Filtering 1-D regular data

D. Renard

June 20th, 2014

Abstract

RGeostats is a library of geostatistical procedures available in the R environment. It is based on the geostatistical library **Geoslib** written in C language. The specificity of RGeostats is that it can deal with information defined in a space of any dimension, and can process any number of variables simultaneously.

This case study is designed in order to illustrate the use of geostatistics for filtering an information provided along a regular 1-D data set. This technique is known as *factorial kriging analysis* and is usually applied in the multivariate framework. However here it will be used on a single variable.

The interested user can also perform the contents of this test using the command *demo(RGeostats.1D)*.

1 Data set

1.1 Input file

The data set consists in the conductivity information collected regularly along a well log. A set of 850 samples is provided in the file called *Data1D.dat*. The file is composed of two columns of numbers:

- the first column contains the depth coordinate
- the second column contains the *conductivity* information

The first line of the file contains the variable names. When reading the *depth* column, one can easily check that the samples are regularly organized along a 1-D grid with a mesh of 0.02002m and an origin at 519.00074m.

1.2 Loading the information

The first task is to read this information and store it as a *R* format table called *Exdemo_1D_well.table*, using the following command:

```
Exdemo_1D_well.table <- read.table(file="Data1D.dat",header=TRUE)
```

For convenience, this data set is also saved as an R object. It suffices to use the following command:

```
data(Exdemo_1D_well.table)
```

The next step is to load the information in a new DB structure called *well.db*.

```
well.db <- db.create(Exdemo_1D_well.table,flag.grid=TRUE,
x0=519.00074,dx=0.02002,nx=850,autoname=FALSE)
```

The new DB structure created can be checked either by typing the following command:

```
print(well.db)
```

or by simply typing the name of the DB. We obtain the following printout:

```
File is organized as a regular grid
Space dimension = 1
Number of attributes = 4
Total number of samples = 850
Grid characteristics:
Origin :  519.001
Mesh :  0.020
Number :  850

Field = 1 - Name = rank - Locator = rank
Field = 2 - Name = x1 - Locator = x1
Field = 3 - Name = Depth - Locator = z1
Field = 4 - Name = Conductivity - Locator = NA
```

The two first fields have been generated automatically, the two columns of the table are loaded in fields 3 and 4.

The field #4 is now considered as the target variable:

```
well.db <- db.locate(well.db,4,"z",1)
```

Finally the field #3 is similar the field #2 (generated automatically) and can thus be deleted:

```
well.db <- db.delete(well.db,3)
```

Typing the name of the DB again, we obtain the following printout (only the variable paragraph is produced here):

```
Field = 1 - Name = rank - Locator = rank
Field = 2 - Name = x1 - Locator = x1
Field = 3 - Name = Conductivity - Locator = z1
```

We must finally check the statistics on the different variables contained in *well.db* structure, by typing the command:

```
print(well.db,flag.stats=T)
```

and obtaining the following printout (truncated here):

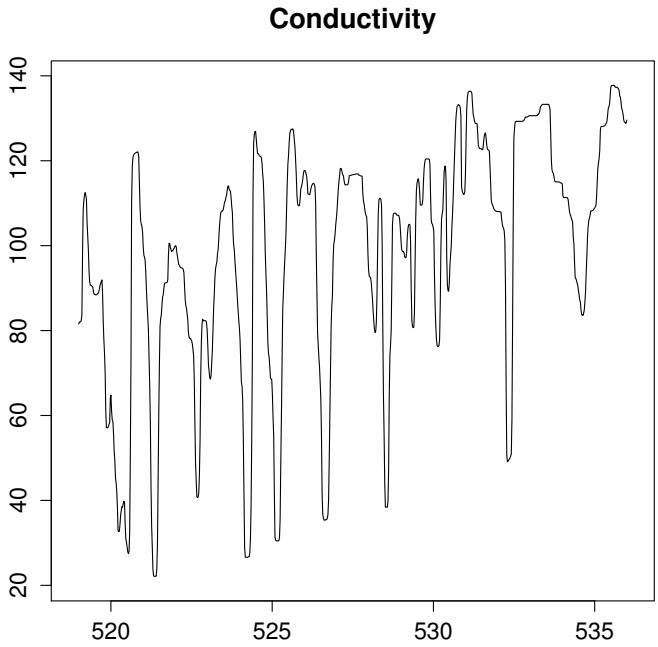
```
Data Base Statistics
--------------------
Locator = NA
Nb of active values = 850
Minimum value = 1.000
Maximum value = 850.000
Mean value = 425.500
Standard Deviation = 245.374
Variance = 60208.250

Locator = x1
Nb of active values = 850
Minimum value = 519.001
Maximum value = 535.998
Mean value = 527.499
Standard Deviation = 4.912
Variance = 24.131

Locator = z1
Nb of active values = 850
Minimum value = 22.113
Maximum value = 137.738
Mean value = 97.633
Standard Deviation = 28.816
```

