variables. The Model Webcouples models that interact via web servicesand whose inputs and outputs may have different scales. Thus, aggregation and/or disaggregation services are needed when the output of one model in the chain is at a different scale than the input of the ensuing model in the chain. Since the UncertWeb project aims to realise an uncertainty-enabled Model Web, the tools presented in this document must accomplish the aggregation and disaggregation in a situation where there is uncertainty about the variables for which a change of scale is required.This is achieved by adopting a Monte Carlo simulation approach, whereby the (dis)aggregation is done repeatedly, once for each realization from the Monte Carlo sample. In this way the (dis)aggregation tools may be built for the deterministic case and can still be used to analyze the effects of aggregation and disaggregation under uncertainty.

Prior to the development of tools for aggregation and disaggregationa requirement analysis was done. For this a questionnaire was prepared and distributed to all use case work packages of UncertWeb. The results of the questionnaire show that a variety of aggregation processes are considered to be useful in the application scenarios. Spatial disaggregation is also required by almost all use cases.

Two tools were developed for aggregation. The *spacetime* package in R provides aggregation functionality that can be directly integrated with existing R code of a model. In order to support standard spatial data types and deploy the spatial aggregation processes for these types, an extension of the 52°North WPS was developed that allows to upload R scripts and publish them as WPS processes. The *Spatio-Temporal Aggregation Service* (STAS) provides spatio-temporal aggregation functionality in UncertWeb as a profile of the Web Processing Service (WPS). Thus STAS provides the standardized Web service interface for integrating aggregation processes in the UncertWeb infrastructure. While the STAS provides a common interface to deploy the spatio-temporal aggregation processes in the Web, it uses the *spacetime* package of R in the backend.

The *krige0* function of the *gstat* package in R was extended with Area to Point kriging to povide a tool for disaggregation. This first required that the statistical methodology of Area to Point kriging was worked out for the purpose of UncertWeb. Similar to the *spacetime* package, this tool could be incorporated as a backend in the STAS, thus providing a web-based service for disaggregation.

Contents

1 Introduction

The “Model Web” can be seen as a dynamic modelling infrastructure composed of loosely coupled modelsthat interact via Web services and are independently developed, managed, and operated [Geller 2009].The UncertWeb project aims to realise an uncertainty-enabled Model Web.In order to achieve this goal existing tools and technologies for the Model Web need to be extended to support processing and communication of uncertainties associated with spatio-temporal information. Both data and associated uncertainties transform under a change of scale and it is therefore necessary to develop and implement methods that accomplish both aggregation and disaggregation under uncertainty. This task and associated deliverable focus on the development and web-based implementation of aggregation methods, while disaggregation methods are only briefly addressed and their implementation as a web service is only explored.

Aggregation of spatio-temporal information is needed in the Model Web for two main reasons that are illustrated in Figure 1. First, when feeding models with observations from various sensor sources or with output from other models, the required spatio-temporal resolution ofthe input data required by the current model might differ from the resolution provided by the sensors or the output of other models. Second, there is a trade-off between the spatio-temporal resolution of data and their uncertainty. Spatio-temporal aggregation may be carried out intentionally to reduce the uncertainty in the data. Usually, the more the data is aggregated, the less uncertainty remains in the data. This happens of course at the cost of spatial and/or temporal detail.In this deliverable, we describe two tools developed in the UncertWeb project that can be used to aggregate data in space and time. These are the *spacetime* package in R and the *Spatio-Temporal Aggregation Service* (STAS). We also present methodology and R code for spatial disaggregation. The *spacetime*R package and disaggregation code are standalone tools that are not integrated in the UncertWeb Service. The integration is established by the STAS service, as explained below.

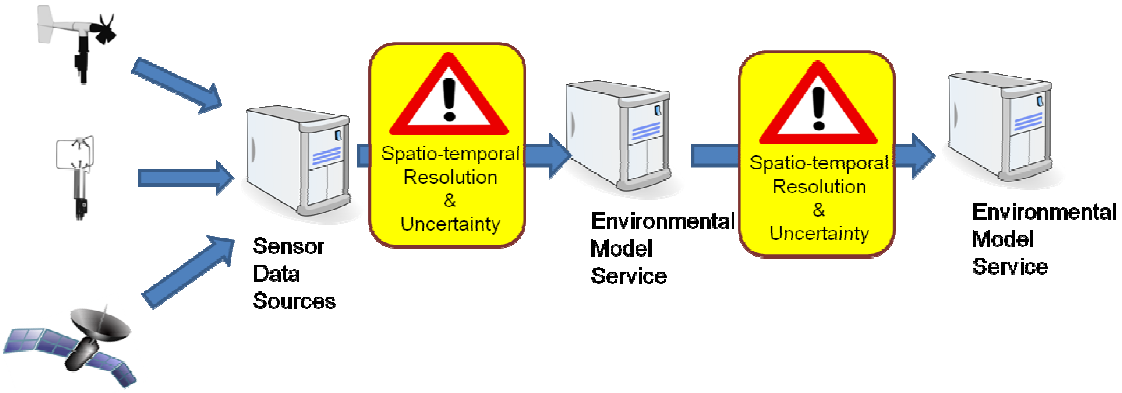


Figure 1: Need for spatio-temporal (dis)aggregation in the Model Web.

The *spacetime*package provides classes andmethods to deal with spatio-temporal data in the R environment. It also includes methods for spatio-temporal aggregation. The package is developed in the UncertWeb project for two reasons. First, the *spacetime* package and its aggregation functionality can be directly integrated in existing R code that implements a certain model, ifthe aggregation needs to be done within a model. Second, while Java or C# provide advantages in implementing Web services based on technologies such as HTTP and XML, it is not the best choice for implementing mathematical functions. Therefore, the *spacetime* package in R can be used at the backend of Web services to implement aggregation

functionality, while Java is used at the frontend of the Web services to expose the aggregation functionality in the Web, combining the benefits of the two languages.

To provide spatio-temporal aggregation functionality in UncertWeb as a Web service, the Spatio-Temporal Aggregation Service (STAS)is defined as a profile of the Web Processing Service (WPS) [Schut 2007].The WPS is defined by the Open Geospatial Consortium (OGC) as a standardized Web service interface for integrating processing functionality in Web based spatial data infrastructures. The STAS provides the standardizedWeb service interface for integrating aggregation processes in the UncertWeb infrastructure. This infrastructure consists of different Web services encapsulating models and data sources. While the STAS provides a common interface to deploy the spatio-temporal aggregation processes in the Web, an implementation of the STAS does not need to implement all different aggregation functions by itself, but can use the *spacetime* package of R in the backend.

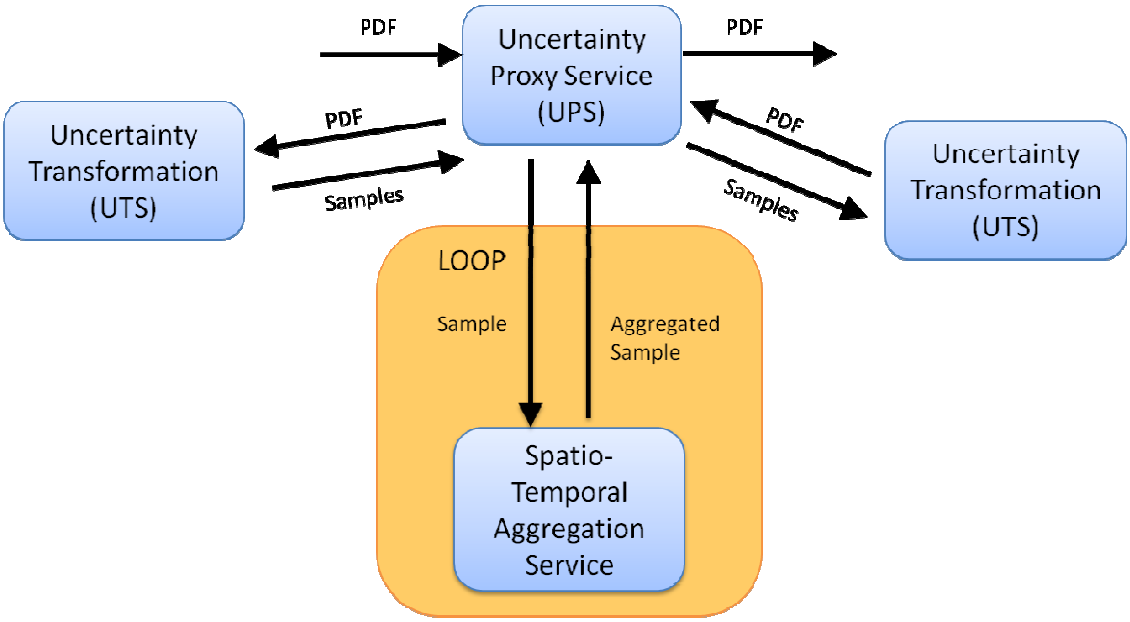


Figure 2: Uncertainty-enabled spatio-temporal aggregation. The Uncertainty Transformation Service (UTS) is used to transform Probability Distribution Functions (PDF) in Samples and the Uncertainty Proxy Service (UPS) is used to run the uncertainty enabled aggregation. The Spatio-Temporal Aggregation Service does not need to be uncertainty enabled.

Handling uncertainty in aggregation

To allow for maximum flexibility in terms of probability distributions that are supported, the propagation of uncertainty in spatio-temporal aggregation is done numerically using Monte Carlo simulation (e.g. [Heuvelink 1999]). Toalso allow maximum flexibility with respect to aggregation domain (regular as well asirregular) and type of aggregation (linear as well as non-linear), different aggregation predicates and aggregation functions are allowed.Because use is made of a Monte Carlo simulation approach, the aggregation tools implemented in the *spacetime* package and the STAS do not need to explicitly address uncertainty. Instead these tools are applied repeatedly, as many times as there are Monte Carlo simulation runs. Next the Monte Carlo set of aggregated simulations is used to assess the uncertainty in the aggregated output.

For running the MonteCarlo simulation, common tools as developed and implemented in the UncertWeb project can be used as shown in Figure 2.The Uncertainty Transformation Service (UTS) allows to generate samples from probability distributions and the Uncertainty Proxy Service (UPS) allows executing the aggregation for each sample. Finally, the aggregation results can also be represented by a probability transformation using the UTS again to create a probability distribution from the aggregated simulations.

1.1 Structure of Deliverable

This deliverable is organized as follows. Chapter 2 describes the results of a requirements analysis for spatio-temporal aggregation in the application scenarios of the UncertWeb project. Each of the use cases of the project was asked to return the questionnaire provided in Appendix A and chapter 2 summarizes the results, both for the requirements for aggregation and disaggregation.Chapter 3 introduces the *spacetime* package for R, a free and open source environment for computational research, primary statistics, and graphics. It includes methods for spatio-temporal aggregation and provides tools needed to implement the aggregation methods. Chapter 4 describes the *Spatio-temporal Aggregation Service* (STAS) that integrates the aggregation methods needed in the UncertWeb service architecture.Chapter 5 has a more statistical character and describes the implementation of a particular disaggregation method, Area-to-Point Kriging, in R. Disaggregation is statistically more challenging than aggregation and emphasis is therefore on the statistical methodology and how this can be implemented in software, but implementation as a web-service is beyond the aims of this task. Chapter 6 presents conclusions. The remainder of this introductory chapter gives definitions and a list of acronyms and abbreviations.

1.2 Definitions

To ensure a standard set of terms the following definitions are adhered to throughout the document.The definitions are taken from [Bierkens 2000] and [Jeong 2004].

Aggregation: An aggregation process computes a value (an aggregate) for a group of attribute values by means of an aggregation function. The attribute values are grouped by a partitioning predicate.

Aggregate : Aggregated value.

Aggregation Function: Function used to compute the aggregate.

Disaggregation: Opposite of aggregation, i.e. the computation of individual values for group members from an aggregate for the group as a whole.

Partitioning Predicate: Predicate used to group objects before aggregating the values attached to these objects.

Spatio-temporal Aggregation: A spatio-temporal aggregation process combines objects in space and time by spatial and/or temporal partitioning predicates and computes aggregates for certain attribute values attached to these objects by an aggregation function.

TimeInstant: Point in time.

TimeInterval: Duration in time with TimeInstants as start and end point of duration.

TimeCoverage: Tessellation of a temporal extent into TimeIntervals with equal duration and without gaps between the TimeIntervals.

1.3 Abbreviations

FTP	File Transfer Protocol
GDAL	Geographic Data Abstraction Layer
GML	Geography Markup Language
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
NetCDF	Network Common Data Format
OGC	Open Geospatial Consortium
O&M	Observations & Measurements
R	Statistical software framework R
SensorML	Sensor Model Language
SOS	Sensor Observation Service
STAS	Spatio-Temporal Aggregation Service
UncertML	Uncertainty Markup Language
U-O&M	UncertWeb profile of Observations&Measurements
U-NetCDF	UncertWeb profile of NetCDF
WPS	Web Processing Service
WPS4R	Web Processing Service for R

2 Aggregation Requirements

As there are many different ways to aggregate spatio-temporal information, we created a questionnaire about spatio-temporal aggregationto gather specific requirements for spatio-temporal aggregation within the context of the UncertWeb project (i.e. the use cases). The questionnaire is attached to this document in Appendix A.The questions were grouped by aggregation and disaggregation. In the following, we summarize the results of the responses provided by the partners implementing the application scenarios.

2.1 Requirements for Aggregation

This subsection describes the requirements for aggregation. We start with the grouping predicates that describe how the spatial and/or temporal geometry types are changed in spatio-temporal aggregation. The required spatial grouping predicates forthe different spatial geometry types in the UncertWeb project are shown in Table 1. Points can be aggregated to Lines, Polygons and Grid Cells. Lines can be aggregated to Grid Cells, whereas Polygons only are aggregated to larger Polygons. Grid Cells need to be aggregated to (larger) Grid Cells or Polygons.

Table 1: Required spatial grouping predicates for different spatial geometry types.

Type of spatial geometry in the original data	**Spatial grouping predicate**
Point	Line, Polygon, Grid Cell
Line	Grid Cell
Polygon	Polygon
Grid Cell	Grid Cell, Polygon

Similarly, to the spatial grouping predicates, the required temporal grouping predicates are listed for each temporal geometry type in Table 2. TimeInstants need to be aggregated to TimeIntervals or TimeCoverages, TimeIntervals to (larger) TimeIntervals or TimeCoverages, and TimeCoverages to (coarser) TimeCoverages.

Table 2: Required temporal grouping predicates for each temporal geometry type.

Type of temporal geometry in the original data	**Temporal grouping predicate**
TimeInstant	TimeInterval, TimeCoverage
TimeInterval	TimeInterval, TimeCoverage
TimeCoverage	TimeCoverage

While the spatial and temporal grouping predicates define how the data are grouped spatially and/or temporally before applying a certain aggregation function, the type of the aggregation function needs to be defined as well. In most of the application scenarios, simple linear aggregation functions are used, e.g. the mean or the sum of values in an area or time period. However, in some cases, also non-linear aggregation functions may be needed, such as the median or any other percentile, and computation of the fraction of the area or time period where a certain threshold is exceeded.

Regarding the uncertainty in spatial aggregates, there is no clear requirement. Some of the application partners do not need any uncertainty information in the aggregate, while others

would like to have an uncertainty estimate of the aggregated data (e.g. line to grid emissions). The uncertainty in the results of the aggregate must be based on the uncertainty of the original data, taking into account that the original data may be -to some degree- correlated in space and time. As we are using Monte Carlo simulation to propagate uncertainty during aggregation, some of the application partners are planning to use the Elicitator developed in UncertWeb (i.e. see deliverable D3.1) to elicitthe uncertainty in the original (non-aggregated) data and next use the results of the elicitation to trace uncertainty propagation to the aggregated data.

The partners involved in the application scenarios have also been asked whether additional metadata are needed in the aggregated data.Provenance information is considered to be useful. This should include, for example, the aggregation function used, a reference to the original data, or information about the number of data items as well as the variability of the original data.

In summary, there are a variety of different aggregation processes that are considered to be useful in the application scenarios. Thus, the tools for implementing aggregation processes in the UncertWeb project need to be sufficiently generic to provide a variety of different aggregation methods.

2.2 Requirements for Disaggregation

We also inquired whether the four use cases of the project (i.e. the biodiversity case of WP4, the land-use and agricultural policy case of WP5, the air quality case of WP6 and the activity–travel patterns case of WP7) whether these have a need for spatial and temporal disaggregation of model inputs and outputs.The proposed types of spatial – temporal input data and the corresponding disaggregated types are listed in Tables 3 and 4.

Table 3: Requested spatial disaggregation predicates for different spatial geometry types.

Type of spatial geometry in the original data	Type of Spatially disaggregated data
Line	Point
Polygon	Line, Point, Grid Cell, smaller Polygon
Grid	Point, Grid Cell

None of the work packages require spatio-temporal disaggregation. There is only one requirement from WP6 for temporal disaggregation, which requires disaggregation of daily-based data to hourly-based data with uncertainty quantification. Most requirements focus on disaggregation of spatial data, in particular disaggregationfrom large grid cells to smaller grid cells, from large grid cells to smaller polygons and from large polygons to smaller grid cells.

Table 4: Requested temporal disaggregation predicates for different temporal periods.

Type of temporal period in the original data	Type of temporallydisaggregated period
TimeInterval	TimeInstant, TimeInterval, TimeCoverage
TimeCoverage	TimeInstant, TimeCoverage

In detail, WP5 requires disaggregation from coarse grid cells of future climate prediction to field level polygons, which can maintain the original value types. If the disaggregation tool can be simply implemented for uncertainty quantification, WP5 also requests information about the uncertainty at the disaggregated grid cell level. WP6 requires disaggregation from original polygon data, which is the total consumption of fuel for a whole city, to a grid-based representation. This application also requests uncertainty information about the disaggregated

grid cell values and their autocorrelation in space. WP7 did not state specifically which data input need to be disaggregated but does indicate the need for a tool for spatial disaggregation. The same holds for WP4, for which it was not explicitly clear whether disaggregation is needed. IfWP4 requires disaggregation then this has to be done such that spatial autocorrelation is taken into account during the disaggregation process.

In summary, almost all partners have requirements for spatial disaggregation for their application scenarios. Hence a tool for spatial disaggregation is apparently needed. There are no general requirements or specific requirements for temporal or spatial – temporal disaggregation from all four project partners.

3 The spacetime Package in R

This chapterdescribes the first tool delivered under Task 3.2. The *spacetime* package in R is a requirement for spatio-temporal aggregation in R. Sections 3.1 to 3.3 introduce R, classes for space-time data and the aggregation function implemented in the *spacetime* package. In order to deploy common scripts for spatio-temporal aggregation in the Web, the Web Processing Service for R (WPS4R) is used, which is described in section 3.4. As WPS4R currently only supports the standard formats for spatial information, time is only supported to a limited extent and the WPS inputs might be very specific and differ from aggregation process to aggregation process. Thus, mechanisms for supporting the spatio-temporal data formats in UncertWeb and to integrate these with the Web services (i.e. the STAS service) are required too. These are described in chapter4.

3.1 About R

R is an open source environment for statistical computing and graphics. It implements the language S, which is an object-oriented language (but also a functional language) designed for analysing data. R is extendible by add-on packages. The R project provides not only source code and binary distributions of R, but also a network of mirrored web sites that distribute extension packages, called CRAN. CRAN takes over the burden to compile packages for different software platforms (Unix, Windows, MacOS).

One package on CRAN, called *sp* [Bivand 2008], extends R to deal with spatial data (points, lines, polygons, and grids). Since its release in 2005, many other packages (more than 100) now depend, directly or indirectly, on the classes and methods provided by this package. In total, CRAN hosts over 3300 extension packages that strongly make use of each other. As such, one could consider R as an early example of a successful model web, where the interoperability is not achieved through communication through web services, but by adopting a single system (R) to handle the data. The reasons for its success are the trust in open source software, the availability of a trusted platform (CRAN) to provide, exchange and share source code, and a large number of mailing lists and forums to communicate with other users and developers.

Without any doubt, today R is the largest open source statistical project worldwide. It is often said that the more people use, verify, and look at source code, the better it gets. Instead of re-implementing a lot of statistical functions, such as distribution functions and functions to sample from them from scratch, we try to reuse where possible existing code in R in UncertWeb, because it is tested, trusted, and maintained.

With the requirements of UncertWeb in mind, and also with the availability of reusable functions, classes and methods in R, and the potential of enriching R with useful functionality, we developed an R package called *spacetime* with classes and methods for spatio-temporal data [Pebesma 2011]. The package is available from CRAN, and has been used already in several case studies outside UncertWeb.

3.2 Classes for spatial data, and for spatio-temporal data

The classes available for spatial data in R include points, polygons, lines and grids. Lines and polygons resemble the simple feature specification [Herring 2011]; points are simple points, no multipoints or multigeometries are supported. Grids reflect simple raster or grid data

structures. Objects can contain only geometry, or can be extended with attributes of any kind (numeric, logical, factor/categorical, date, character).

Classes for spatiotemporal data specify collections of points in a space-time layout. The spatial "locations" refer to any of the Spatial types (points, lines, polygons, grids). The temporal points can be expressed in any of the Date or Time formats supported by R (Date, dateTime, POSIXct, yearQuarter, yearMonth). Attributes of any kind can be stored for each of the space-time points. Time points need not be regularly spaced, but typically are.

Space-time layouts implemented in the R spacetime package are:

- completely regular, meaning that for each location a full set of time points is available (alternatively: for each time instant the full set of spatial locations is available),
- sparse regular, meaning that for a completely regular layout a possibly incomplete subset is stored, or
- irregular, meaning that space-time points have no apparent organization in space-time.

A number of methods are available for the spacetime classes. They include (i) construction of classes, (ii) coercion of objects into Spatial objects, into time series objects, or into simple tables, (iii) selection of records by (space or time) record number, by spatial area, by temporal period, or by attribute value, and (iv) selection, assignment or replacement of attributes. In addition, summary methods are provided.

3.3 Analysis methods for spatial and spatio-temporal data

Analysis methods developed in packages *sp* (for spatial data) and *spacetime* (for spatio-temporal data) include two methods that are of particular relevance to this deliverable. The first is method "over", which overlays two (spatial or spatio-temporal) geometries, and can generate an aggregation predicate. The second is "aggregate", which aggregates attributes for spatial or spatio-temporal objects over a second geometry, given an aggregation function (that can be any R function).

3.3.1 Aggregation predicates: method "over"

Aggregation predicates, the grouping of observations to be aggregated by a particular aggregation function, are obtained by a (spatial, temporal, or spatio-temporal) overlay. This is implemented by method `over`. In essence, the function call

```
over(x,y)
```

returns an object with the same number of locations of x, containing the index or set of indices of y (spatially, temporally, or spatio-temporally) corresponding with x. It should be noted that the spatial component of x and y can be any of (points, lines, polygons and grids), and correspondence implies intersection (two lines), coinciding (two points) including (point in polygon), or overlapping (two polygons).

As R is a language for data analysis, it already contains an `aggregate` method for basic types such as `data.frames` (data tables), matrices and (higher-dimensional) arrays. This function can be extended by methods for new classes. The generic definition is

```
aggregate(x, by, FUN, …, simplify=TRUE)
```

where `x` is the object to aggregate, by the aggregation predicate, `FUN` the aggregation function to be applied to the attribute values of object x, and `simplify` a Boolean indicating whether the return argument should be simplified, when possible. Spatial objects can take a spatial object as predicate, which allows the aggregation of e.g. points within larger grid cells, small grid cells in large grid cells, points in polygons, etc. Spatio-temporal objects can be aggregated using a temporal predicate, a spatial predicate, or a spatio-temporal predicate. The temporal predicate can be a sequence of times (interpreted as time intervals), or a function that generates a predicate from the time instances (such as `as.yearqtr`, which computes in which quarter a particular date falls, for aggregation to quarterly values). Spatial predicates can be, as for spatial data, spatial objects, and result in no aggregation over time. Spatio-temporal predicates provide the means to aggregate over spatio-temporal geometries. This allows, for instance, aggregation in one step from daily values at measurement stations to yearly values (e.g. medians) for countries. Another possibility is the aggregation of a regular spatio-temporal grid layout over an irregular spatio-temporal trajectory.

Appendices B and Cprovide full documentation (with examples and illustrations) of the functionality of spatial and spatio-temporal overlay and aggregation in R.

3.4 WPS4R

The Web Processing Service for R (WPS4R) allows the deployment of R scripts as processes in a WPS. The workflow of uploading the R scripts consists of three steps: First, the R scripts need to be annotated in order to declare inputs and outputs for the WPS. Second, the script needs to be uploaded to the WPS. Finally, the process can be executed as a usual WPS process and integrated in SDIs. The three steps are described in more detail below.

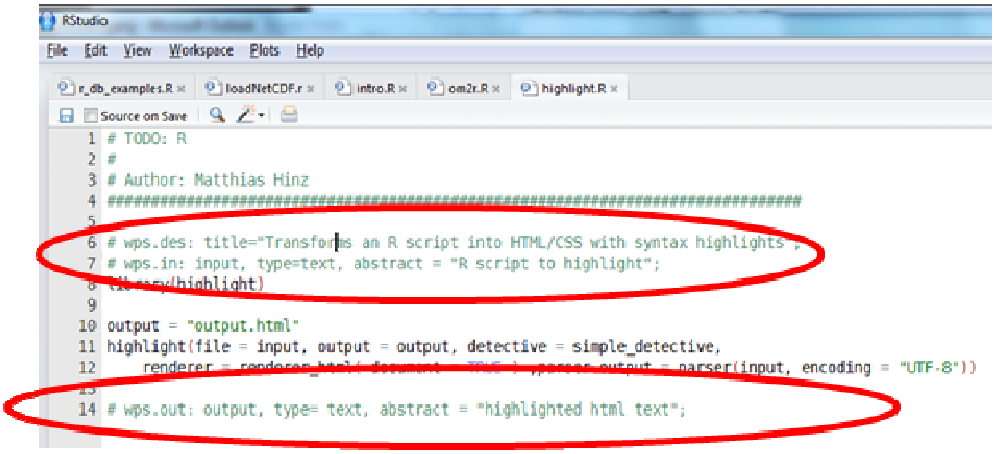


Figure 3: Annotated R script. The annotation are shown in the red ellipses.

Figure 3 shows an example of an annotated R script. The annotations are used to define a description of the process implemented in the script and to define the inputs and outputs of the process.Table 5 shows the identifiers of the annotations, the schema how the values have to be provided and a short description. The `wps.des` annotation provides the identifier, a title of the process and an abstract containing a short textual description of the process. Only one description annotation can be providedper R script. The `wps.in` annotation can be used to declare any inputs of the WPS process in the script. Thus, there can be more than one `wps.in` annotation. Besides the `identifier`, `type`, `title` and `abstract`, also a

default value as well as information about the cardinality can be provided (`value`, `minOccurs` and `maxOccurs` are optional). The `wps.out` annotation is used to define process outputs.The types currently supported are basic data types such as `double` or `string` as well as common spatial input types such as shapefile for feature data,`geotiff` for raster data or `jpeg` for image data. The `title` and `abstract` annotations refer to the title and abstract in the WPS process descriptions.

Table 5: Supported annotations in R script.

Annotation Identifier	Value Schema	Description
`wps.des:`	id, title, abstract	description of the process
`wps.in:`	id, type, title, abstract, value = null, minOccurs = 1, maxOccurs = 1	declaration of process input
`wps.out:`	id, type, title, abstract	declaration of process output

Once a script has been implemented and the annotations have been added, the script can be deployed in the WPS4R. Therefore, the Web Admin Console of the 52°North WPS has been extended by an interface for uploading an R script. Deployed processes are then listed in the Web Admin Console (Figure 4). Afterwards, the processes are available as WPS processes and can be used by common WPS clients and integrated in spatial data infrastructures.

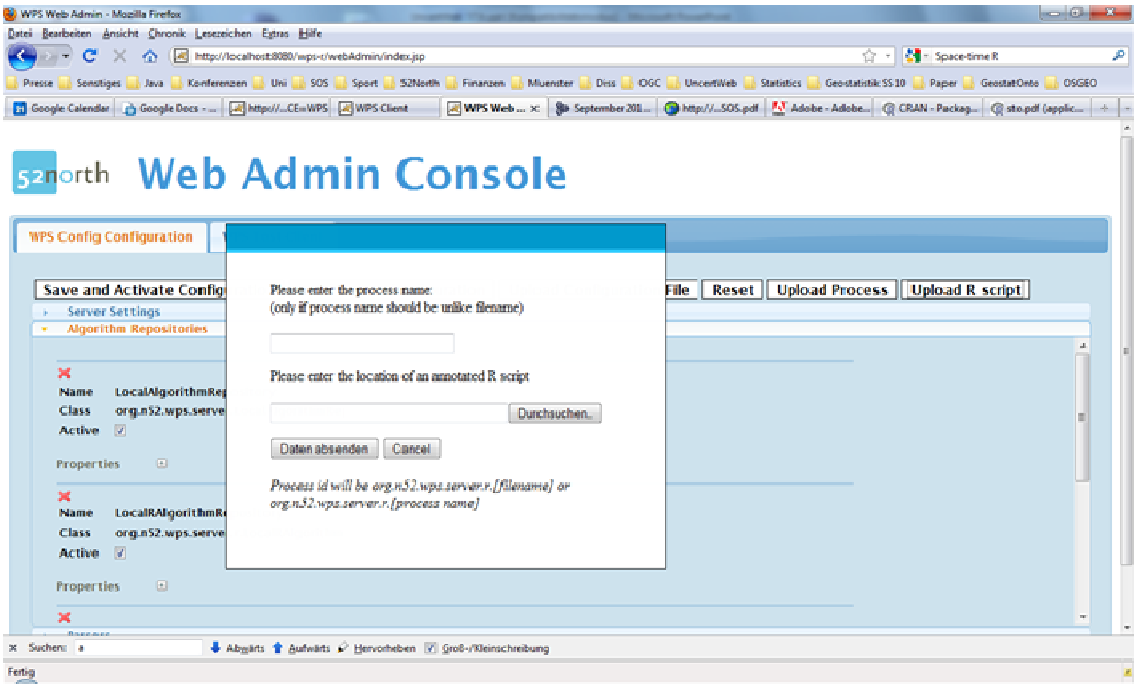


Figure 4: Web Admin Console of WPS4R.

In UncertWeb, the WPS4R can be used to deploy R scripts implementing aggregation processes for standard spatial data formats, e.g. GeoTIFF. However, as there is no common standardized exchange format for spatio-temporal data in the Web yet and the WPS4R is using the standard tools for importing and exporting data, another service is needed to integrate spatio-temporal aggregation in UncertWeb resulting in different signatures for the aggregation processes. Thus, in order to support the exchange formats for spatio-temporal data in the UncertWeb project and to provide common pre-defined aggregation process signatures.

4 Spatio-temporal Aggregation Service

Having introduced the *spacetime* package for R in the previous chapter and having explained how aggregation implemented in R can be exposed in a web service using WPS4R, this chapter explains how the aggregation methods can be integrated in the UncertWeb architecture. For this we have defined the *Spatio-Temporal Aggregation Service* (STAS) as a profile of the WPS [Stasch 2011]. We first illustrate how the STAS utilizes the data models defined in UncertWeb and nextdescribe the service interface of the STAS. After that the implementation of the STAS based upon the 52°North WPS implementation is described. We conclude this chapter with a case study for the aggregation of air pollutant measurements in Germany.

4.1 Information Models used in the STAS

For aggregation of spatio-temporal information in the (Model) Web, spatio-temporal data models are needed that can be used to exchange spatio-temporal data at different aggregation levels in the Web. Furthermore, as stated in the requirements (Section 2), additional metadata about the aggregation processes as well as provenance information is needed in the aggregated data.In the UncertWeb project, three main information models are used to exchange data between model services. These are UncertML [UW-D1.2 2011] for exchanging uncertainty information, U-O&M for exchanging spatio-temporal information with uncertainties attached to spatial vector geometries [UW-D1.1 2011] and U-NetCDF for exchanging gridded spatio-temporal information with uncertainties[UW-D2.2 2011].For UncertML, no changes are needed, as the spatio-temporal geometries and metadata can be provided in U-O&M and U-NetCDF.

As U-O&M supports the different geometry types Points, Lines, Polygons and Grids, the model can be used at different spatial aggregation levels. Similarly, the temporal geometries can be instants or periods in time allowing the use of U-O&M at different temporal aggregation levels. To provide aggregation metadata, we have extended the U-O&M model to allow for provenance information. As there is no pre-defined information element for provenance in NetCDF, we have not extended the U-NetCDF. In case metadata about e.g. provenance or the aggregation process applied is needed when aggregating NetCDF data, O&M can be used to provide the metadata, whereas NetCDF is used to provide the non-aggregated and aggregated gridded uncertain spatio-temporal information (see example inFigure 6 below).

A UML diagram of the extended U-O&M model is shown in Figure 5. The blue boxes show the elements of the basic O&M model, the yellow boxes show restrictions of the U-O&M profile and the grey elements are restrictions/extensions for aggregation. The basic O&M model provides elements for the time of an observation (`phenomenonTime`), the procedure that has created an observation (`procedure`), the variable that is observed (`observedProperty`) and the result value which is not further restricted (`result`). The U-O&M profile restricts the possibilities of procedures, result types and featureOfInterests. For example, an observation can provide an uncertainty value (e.g. a normal distribution as probability distribution function) for a sampling point gathered by a system of sensors.

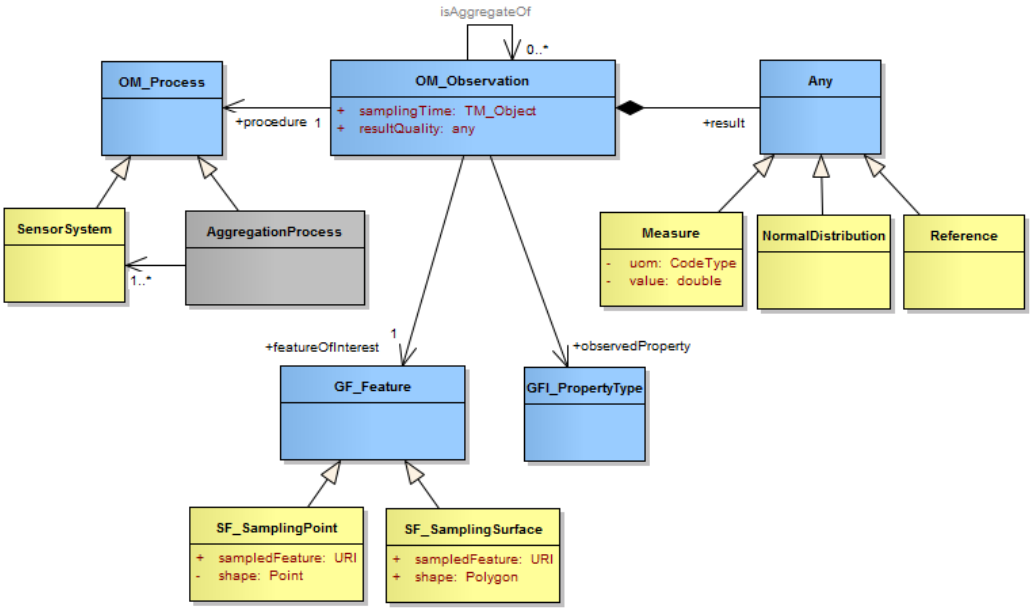


Figure 5: UML diagram of the extended U-O&M model.

The two elements which have been added to the basic model for aggregation are the optional`isAggregateOf` relationship pointing to at least two single observations or to a collection of observations containing more than two observations. Furthermore, the `procedure`element of an observation that describes how an observation value has been computed can now be an `AggregationProcess`. In case the aggregated data is a NetCDF file, the reference result can carry a reference to a WCS providing the aggregated data in a NetCDF file, while the O&M provides metadata about the aggregation process and the original dataset as shown in Figure 6.

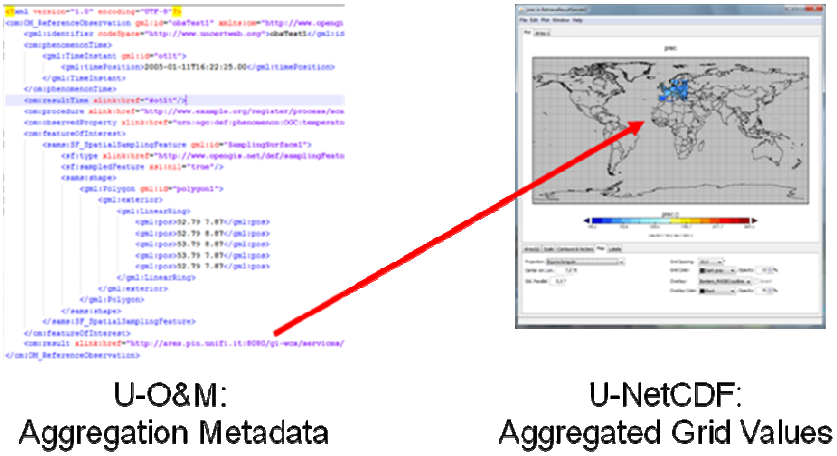


Figure 6: Encoding of U-O&M for aggregated NetCDF data.

The model used for the `AggregationProcess` is shown in Figure 7. In an aggregation process, data are grouped by a partitioning predicate before applying a certain aggregation function to its variable values (see Section 1.2). We currently consider the partitioning predicates to be spatial and/or temporal. Thus, observations are grouped spatially and/or temporally and aggregation functions are then applied to the result values of the observations in one group to calculate a new aggregated observation.

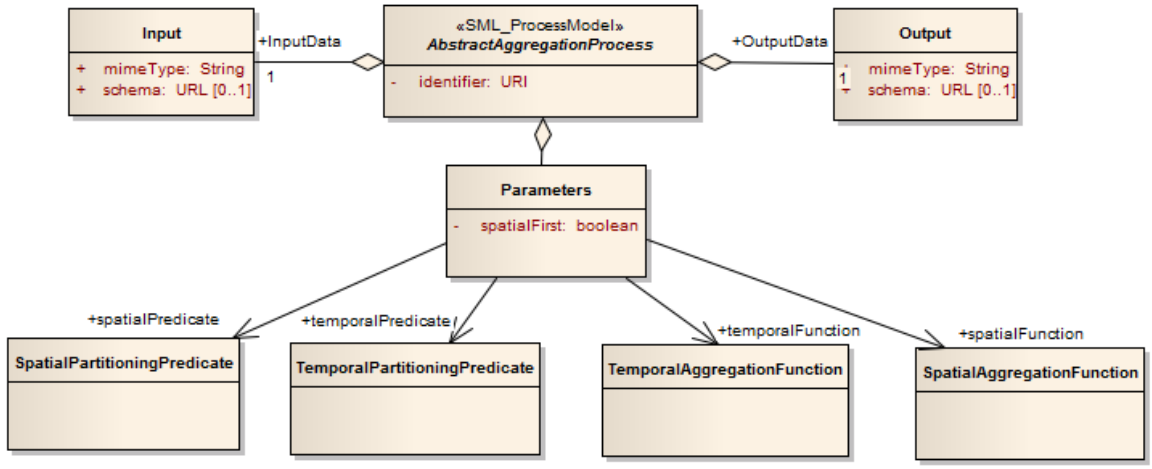


Figure 7: UML diagram of the aggregation process model used in the STAS.

The central class of the process model is the `AbstractAggregationProcess` that inherits from the `ProcessModel` defined in the Sensor Model Language (SensorML), a standard for the description of sensing processes as well as post-processing steps [Botts 2007]. Each aggregation process has an `Input`, an `Output`, and `Parameters`. Additionally, an aggregation processcan have a name, a description and a global identifier. For simplicity, the diagram only shows the `identifier` of the aggregation process. The `Input` of an `AbstractAggregationProcess` acts as a container for the original data thatcan be encoded as U-O&M or U-NetCDF. Similarly, the `Output` of an `AggregationProcess` is a container for the aggregated data. Both `Input` and `Output` can either directly contain the data or reference server instances providing the data (e.g. a Sensor Observation Service (SOS) or a Web Coverage Service (WCS)). The `Parameters` class contains the spatial and temporal predicates as well as the aggregation functions that are used to aggregate the attached variable values. The subtypes of the `AbstractAggregationProcess` are defined by the subtypes of the predicates and aggregation functions.

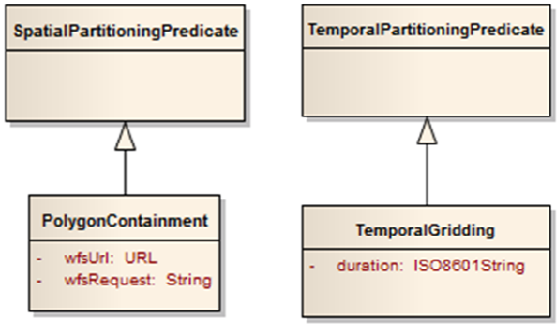


Figure 8: UML diagram showing example subtypes of partitioning predicates for an aggregation process that aggregates point measurements to polygons in space and to grids in time.

Several subtypes of the predicates and aggregation functions can be defined.Figure 8 shows two subtypes of the partitioning predicates. The spatial predicate `PolygonContainment` is defined to aggregate point measurements (possibly also other geometry types like lines) to polygons that are retrieved from a Web Feature Service (WFS), the main web service for retrieving spatial vector data. Thus, the type has two additional parameters, namely an URL

pointing to a WFS and a request which selects certain polygon features. The temporal predicate `TemporalGridding` groups the temporal extent of all observations into time intervals of equal duration and thus needs a parameter duration that defines the length of the duration.

Depending on the chosen aggregation function, the order of grouping and the order of applying an aggregation function (spatial first or temporal first) can be of importance. For example, computing the daily maxima first and then averaging them spatially is different to calculating spatial averages first and then applying the max function to the spatial averages. Thus, an additional parameter indicating the order is introduced with`spatialFirst`. Together with the extendedU-O&M model, we now have the data models needed to define the aggregation processes provided by the spatio-temporal aggregation service.

4.2 Service Interface for Spatio-temporal Aggregation

To integrate the aggregation processes in the UncertWeb architecture [UW-D2.2 2011], we defined the Spatio-Temporal Aggregation Service (STAS) as a profile of the OGC WPS.Figure 9 illustrates in an exemplary workflow how the STAS can be integrated in UncertWeb. The STAS can be linked dynamically with server instances to retrieve input observations and publish the aggregated observations. These can be Sensor Observation Services in case of U-O&M data or Web Coverage Services in case of U-NetCDF data, but also just plain HTTP or FTP servers. Thus, the aggregated results are stored and can be retrieved by several environmental models or several times by the same environmental model.

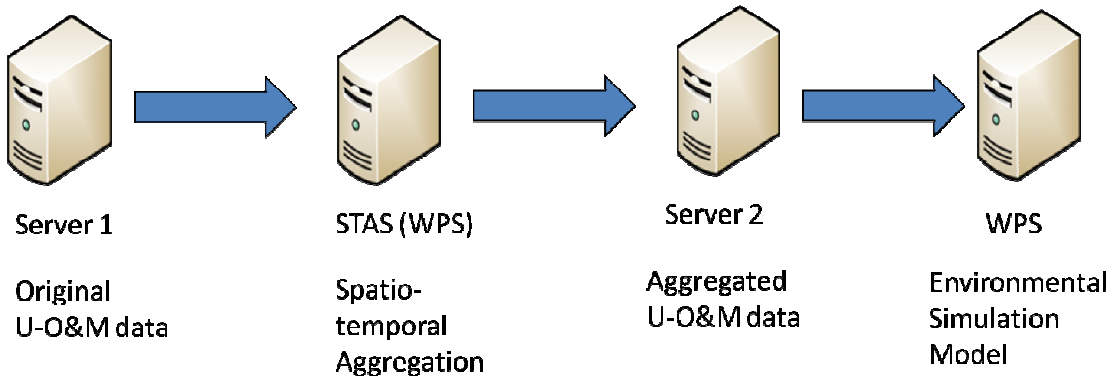


Figure 9: Workflow example illustrating how the STAS can be integrated in the UncertWeb architecture.

According to the WPS specification, a profile consists of (i) unique identifiers for its processes implemented as UnifiedResource Names (URN), and of (ii) process descriptions including the definitionof input and output parameters.We now describe these two required parts of our STAS profile: (i) An URN scheme for defining the identifiers of the processes (Section 4.2.1), and (ii) the implementation of process descriptions according to the aggregation process model as defined above (Section 4.1).

4.2.1 URN Scheme for Aggregation Processes

Following the common URN scheme of the OGC as defined in [Whiteside2009], a basic URN has the form `urn:ogc:def:objectType`. Thus, the basic URN for aggregation processes is defined as `urn:ogc:def:aggregationProcess`. Originally, only one generic aggregation process that also receives the partitioning predicates and aggregation

functions as parameters was planned. However, as the partitioning predicates and the aggregation functions might require additional process parameters (see, for example, Figure 8), it was decided to create a process subtype for each of the different combinations of partitioning predicates and aggregation functions. Thus, the spatial grouping predicate (`sgp`) and spatial aggregation function (`saf`) are appended to the basic URN followed by the temporal grouping predicate (`tgp`) and temporal aggregation function (`taf`). This results in the generic URN as `urn:ogc:def:aggregationProcess:sgp:saf:tgp:taf`.

For example, the URN `urn:ogc:def:aggregationProcess:polygonContainment:sMean:tGridding:tMax` identifies the process which averages observations (`sMean`) that are contained in polygons (`polygonContainment`) and calculates the maximum value (`tMax`) of these aggregates for temporal intervals (`tGridding`). Depending on the value of the `spatialFirst` parameter (described in the next section), the temporal aggregation can also be applied first. The additional parameters needed for the predicates and aggregation functions are defined in the process descriptions as defined in the next paragraph.

4.2.2 Process Descriptions of Aggregation Processes

To perform a spatio-temporal aggregation, the `Execute` operation of the STAS has to be invoked. The parameters of the `Execute` request for a specific aggregation process are described in its process description, which can be retrieved through the `DescribeProcess` operation. We now describe the parameters of the aggregation processes by mapping the parameters defined in our aggregation process model (Section 4.1) to the STAS process descriptions. The aggregation process parameters that every aggregation process inherits from the `AbstractAggregationProcess` are shown in Table 6. As described before, the partitioning predicates and aggregation functions are defined in the URN of the aggregation process and not passed as parameters.

Table 6: Common input parameters of all aggregation processes

Input Parameter Name	Cardinality	WPS Input Type	Description
`Identifier`	1	LiteralData	The identifier of the aggregation (following the URN scheme as defined in 4.2.1)
`Variable`	0..*	LiteralData	If there is more than one variable in the data, this property indicates which variables should be aggregated
`InputData`	1	ComplexData	Original data that should be aggregated. Format has to be U-O&M or NetCDF.
`SpatialFirst`	0..1	LiteralData	Indicates whether the spatial Aggregation should be done first (true) or not (false); default value is true.
`TargetServer`	0..1	LiteralData	URL of server to which aggregated data should be written. If not passed, default URL of WPS for providing output references is used.
`TargetServerType`	0..1	LiteralData	Should indicate the type of server (e.g. FTP, OGC SOS, OGC WCS, etc.)

The `Identifier` parameter has to be passed and contains the URN of the aggregation process to identify the aggregation process. The `InputData` parameterwhich contains the original data also has to be provided in every `Execute` request. Usually, this is a reference to a server providing the data, but data can also be encoded inline.One or more `Variable` parameters can be passed that identify the variable(s) that should be aggregated. In case of a NetCDF input file containing multiple variables, several variables whose values should be aggregatedmight be selected. The input can be either passed in the request or a reference to an external server can be provided. The `SpatialFirst` parameter indicates whether the spatial aggregation should be done first (default) or not. In case the aggregation results should be published on a certain server, clients are able to pass the URL of the target server in the `TargetServer` parameter. To indicate the type of the TargetServer (e.g. SOS or WCS), the `TargetServerType` parameter can be used.In addition, an `Execute` request can contain optional additional parameters depending on the spatial and temporal grouping predicates and aggregation functions. An example for an aggregation function requiring additional parameters is Block Kriging, where an additional parameter is a variogram model. Taking the aggregation process example as introduced in Section 4.2.1, the additional parameters of this process are shown in Table 7and have to be defined when invoking the aggregation process by the `Execute` operation.

Table 7: Additional input parameters of the aggregation process urn:ogc:def:aggregationProcess:polygonContainment:sMean:tGridding:tMax

Input Parameter Name	Cardinality	WPS Input Type	Description
`FeatureCollection`	1	ComplexData	FeatureCollection containing the polygons to which the data should be aggregated spatially.
`Duration`	1	LiteralData	Duration of time intervals (as specified in ISO:8601) into which the temporal extent of data should be partitioned; the original data is then aggregated temporally to these time intervals[2].

The response of an `Execute` operation has to provide only one element that is shown in Table 8. The `OutputData` of an `Execute`operation contains the aggregated data, encoded either as U-O&M or as U-NetCDF.

Table 8: Output parameter of all aggregation processes

Output Parameter Name	Cardinality	WPS Input Type	Description
`AggregatedData`	1	ComplexData	Aggregated data. Format has to be U-O&M or U-NetCDF.

4.3 Implementation

The STAS has been implemented as an extension of the 52°North WPS implementation[3]. Figure 10 shows the components used in the STAS implementation. Blue components are provided by the 52°North Geoprocessing community, the STAS component is shown in mustard colour and the pale yellow boxes are other components developed in the UncertWeb project. The STAS implementation uses the 52°North WPS framework. In order to enable the aggregation of data encoded in the UncertWeb data models, it uses the UncertWeb WPS IO Extension that in turn makes use of the different APIs for U-NetCDF, UncertML and U-O&M.

[2]Usually, the duration chosen does not allow to partition the temporal extent into time intervals of equal length. We are starting with the first time instant and then calculating the intervals. Thus, the last interval might be shorter then the rest of the intervals.

[3]The STAS implementation is available under an open source licence (GPL v2), managed by 52North. It can be assessed at https://svn.52north.org/svn/geostatistics/main/uncertweb/stas/trunk.

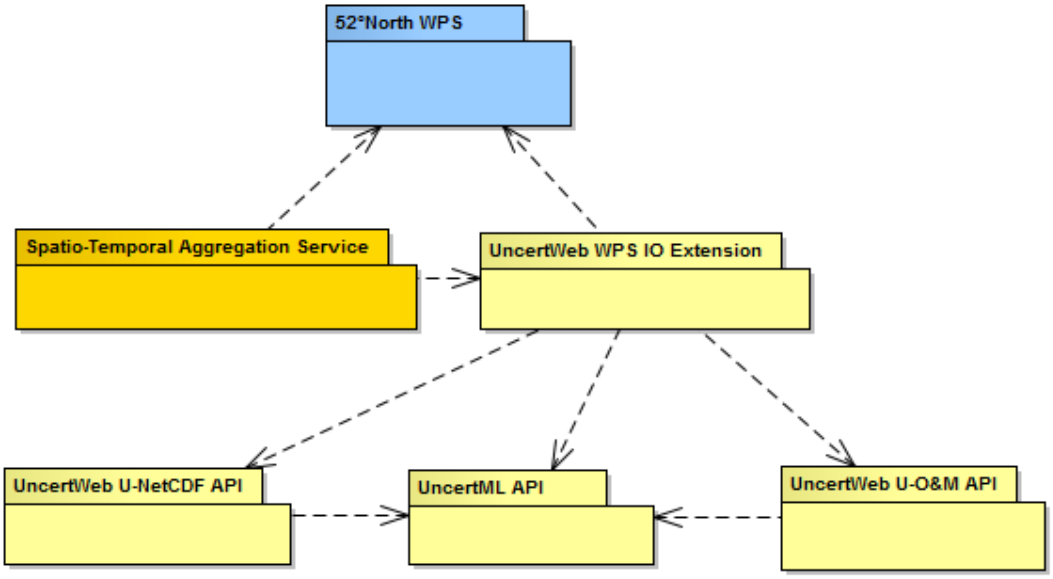


Figure 10: UML component diagram of the components used in the STAS implementation.

4.3.1 Usage of the STAS

In this subsection, we illustrate how the STAS can be used within the UncertWeb project. A typical use of the STAS is shown in Figure 11. The `User` in the figure can be a person directly invoking the server, but the `User`can also be an orchestration component or another model implemented as a Web service. In order to retrieve information which aggregation processes are supported by the STAS, the `User` invokes the `GetCapabilities` operation. The response of this operation lists the different aggregation processes that are supported. In order to retrieve the exact parameters of a specific aggregation process, the `DescribeProcess` operation has to be invoked by passing the identifier of the aggregation process. The STAS then returns a description of the aggregation process indicating the different parameters needed. Having this information, the user is now able to execute an aggregation process by passing the input parameters to the STAS.

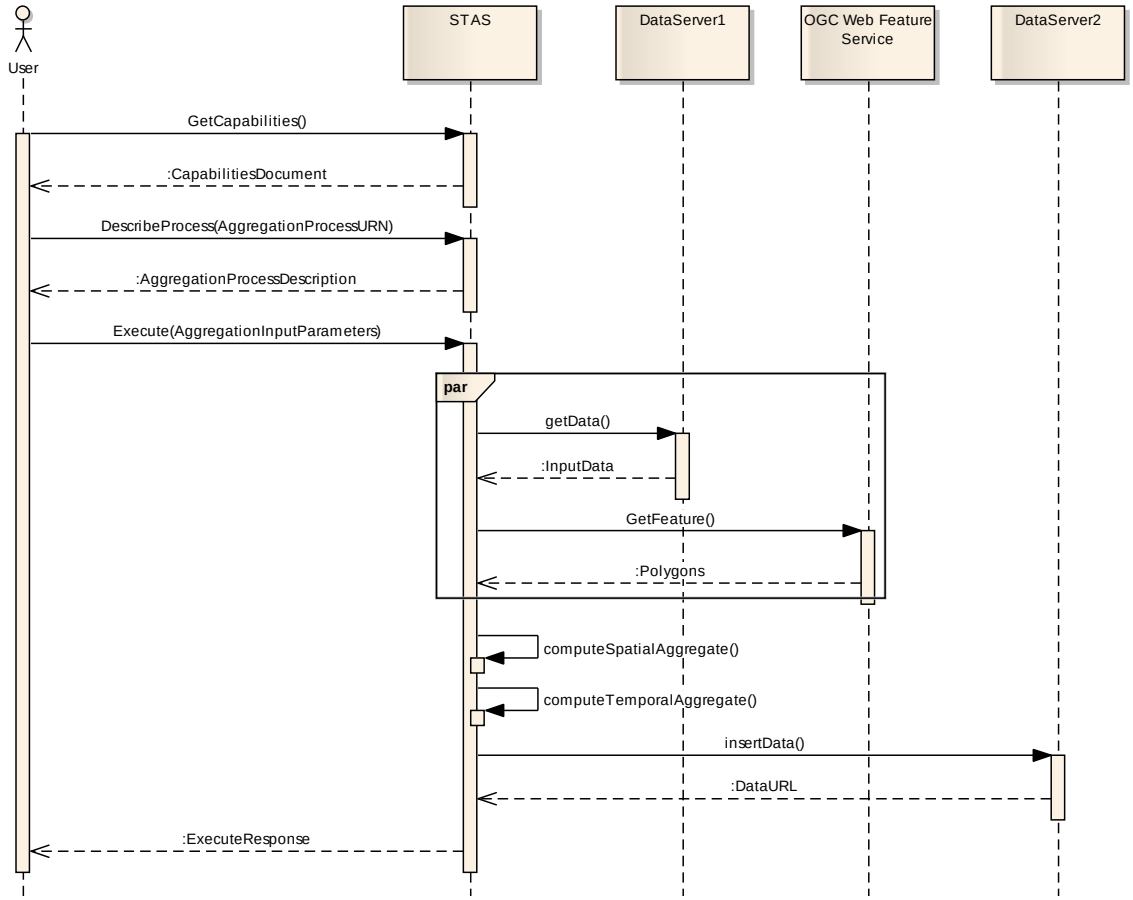


Figure 11: UML sequence diagram illustrating the usage and aggregation execution of STAS.

The internal execution of the STAS gets the required input data from an external server (the data can also be passed directly in the Execute request). In addition, geometries that are needed for the spatial grouping predicate can also be queried from an external server. In Figure 11, the example from Section 4.2.1is shown where Polygons are retrieved from an OGC Web Feature Service and next used as grouping predicate for spatial point measurements. The STAS then is able to group the data spatially and temporally and compute aggregates for these groups. When the aggregation is completed, the aggregated data can be published on another server in order to provide it to other Web Services using the aggregated data. The `ExecuteResponse` then can either contain an URL pointing to the server where the aggregated data are available or provide these inline.

4.3.2 Implementation of additional aggregation processes in the STAS

Currently, the STAS implementation provides prototypical implementations of several grouping predicates that allow grouping of U-O&M point measurements to Polygons provided by a WFS. Furthermore, temporal gridding of a temporal extent (a tessellation into intervals of equal length) is supported for time. The supported aggregation functions are ArithmeticMean, Maximum, Median, Minimum, and Sum. In order to easily add new spatial or temporal grouping predicates and functions, the STAS implements an AlgorithmRepository that automatically detects new grouping predicates and aggregation functions and creates aggregation processes out of them. Therefore, annotations in the Java code of the grouping predicates and aggregation functions are used to indicate which kind of

grouping it is and how to map it to identifiers in the URNs (see Section 4.2.1). The process descriptions are then created automatically without additional XML editing.

In case more complex (non-linear) aggregation methods are needed,an aggregation process class can be implemented. In order to retrieve the common inputs, we have implemented an `AbstractAggregationAlgorithm` class that provides the common inputs as well as the output type definition automatically. A specific aggregation process can then inherit the common inputs and outputs and add an additional implementation of an aggregation process that might require additional input parameters. For this, we have also defined a utility class `SingleProcessInput` that is used to automatically create the service descriptions from the algorithm implementation without additional configuring of XML. Furthermore, the mechanism allows the user to add new aggregation processes at runtime. A tutorial how to add new processes to the STAS implementation is available online [UW-STAS-TUT 2011].

4.3.3 Case Study – Aggregation of European Air Quality Observations

To demonstrate the developed approach,this section presents how the STAS prototype is deployed to aggregate European air quality observations, which are provided by the European Environmental Agency (EEA). In particular, the approach is implemented to aggregate Ozone observations collected in Germany to mean and maximum values in temporal intervals and to averages of these temporal aggregates in space. The service deployment for the case study is shown in Figure 12. A SOS instance[4] serves approximately 30 million observations extracted from the AirBase database files of the European Environmental Agency [EEA 2011]. Administrative areas of Germany are provided as vector-based features by a WFS.

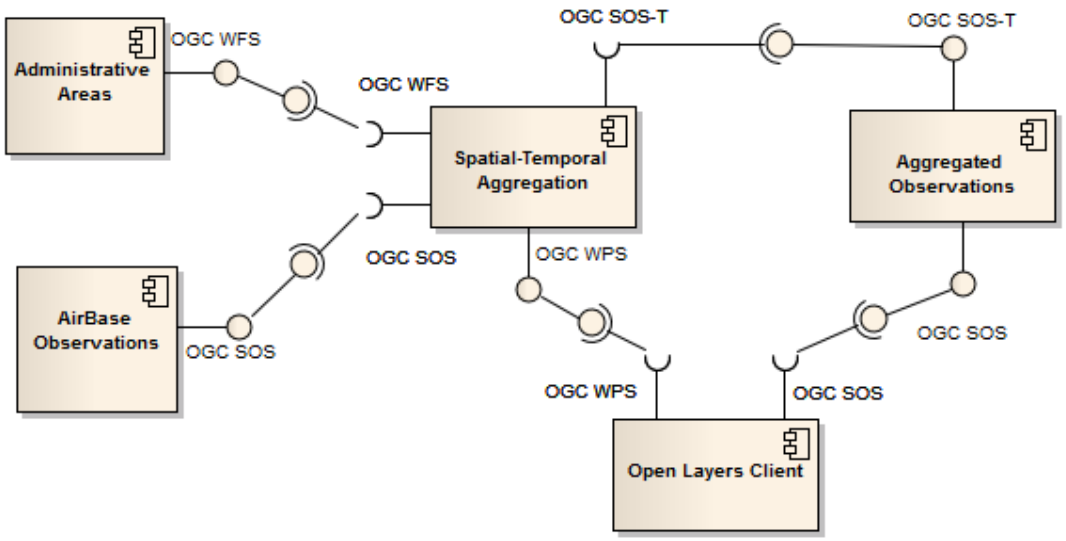


Figure 12: UML component diagram showing the different components used for the aggregation of European air quality observations.

The STAS uses a subset of the observations offered by the SOS (Ozone observations for Germany) to aggregate these to the boundaries of the German federal states as shown in the sequence diagram in Figure 10. In case of the non-linear temporal aggregation (`max`), the temporal aggregation has to be executed before the spatial aggregation which is indicated

[4] As the SOS server is providing its observations in O&M 1.0, we have been using O&M 1.0 in this case study.

using the `spatialFirst` flag. The aggregated observations are then inserted in another SOS which is used to provide the aggregated observations.As a client, the OpenLayers SOS client [Eijnden 2010] has been extended to support point as well as extensive observations in space and time. In addition, a simple user interface has been developed that allows the sending ofplain XML requests to SOSs and WPSs.Figure 13shows the OpenLayers client visualizing observations before the aggregation (left side) and the resulting observations after the aggregation (right side). In this example, the observations have been averaged over polygons in space and time intervals. The variances of the original observations have been used for the error bar shown in the aggregated result visualization.

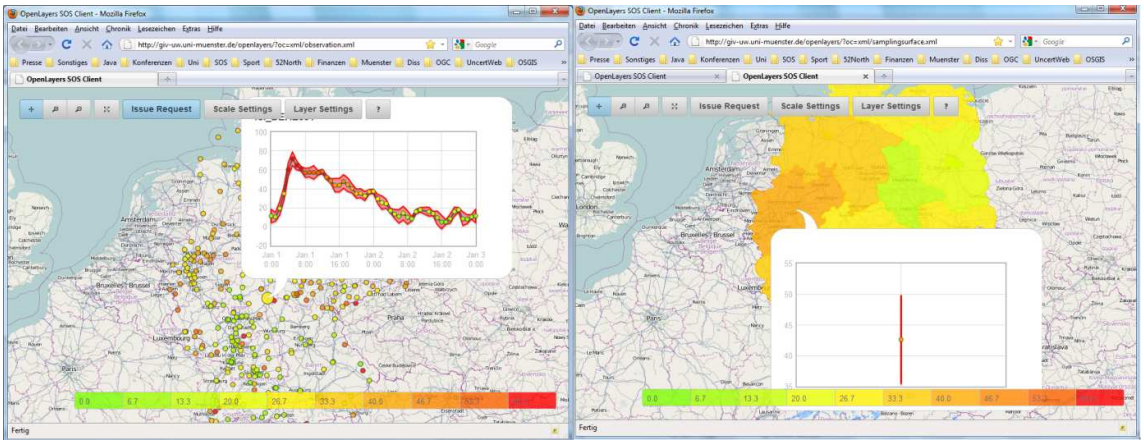


Figure 13: OpenLayers client visualizing the observations before (left) and after (right) aggregation.

5 Disaggregation

In this chapter we describe the theoretical background of the disaggregation method adopted by the UncertWeb project, i.e. Area to Point Kriging. We also explain how this method was implemented in the R programming language and provide a simple illustration. Providing the statistical methodology of spatial disaggregation under uncertainty is important because this is a more complex and more recentlydeveloped methodology than that used for spatial aggregation. We illustrate Area To Point kriging with a simple case.

5.1 Geostatistical framework of Area to Point Kriging

Area to point (ATP) kriging makes predictions of an environmental variable at point locations from areal data of that same variable. This allows disaggregating or downscaling data from large support to small support. The geostatistical framework of ATP kriging is described in [Kyriakidis 2004, Goovaerts 2008, Yoo and Kyriakidis 2006]. The advantage of ATP kriging as with any kriging algorithm is that it not only makes predictions of the target variable but that it also quantifies the accuracy of these predictions, by means of the kriging standard deviation. This is essential for UncertWeb because handling uncertainty and uncertainty propagation is at the heart of the UncertWeb framework. In addition, ATP kriging also satisfies the condition that the average/sum of the predictions (and simulations) at the point locations within any area equals the average/sum value of that area used as observational data in the kriging algorithm.

For these reasons ATP kriging is a valuable method for achieving spatial disaggregation within the UncertWeb project.

The methodology presented here is restricted to Ordinary ATP kriging, which makes the following assumptions about the spatial distribution of the variable at point and areal support:

- the value of the variable Z at point support (Z_p) is a realisation of anormally distributedsecond-order stationary random process **Z**defined on the area of interest. The value of Z at block support (Z_B) is the average of all values of Z at point support within the block*B*. That is:

$$\boldsymbol{Z_B} = \frac{\mathbf{1}}{|\boldsymbol{B}|}\int_{\boldsymbol{p}\in\boldsymbol{B}} \boldsymbol{Z_p}\boldsymbol{dp} \tag{1}$$

  where $|B|$ is the area of *B.*

- Since $\mathbf{Z}_p$is normally distributed, $\mathbf{Z}_B$is also normal. The joint distribution of point and block values ($\mathbf{Z}_p$, $\mathbf{Z}_B$) isalso normal, with constant mean and variance–covariance matrix:

$$\begin{pmatrix}\boldsymbol{Z_p}\\ \boldsymbol{Z_B}\end{pmatrix} \sim \boldsymbol{N}\left(\boldsymbol{\mu}, \begin{bmatrix}\boldsymbol{C_{pp}} & \boldsymbol{C_{pB}}\\ \boldsymbol{C_{Bp}} & \boldsymbol{C_{BB}}\end{bmatrix}\right), \tag{2}$$

  where μ is vector of constants, i.e. the constant mean value of **Z**, which is invariant under change of support,

  $C_{pp}$is the variance-covariance matrix between point support values,

  C_{BB} is the variance-covariance matrix between block support values,

  C_{pB} is the covariance matrix between point and block support values,

$C_{Bp}$is the covariance matrix between the block and point support values.

- The spatial autocorrelation at point support is known. It is assumed that C_{pp}can be derived from a parametric stationary covariance function or variogram. Next C_{BB} and C_{pB} can be derived from C_{pp} through:

$$\boldsymbol{C_{pB} = \frac{1}{|B|}\int_{p' \in B} C_{pp'} dp'} \quad \textbf{(Error! Bookmark not defined.)}$$

$$\boldsymbol{C_{BB'} = \frac{1}{|B|}\frac{1}{|B'|}\int_{p \in B}\int_{p' \in B'} C_{pp'} dp dp'} \textbf{(Error! Bookmark not defined.)}$$

Since the joint distribution is normal, the optimal prediction of $\boldsymbol{Z_p}$ is obtained from block observations as follows:

$$\boldsymbol{\hat{Z}_p = \mu + C_{pB} C_{BB}^{-1}(Z_B - \mu)} \textbf{ (Error! Bookmark not defined.)}$$

The prediction error variance is given by:

$$\boldsymbol{Var(Z_p - \hat{Z}_p) = C_{pp} - C_{pB} C_{BB}^{-1} C_{pB}} \textbf{(Error! Bookmark not defined.)}$$

5.2 Implementation

For implementation we discretize the block B and replace the integrals in equations (3) and (4) by approximate sums:

$$\boldsymbol{C_{pB} \cong \frac{1}{|B|}\sum_{p' \in B, k=1}^{K} C_{pp'(k)}} \textbf{ (Error! Bookmark not defined.)}$$

where K is number of discretized points in block B. Similarly:

$$\boldsymbol{C_{BB'} = \frac{1}{|B|}\frac{1}{|B'|}\sum_{p \in B, k=1}^{K}\sum_{l \in B', l=1}^{L} C_{p(k)p'(l)}} \textbf{(Error! Bookmark not defined.)}$$

These equations were programmed in the R language and the result used to evaluate equations (5) and (6) using ATP kriging, also in R.

The R code for Ordinary ATP kriging from areal support data is provided in Appendix D.The code provides the various functions used in ATP kriging and illustrates these by running the example described in the next section.

5.3 Illustration

To check the code and illustrate ATP kriging it was applied to the Meuse data set (http://cran.r-project.org/web/packages/gstat/vignettes/gstat.pdf).Figure 14shows the observed block values of the organic matter concentration of the soil for grid cells that are 200 by 200 meters. These block data were used as conditioning data to predict the organic matter concentration at point support, for which the following point support variogram was used:

$$\boldsymbol{\gamma_{pp'} = f(|p - p'|) = 3 + 10 \cdot \left(\frac{3}{2}\frac{|p-p'|}{10^3} - \frac{1}{2}\left(\frac{|p-p'|}{10^3}\right)^3\right)} \qquad (9)$$

The result is given in Figure 15, both in terms of the predicted value as well as the associated prediction error standard deviation. The results make sense and confirm that ATP kriging works well and was correctly implemented. The average of the point predictions within each block was equal to the block averages used as conditioning data, and the predictions are also spatially smooth which agrees with the stationary assumption. The ATP kriging standard deviations are realistic too and confirm that block conditioning data can substantially reduce uncertainty, except for the nugget variance part of the variogram.

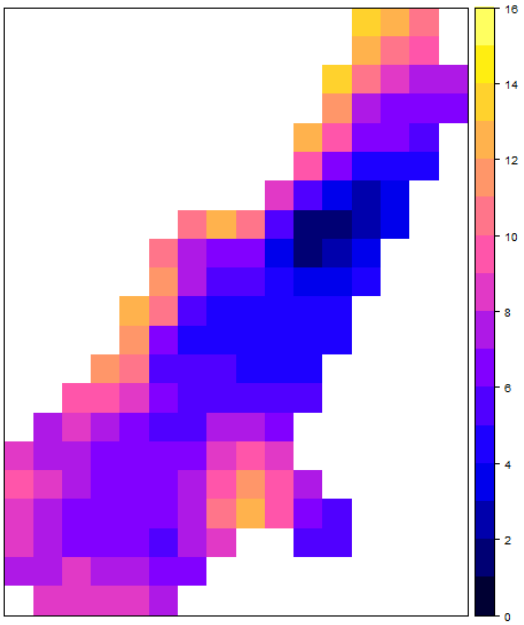


Figure 14: Block averaged values of topsoil organic matter content for the Meuse area used as conditioning data in ATP kriging

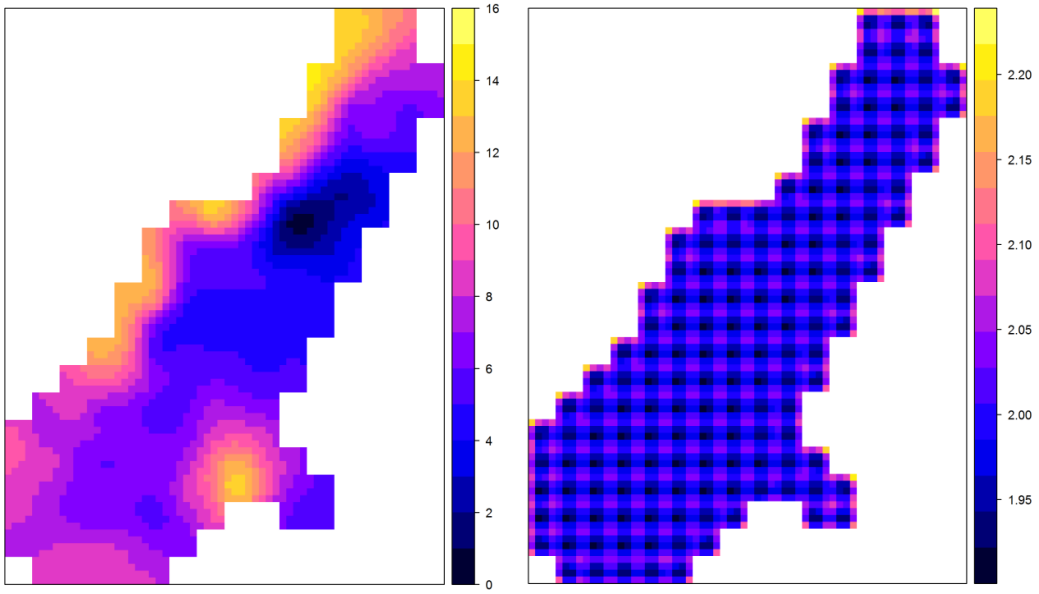


Figure 15: Point prediction (left) and standard deviation map (right) of topsoil organic matter content as obtained with ATP kriging.

6 Conclusions

In this deliverable we presented requirements and tools for aggregation and disaggregation of spatio-temporal data in the Model Web. The requirements were analysed in the application scenarios of the UncertWeb project and emphasize that there are different needs for aggregation and disaggregation as well as for uncertainty representations. As there are different requirements for aggregation and disaggregation and because unlike aggregation the statistical methodology behind disaggregation is more complex and much still in an experimental phase, aggregation and disaggregation were addressed separately. This is also confirmed by the task description in the Description of Work of UncertWeb, where for aggregation a web-based implementation was envisaged whereas for disaggregation it was agreed to only explore how simple methods could be included in the service.

In order to address the needs for aggregation and to allow for maximum flexibility, the propagation of uncertainty in spatio-temporal aggregation is done numerically using Monte Carlo simulation. Hence, the aggregation tools do not need to be aware of uncertainty, but are run for each Monte Carlo simulation. We developed a generic approach for deploying the aggregation processin the Model Web that allows for different partitioning predicates as well as aggregation functions. The Spatio-Temporal Aggregation Service (STAS), a service interface for aggregation processes, has been defined as a profile of the OGC Web Processing Service. The STAS provides well-defined aggregation process identifiers as well as common inputs and outputs of aggregation processes, based upon the data models used in UncertWeb. A prototype that implements the service interface has been developed as an extension of the 52°North Web Processing Service and has been deployed in a case study for aggregating air quality observations. The STAS will also be used in the use cases of UncertWeb in the third year of the project.

While the STAS defines the service interface for providing the aggregation processes in the Web, the *spacetime* package in R has been implemented to allow for an easy and flexible implementation of aggregation processes. In this way the aggregation functions and predicates need not be re-implemented in a STAS implementation, but can be used as a backend of the STAS. Furthermore, the *spacetime* package in R allows fora direct integration of aggregation processes in existing model implementations written in R, in case aggregation is needed within a model. In order to support standard spatial data types and deploy the spatial aggregation processes for these types, an extension of the 52°North WPS has been developed that allows to upload R scripts and publish them as WPS processes. While the current prototype is limited regarding the support of spatio-temporal data types, a future version of the WPS4R might offer an easy way to implement the STAS interface.

As there is also a need for disaggregation, area to point (ATP) Kriging has been implemented in R using the *krige0*function in the *gstat*package. Basic testing showed that the methodology is sound and the implementation correct, but further testing is required. In principle it should not be difficult to incorporate ATP disaggregation as a backend in the STAS, in a similar fashion as was done with the aggregation functionality of the *spacetime* package.

As the implementation of the scenario work packages is currently in progress, different aggregation and disaggregation processes will be deployed via the STAS depending on the specific needs of the implementations. Therefore, R can be used to easily implement the aggregation and disaggregation processes in a model while WPS4R and the STAS interface allow for the deployment of these processes in the uncertainty-enabled Model Web. The flexibility and ease of use of these services will be tested in work packages 4 to 7 where aggregation and disaggregation operations are employed in the use cases.

7 References

[Bierkens 2000]
Bierkens, M. F. P. , Finke, P.A. and De Willigen, P., 2000. Upscaling and Downscaling Methods for Environmental Research. Dordrecht, Kluwer.

[Bivand 2008]
Bivand, R.S., Pebesma,E.J., and Goméz-Rubio,V. 2008. Applied spatial data analysis with R. Springer, New York.

[Cox 2007]
Cox, S. (Ed.), 2007. Observation and Measurements - Part 1 - Observation Schema. OGC 07-022r1. Open Geospatial Consortium, Inc., 86pp, http://portal.opengeospatial.org/_les/?artifact id=22466, [accessed 22 March, 2011].

[EEA 2011]
European Environmental Agency. AirBase - The European air quality database - datasets.http://www.eea.europa.eu/data-and-maps/data/airbase-the-european4air-quality-database-2 [accessed 22 March, 2011].

[Eijnden 2010]
Eijnden, B.v., 2010. OpenLayers: SOS and Inspire. Bart van Eijnden.Presentation at FOSS4G. http://2010.foss4g.org/presentations/2891.pdf[accessed 22 March, 2011].

[Geller 2009]
Geller G. and Turner, W., 2009.The Model Web: A Concept for Ecological Forecasting, Geoscience and Remote Sensing, IEEE Transactions on, 47(1), 80 - 91.

[Goovaerts 2008]
Goovaerts P, 2008, "Kriging and semivariogram deconvolution in the presence of irregular geographical units" Mathematical Geosciences 40 101-128

[Yoo and Kyriakidis 2006]
Yoo E H, Kyriakidis P C, 2006, "Area-to-point Kriging with inequality-type data" Journal of Geographical Systems 8 357-390

[Heuvelink 1999]
Heuvelink, G. and Pebesma, E., 1999. Spatial Aggregation and Soil Process Modelling. Geoderma, 89, 47-65.

[Herring 2011]
Herring, J. (Ed.), 2011. OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture. OGC 06-103r4. https://portal.opengeospatial.org/modules/admin/license_agreement.php?suppressHeaders=0&access_license_id=3&target=http://portal.opengeospatial.org/files/%3fartifact_id=25355. [accessed 10 October, 2011]

[Jeong 2004]
Jeong, S.H., Fernandes, A.A.A., Paton, N.W., Griffiths, T., 2004. A Generic Algorithmic Framework for Aggregation of Spatio-Temporal Data, in: SS-DBM '04: Proceedings of the 16th International Conference on Scientific and Statistical Database Management, IEEE Computer Society, Washington, DC, USA. p. 245.

[Kyriakidis 2004]
Kyriakidis P C, 2004, "A Geostatistical Framework for Area-to-Point Spatial Interpolation" Geographical Analysis 36 259-289

[Pebesma 2011]
Pebesma, E., 2011. Classes and Methods for spatio-temporal data in R. Journal of Statistical Software, submitted.

[Portele 2007]
Portele, C. (Ed.), 2007 OpenGIS® Geography Markup Language (GML) Encoding Standard. OGC 07-036. http://portal.opengeospatial.org/files/?artifactid=20509 [accessed 14 September, 2011].

[Schut 2007]
Schut, P. (Ed.), 2007. OpenGIS Web Processing Service. OGC 05-007r7. Open Geospatial Consortium, Inc., 87pp.http://portal.opengeospatial.org/les/?artifact id=24151 [accessed 22 March, 2011].

[Stasch 2011]
Stasch, C., Autermann, C., Foerster, T., Pebesma, E., 2011. Spatio-Temporal Aggregation of European Air QualityObservations in the Sensor Web. Computers &Geosciences, accepted.

[UW-D1.1 2011]
UncertWeb Consortium, “UncertML requirements”, Deliverable 1.1, 2011.

[UW-D1.2 2011]
UncertWeb Consortium, “UncertML best practice proposal”, Deliverable 1.2, 2011.

[UW-D2.2 2011]
UncertWeb Consortium, “Service frameworks for modelling resources”, Deliverable 2.2, 2011.

[UW-D8.1 2011]
UncertWeb Consortium, “Report on integration requirements in UncertWeb”, Deliverable 8.1, 2011.

[UW-DOW 2009]
UncertWeb Consortium, “UncertWeb - Annex I – „Description of Work□”, 2009

[UW-STAS-TUT 2011] AggregationProcessHowTo. https://wiki.aston.ac.uk/foswiki/bin/view/UncertWeb/AggregationProcessHowto [accessed 16 October, 2011]

[Whiteside 2009]
Whiteside, A. (Ed.), 2009. Definition Identifier URNs in OGC namespace. OGC 07-092r3. Open Geospatial Consortium, Inc., 46pp. http://portal.opengeospatial.org/les/?artifact id=30575 [accessed 22 March, 2011].

[Williams 2009]

Williams, M.; Cornford, D.; Bastin, L., Pebesma, E. (Ed.) Uncertainty Markup Language. OGC 08-122r2. Open Geospatial Consortium, Inc., 61pp. http://portal.opengeospatial.org/files/?artifactid=33234 [accessed 22 March, 2011]

Appendix A: Aggregation Questionnaire

UncertWeb Questionnaire about Aggregation/Disaggregation Requirements, 08/04/2011

Please respond to this document byMay 15th. You will need 10-15 minutes to answer the questions. Thanks in advance!

1. Introduction

This questionnaire is developed within the UncertWeb project in order to mainly assess the requirements for the uncertainty-enabled spatio-temporal aggregation service that will be developed in Task 3.2. Additionally, it will briefly assess the requirement of spatio-temporal disaggregation.

In the framework of the UncertWeb project, we consider two main requirements for a spatio-temporal aggregation:

1. Aggregating observations that are input of a model to an aggregation level required by the model.
2. Mediating between different models where the output of one model is at another aggregation level then required for the input of the next model.

For disaggregation requirements, the main considerations are also about the use of data at different disaggregation levels for model inputs.

The first part of this questionnaire is about the requirements for aggregation. The second part is about the requirements for disaggregation. The third part is about the data availability.

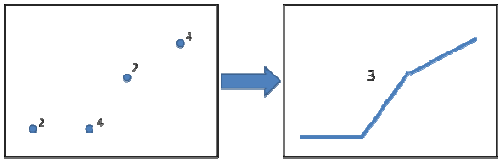
Basic Terms

Find below a definition of the basic terms used in our survey. The definitions are taken from Bierkens et al. (2000) and Jeong et al. (2007).

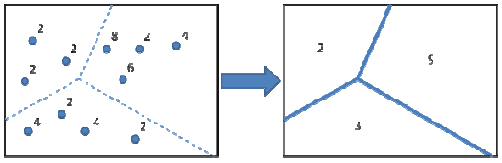
Terms	**Definition**
Aggregation	An aggregation process computes a value, an aggregate, for a group of attribute values by means of an aggregation function. The attribute values are grouped by a partitioning predicate.
Aggregate	Aggregated value.
Aggregation Function	Function used to compute the aggregate.
Partitioning Predicate	Predicate used to group objects before aggregating values attached to these objects.
Spatio-temporal Aggregation	A spatio-temporal aggregation process combines objects in space and time by spatial and/or temporal partitioning predicates and computes aggregates for certain attribute values attached to these objects by an aggregation function.
TimeInstant	Point in Time.
TimeInterval	Duration in time.
TimeCoverage	Tessellation of a temporal extent into TimeIntervals with equal duration and *without gaps* between the TimeIntervals.
Disaggregation	The opposite of aggregation.

Part 1: Questions about Aggregation

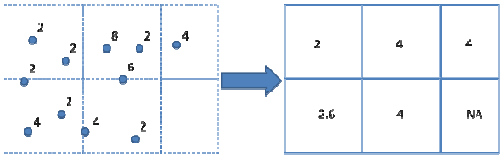
1. In your case of UncertWeb, what spatial geometry types need to be aggregated? (Please mark with x)



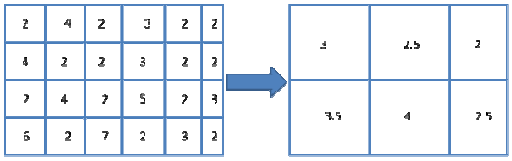
[ ] Points to Lines



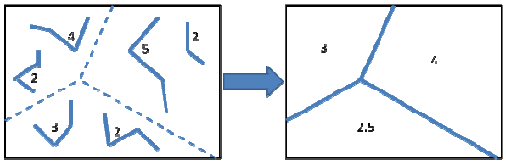
[ ] Points to Polygon



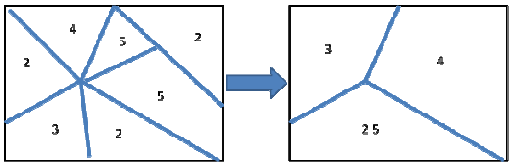
[ ] Points to Grid



[ ] Grid to Grid



[ ] Lines to Polygons



[ ] Polygons to Polygons

Others (e.g. Grid to Polygons):

2. What spatial aggregation functions do you use? Are they linear (e.g. the aggregated value is the Mean or Sum) or non-linear (e.g. Median, proportion of points in area exceeding a threshold)?

3. Do you want to consider uncertainty in your aggregation? If yes, how?

4. What additional information about the spatial aggregate do you consider to be useful (e.g. number of original values, provenance information)?

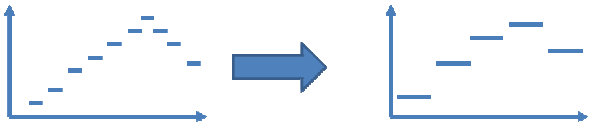
5. What temporal types need to be aggregated in your data? (Please mark with x)

[ ] TimeInstants to TimeIntervals

[ ] TimeInstants to TimeCoverage

[ ] TimeIntervals to TimeIntervals

[ ] TimeIntervals to TimeCoverage



[ ] TimeCoverage to TimeCoverage

Others:

6. What temporal aggregation functions do you use? Are they linear (e.g. aggregated value is the Mean) or non-linear (e.g. Median)?

7. Do you wish to consider uncertainty in your temporal aggregation? If yes, how?

8. What additional information about the temporal aggregate do you consider to be useful (e.g. number of original values, provenance information)?

9. What spatio-temporal geometries need to be aggregated? (Please use the geometries and aggregations as defined in question 1. and 5., e.g. “Points with TimeInstants” to “Lines with TimeIntervals”, etc.)

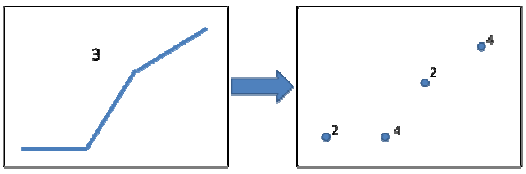
10. What spatio-temporal aggregation functions do you use?

11. Do you wish to consider uncertainty in your spatio-temporal aggregation? If yes, how?

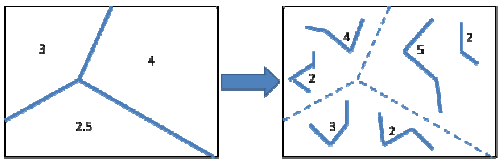
12. What additional information about the spatio-temporal aggregate do you consider to be useful (e.g. number of original values, provenance information)?

Part 2: Questions about Disaggregation

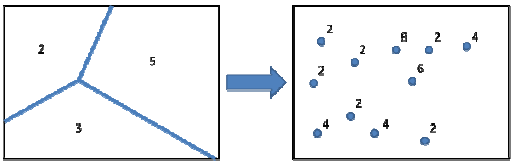
13. What spatial geometry types need to be disaggregated? (Please mark with x)



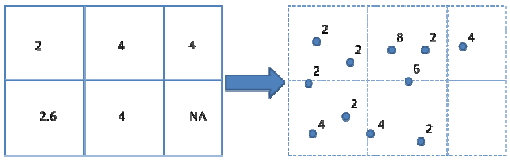
[ ] Lines to Points



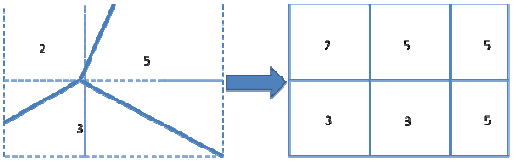
[ ] Polygons to Lines



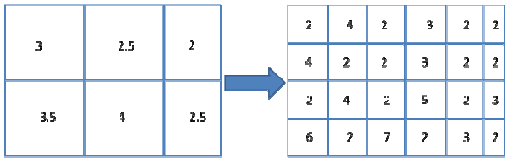
[ ] Polygons to Points



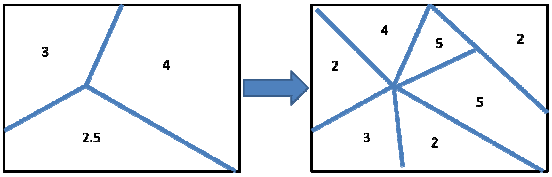
[ ] Grid to Points



[ ] Polygons to Grid



[ ] Grid to Grid



[ ] Polygons to Polygons

Others:

14. What value type do you want to have in your disaggregated data (e.g. the same value type as the original data or something else)? Please specify.

15. Do you wish to consider uncertainty in your spatial data to be disaggregated? If yes, how?

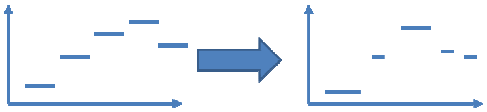
16. What additional information about the spatial disaggregate do you consider to be useful (e.g. information about the method used, provenance information)?

17. What temporal geometry types are disaggregated in your data? (Mark with x)

[ ] TimeIntervals to TimeInstants

[ ] TimeCoverage to TimeInstants

[ ] TimeIntervals to TimeIntervals



[ ] TimeIntervals to TimeCoverage

[ ] TimeCoverage to TimeCoverage

Others:

18.What value do you want to have in your disaggregated data (e.g. the same value type as the original data or something else) ? Please specify.

19. Do you wish to consider uncertainty in your temporal data to be disaggregated? If yes, how?

20. What additional information about the temporal disaggregate do you consider to be useful (e.g. disaggregation method used, provenance information)?

21. What spatio-temporal geometries need to be disaggregated? (Please use the geometries and aggregations as defined in question 13. and 17., e.g. “Polygons, with TimeCoverage to Points with TimeInstants”, etc.)

22. What spatio-temporal disaggregation value do you expect to obtain?

23. Do you consider uncertainty in your spatio-temporal data to be disaggregated? If yes, how?

24. What additional information about the spatio-temporal disaggregate do you consider to be useful (e.g. disaggregation method used, uncertainty, provenance information)?

Part 3 Questions about data availability

25. Could you provide data from your use case that need to be aggregated/disaggregated spatially? If yes, please provide a short description about the data and how to access it (1-2 sentence).

26. Could you provide data that need to be aggregated/disaggregated temporally? If yes, please provide a short description about the data and how to access it (1-2 sentence).

27. Could you provide data that need to be aggregated/disaggregated spatio-temporally? If yes, please provide a short description about the data and how to access it (1-2 sentence).

28. Do you have extra data/information for your disaggregated data (e.g. point measurements, covariance, correlation information)?

THANK YOU!

3. References

Jeong, S.H., Fernandes, A.A.A., Paton, N.W., Griffiths, T., 2004. A Generic Algorithmic Framework for Aggregation of Spatio-Temporal Data, in: SS-DBM '04: Proceedings of the 16th International Conference on Scientific and Statistical Database Management, IEEE Computer Society, Washington, DC, USA. p. 245.

Bierkens, M. F. P. , Finke, P.A. and De Willigen, P., 2000. Upscaling and Downscaling Methods for Environmental Research. Dordrecht, Kluwer.

Appendix B: R vignette of overlay

The document shown here is also published online as part of the *sp* CRAN project, accessible at http://cran.r-project.org/web/packages/spacetime/index.html.

Abstract

Numerical “map overlay” combines spatial features from one map layer with the attribute (numerical) properties of another. This vignette explains the R method “over”, which provides a consistent way to retrieve indices or attributes from a given spatial object (map layer) at the locations of another. Using this, the R generic “aggregate” is extended for spatial data, so that any spatial properties can be used to define an aggregation predicate, and any R function can be used as aggregation function.

Full-text version of the document can be found at
http://cran.r-project.org/web/packages/sp/vignettes/over.pdf

Appendix C: R vignette of spatio-temporal overlay

The document shown here is also published online as part of the *spacetime* CRAN project, accessible at http://cran.r-project.org/web/packages/spacetime/index.html.

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.

Full-text version of the document can be found at
http://cran.r-project.org/web/packages/spacetime/vignettes/sto.pdf

Appendix D: R code for spatialdisaggregation with A2P Kriging

```
library("sp")
data(meuse)
coordinates(meuse) = ~x+y
data(meuse.grid)
gridded(meuse.grid) = ~x+y
summary(meuse$om)
meuse$om[is.na(meuse$om)]<-0
hist(meuse$om)

library("gstat")
fom = gstat(id=c("om"),formula=om~1,data=meuse,nmax=24)
fom.vg = variogram(fom)
plot(fom.vg)
fom.vgm = fit.variogram(fom.vg,vgm(10,"Sph",1000,3), fit.sill=FALSE)
plot(fom.vg,fom.vgm, main = "model vgm(10,Sph,1000,3)")
fom = gstat(fom, id=c("om"),model=fom.vgm)
omsim = predict.gstat(fom,meuse.grid, BLUE=FALSE)
library("raster")
r= raster(omsim)
agg = aggregate(r, fact=5, fun=mean)
rcoarsegrid = rasterToPolygons(agg,fun=NULL, n=4, na.rm=TRUE, digits=12)
proj4string(rcoarsegrid)=CRS("+init=epsg:28992")
p_finegrid = spsample(rcoarsegrid, "regular", n = 2432)

# define variogram model FUNCTION that can deal with x and y
# being of class SpatialPolygons OR SpatialPoints:
vgm_model = function(x, y, vm = vgm(10, "Sph", 1e3, 3), n = 16) {
  stopifnot(is(x, "SpatialPolygons") || is(x, "SpatialPoints"))
  stopifnot(is(y, "SpatialPolygons") || is(y, "SpatialPoints"))
   nx = length(x)
   ny = length(y)
   V = matrix(NA, nx, ny)
   pb = txtProgressBar(style = 3, max = nx)
  if (is(x, "SpatialPolygons")&is(y, "SpatialPolygons"))
  for (i in 1:nx) {
     for (j in i:ny) {
          px = spsample(x[i,], n, "regular", offset = c(.5,.5))
          py = spsample(y[j,], n, "regular", offset = c(.5,.5))
       D = spDists(px, py)
       D[D == 0] = 1e-10
       V[i,j] = mean(variogramLine(vm, dist_vector = D, covariance = TRUE))
      V[is.na(V)]<-0
      V = lower.tri(V)*t(V)+V
     }
     setTxtProgressBar(pb, i)
   }
  else
  for (i in 1:nx) {
    px = spsample (x[i,], n, "regular", offset = c(.5,.5))
   for (j in 1:ny) {
        py = y[j,]
        D = spDists(px, py)
        D[D == 0] = 1e-10
        V[i,j] = mean(variogramLine(vm, dist_vector = D, covariance = TRUE))
   }
     setTxtProgressBar(pb, i)
   }
   close(pb)
   V
}
```

```
#modified krige0
extractFormula = function(formula, data, newdata) {
  # extract y and X from data:
    m = model.frame(terms(formula), as(data, "data.frame"))
    y = model.extract(m, "response")
    if (length(y) == 0)
        stop("no response variable present in formula")
    Terms = attr(m, "terms")
    X = model.matrix(Terms, m)

  # extract x0 from newdata:
    terms.f = delete.response(terms(formula))
    mf.f = model.frame(terms.f, newdata) #, na.action = na.action)
    x0 = model.matrix(terms.f, mf.f)
  list(y = y, X = X, x0 = x0)
}

idw0 = function(formula, data, newdata, y) {
  s = coordinates(data)
  s0 = coordinates(newdata)
  if (missing(y))
    y = extractFormula(formula, data, newdata)$y
  D = 1.0 / (spDists(s0, s) ** 2)
  sumD = apply(D, 1, sum)
  D %*% y / sumD
}

CHsolve = function(A, b) {
  # solves A x = b for x if A is PD symmetric
  A = chol(A)
  backsolve(A, forwardsolve(A, b, upper.tri = TRUE, transpose = TRUE))
}

krige0 <- function(formula, data, newdata, model, vm, beta, y, ...,
    computeVar = FALSE, fullCovariance = FALSE) {
  stopifnot(identical(proj4string(data), proj4string(newdata)))
  lst = extractFormula(formula, data, newdata)
  X = lst$X
  x0 = lst$x0
  if (missing(y))
    y = lst$y
  ll = (!is.na(is.projected(data)) && !is.projected(data))

  s = coordinates(data)
  s0 = coordinates(newdata)
  if (is(model, "variogramModel")) {
    require(gstat)
    V = variogramLine(model, dist_vector = spDists(s, s, ll),
      covariance=TRUE)
    v0 = variogramLine(model, dist_vector = spDists(s, s0, ll),
      covariance=TRUE)
    c0 = variogramLine(model, dist_vector = c(0), covariance=TRUE)$gamma
  } else {
    V = model(data, data)
    v0 = model(data, newdata)
    c0 = vm$psill[1]+vm$psill[2]
  }
  if (!missing(beta)) { # sk:
    skwts = CHsolve(V, v0)
    if (computeVar)
      var = c0 - t(v0) %*% skwts
  } else { # ok/uk -- need to estimate beta:
    skwts = CHsolve(V, cbind(v0, X))
    ViX = skwts[,-(1:nrow(s0))]
```

```
    skwts = skwts[,1:nrow(s0)]
    beta = solve(t(X) %*% ViX, t(ViX) %*% y)
    if (computeVar) {
      # (x0-X'C-1 c0)'(X'C-1X)-1 (x0-X'C-1 c0) -- precompute term 1+3:
      Q = t(x0) - t(ViX) %*% v0
      var = c0 - t(v0) %*% skwts + t(Q) %*% CHsolve(t(X) %*% ViX, Q)
    }
  }
  pred = x0 %*% beta + t(skwts) %*% (y - X %*% beta)
  if (computeVar) {
    if (!fullCovariance)
      var = diag(var)
    list(pred = pred, var = var)
  } else
    pred
}

# execute disaggregation with A2P kriging
vm = vgm(10, "Sph", 1e3, 3)
kr = krige0(value ~ 1, rcoarsegrid, p_finegrid, vgm_model,vm, computeVar =
TRUE)

out = SpatialPixelsDataFrame(p_finegrid, data.frame(val = kr))
spplot(out,zcol = "val.pred", col.regions=bpy.colors())
spplot(out,zcol = "std", col.regions=bpy.colors())
spplot(agg, col.regions=bpy.colors())
```