1.3 Displaying the information

We can now produce a graphic where the information along the well is plotted, by typing the command:

```
plot(well.db)
```

Figure 1: Information along the well



We can see that the variable presents high frequency irregular variations.

2 Variography

2.1 Experimental variogram

We now wish to calculate the variogram of the *Conductivity* variable. As the information is defined on a regular grid, we calculate the variogram with a lag equal to the grid mesh. We calculate the variogram for 500 lags and store the result in a new *variogram* file called *well.vario*.

```
well.vario <- vario.grid(well.db,nlag=500)
```

The contents of the *variogram* structure can be defined by typing the following command:

```
print(well.vario)
```

or by simply typing the file name. We obtain the following printout (truncated here):

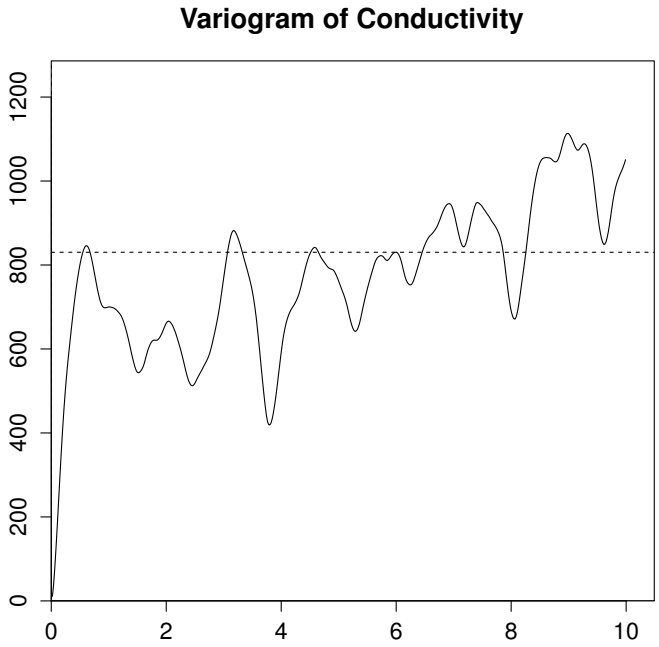
```
Variogram characteristics
=========================
Number of variable(s) = 1
Number of direction(s) = 1
Space dimension = 1

Direction 1
-----------
Number of lags = 10
Direction coefficients = ( 1.000)
Tolerance on direction = 0.000000
Calculation lag = 0.02002

For variable 1
Variance = 830.349
Rank Npairs Distance Value
0 849.000 0.000 0.000
1 849.000 0.020 10.788
2 848.000 0.040 37.993
3 847.000 0.060 75.783
4 846.000 0.080 120.070
5 845.000 0.100 168.345
6 844.000 0.120 218.862
7 843.000 0.140 269.987
8 842.000 0.160 320.285
9 841.000 0.180 368.870
...
```

We can also display this variogram using the command:

```
plot(well.vario,title="Variogram of Conductivity")
```

and obtain the following graphic:

Figure 2: Experimental variogram



2.2 Model definition

In the next step, we must fit a model which fits the experimental variogram. This operation is performed using an interactive procedure where we must define:

- the number of basic structures
- for each structure, its type, range and additional parameter (if necessary)

Note that a template model has already been prepared as an R object that must be loaded first:

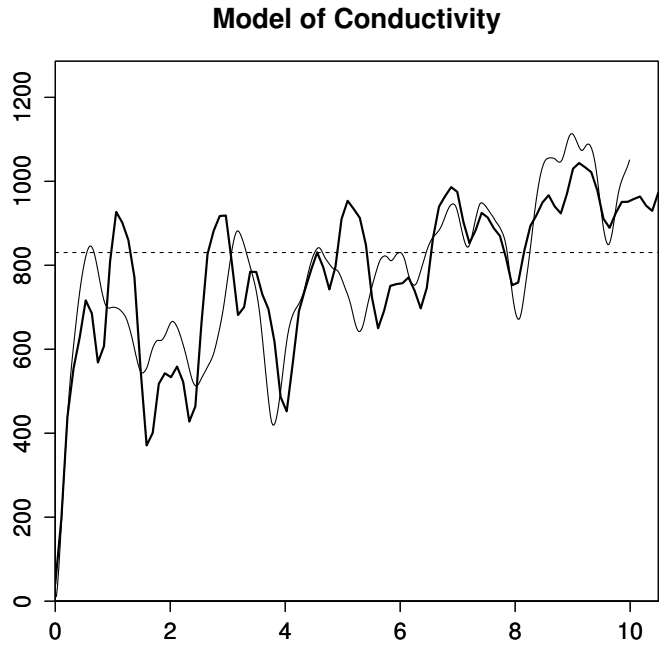
```
data(Exdemo_1D_Exdemo_1D_well.model)
```

Then, an automatic sill fitting procedure is used. The model is fitted using five basic structures: a nugget effect, a linear component and three cosine exponential components. The fitting is performed using the following command:

```
Exdemo_1D_well.model <- model.fit(well.vario,Exdemo_1D_well.model,
title="Model of Conductivity")
```

The following figure shows the experimental variogram and the fitted model.

Figure 3: Experimental variogram and fitted model



We can display the contents of the model by using the following command:

```
print(Exdemo_1D_well.model)
```

or simply by typing the name of the file *Exdemo_1D_well.model*. We obtain the following contents:

```
Model characteristics
=====================
Space dimension = 1
Number of variable(s) = 1
Number of basic structure(s) = 5
Number of drift function(s) = 1
Number of drift equation(s) = 1

Covariance Part
---------------
Nugget Effect
Sill = 40.661

Linear
Range = 1.000
Sill = 42.975

Cosexp
Range = 20.000
Parameter = 0.400
Sill = 64.169

Cosexp
Range = 20.000
Parameter = 0.800
Sill = 183.768

Cosexp
Range = 20.000 P
Parameter = 2.000
Sill = 266.376
```

3 Filtering

We now wish to perform the filtering operation using the geostatistical technique. For that sake, we still have to define the neighborhood parameters, i.e. the number of cell values which will be used in order to compute the filtered estimate for the central cell.

3.1 Neighborhood definition

The neighborhood is defined interactively and stored in a neighborhood file called *Exdemo_1D_well.neigh*, using the following command:

```
Exdemo_1D_well.neigh <- neigh.input(ndim=1)
```

As the image is defined on a regular grid, it makes sense to use an *image neighborhood*, which is composed of a set of grid nodes, centered on the target node. The neighborhood is defined by its radius (set to 100 nodes). When the neighborhood size is large, the calculations may become time consuming. It is then possible to sample the neighborhood by skipping one data over N : N is called the *skipping factor* (skipping ratio is set to 1 in this study).

An R object already prepared can be loaded by typing:

```
data(Exdemo_1D_well.neigh)
```

The contents of the Neighborhood object is obtained by typing:

```
print(Exdemo_1D_well.neigh)
```

and the following printout is obtained:

```
Neighborhood characteristics
============================
Image neighborhood option
Skipping factor = 1
Image radius :
[,1]

[1,] 100.000
```

3.2 Filtering step

We consider the *Conductivity* variable as the sum of five independent components, each component being responsible of each basic component of the model. The factorial kriging technique enables to estimate each component, together with a sixth component which corresponds to the local mean.

When an *image neighborhood* is used, characterized by its radius, the cells located close to the edge of the data set (closer than the neighborhood radius) cannot be estimated.

Each component is estimated individually. For example, the first component, corresponding to the first basic structure of the model, will be stored in the *well.db* file using the *radix s1*, when typing the following command:

```
well.db <- krimage(well.db,model=Exdemo_1D_well.model,
neigh=Exdemo_1D_well.neigh,cov.extract=1,radix="s1",modify.target=FALSE)
```

Similarly, we can extract the component corresponding to the mean and store it using the *radix sm:*

```
well.db <- krimage(well.db,model=Exdemo_1D_well.model,
neigh=Exdemo_1D_well.neigh,drift.extract=1,radix="sm",modify.target=FALSE)
```

If we now check the contents of the file *well.db,* we obtain the following printout:

```
Variables
---------
Field = 1 - Name = rank - Locator = rank
Field = 2 - Name = x1 - Locator = x1
Field = 3 - Name = Conductivity - Locator = z1
Field = 4 - Name = s1.Conductivity.estim - Locator = NA
Field = 5 - Name = s2.Conductivity.estim - Locator = NA
Field = 6 - Name = s3.Conductivity.estim - Locator = NA
Field = 7 - Name = s4.Conductivity.estim - Locator = NA
Field = 8 - Name = s5.Conductivity.estim - Locator = NA
Field = 9 - Name = sm.Conductivity.estim - Locator = NA
```

3.3 Visualizing the results

We now want to plot each component individually. For that sake, each variable must be defined as a target variable, before it can be visualized. This can be done in the following command, where each field is set, in turn, as the target:

```
plot(db.locate(well.db,"s1.Conductivity.estim","z",1))
```

The following figures show the results:

Figure 4: Component #1

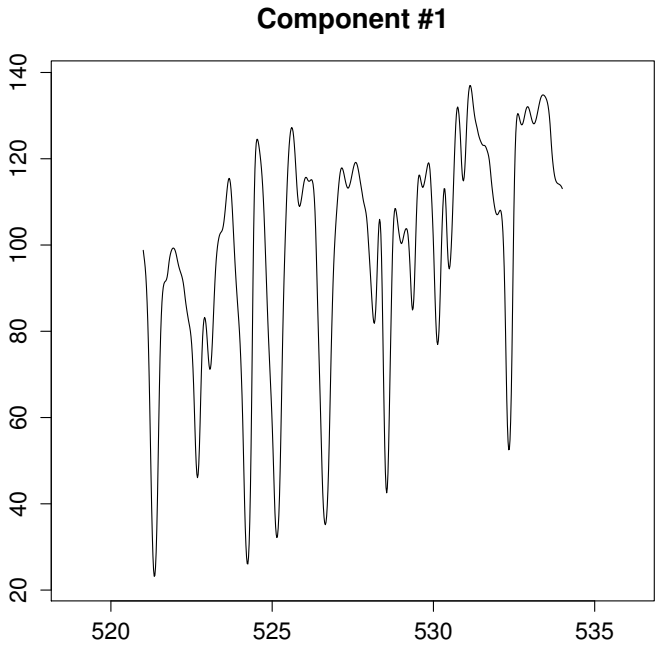


Figure 5: Component #2

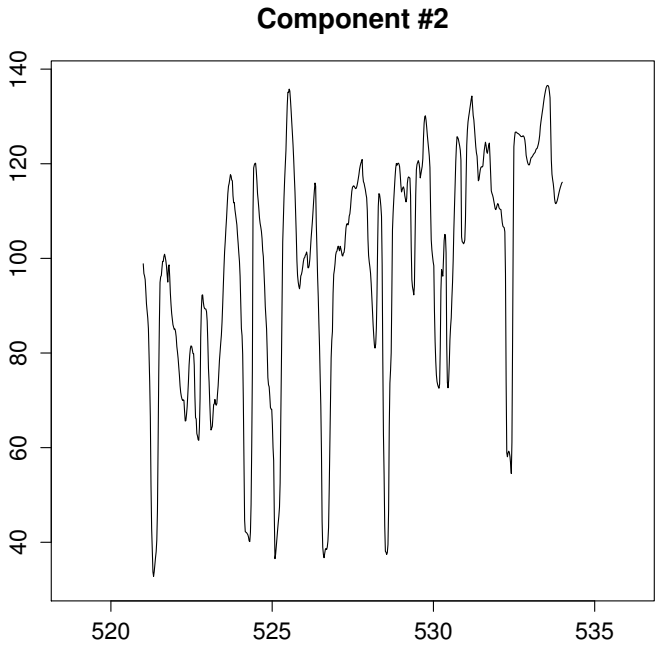


Figure 6: Component #3

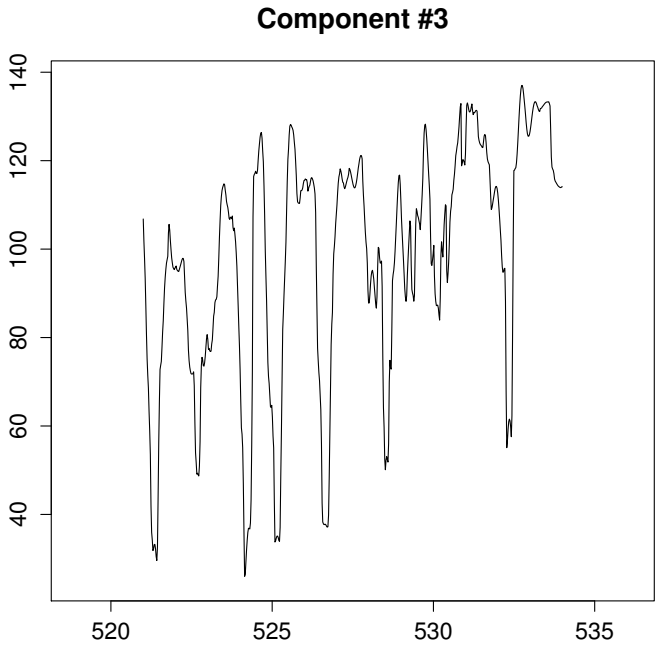


Figure 7: Component #4

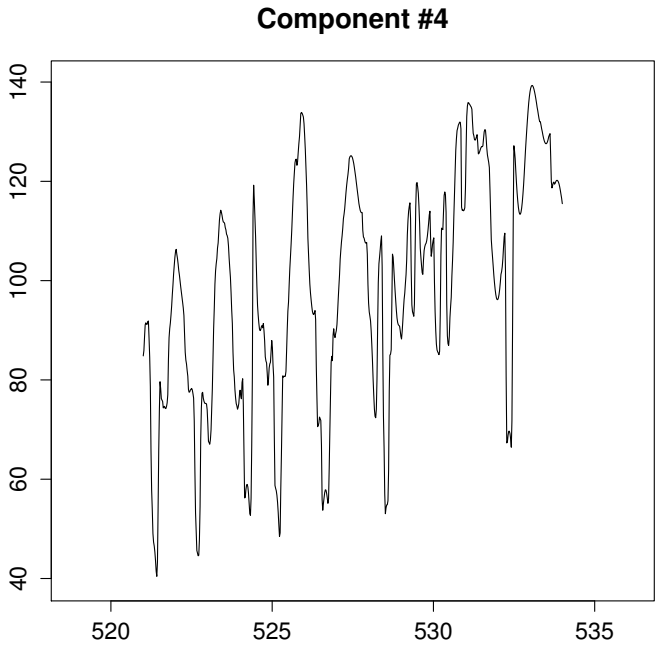


Figure 8: Component #5

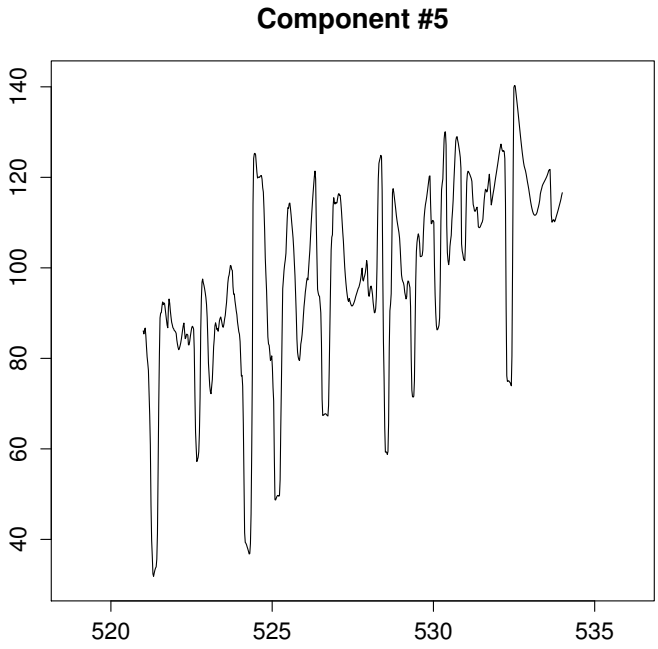
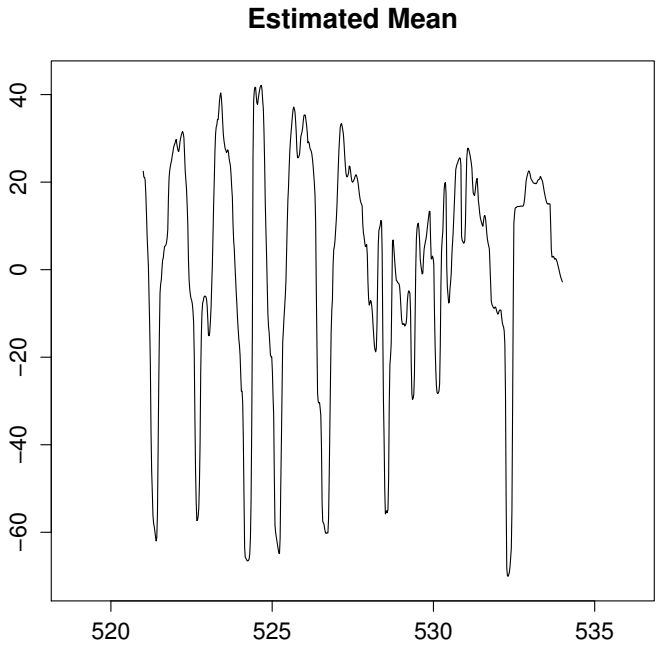


Figure 9: Component of the mean



3.4 Ultimate check

The last check is to ensure that the sum of the six extracted components results into the initial *Conductivity* variable.

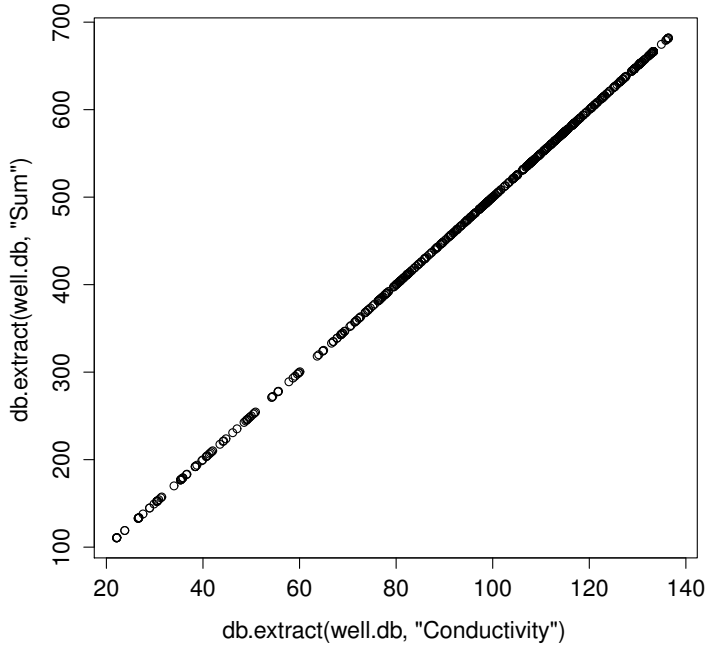
We construct the new variable *sum* which corresponds to the sum of the six extracted components by typing:

```
well.db <- db.add(well.db,Sum=
s1.Conductivity.estim+
s2.Conductivity.estim+
s3.Conductivity.estim+
s4.Conductivity.estim+
s5.Conductivity.estim+
sm.Conductivity.estim)
```

We can now draw the scatter diagram between the initial variable *Conductivity* and the newly created variable *sum*, by typing the command:

```
plot(db.extract(well.db,"Conductivity"),db.extract(well.db,"Sum"))
```

Figure 10: Proof of the decomposition



We obtain the following figure which proves that the six extracted components add up to the initial *Conductivity* variable: