Technical Note: Co-kriging with the gstat package of the R environment for statistical computing

D G Rossiter
University of Twente, Faculty of Geo-Information Science & Earth Observation (ITC)
Enschede (NL)

December 11, 2012

Contents

List of Figures

1 Introduction

This technical note shows how to perform co-kriging using the `gstat` geostatistical package [10] of the R environment for statistical computing and visualisation [3]. It does not present the theory of co-regionalisation or co-kriging; this is dealt with in many texts [e.g. 7, 2, 18, 4]. The main aim is to show various `R` techniques for data manipulation, calculation and graphics; emphasis is on the `gstat` geostatistical package, the `sp` spatial data package [9], and the `lattice` graphics package [14].

It also shows how to evaluate the success of a kriging interpolation by validation from an independent data set and by cross-validation from the sample data set.

These notes are designed so that you can *cut* the code shown in the boxes labelled **R code** and *paste* it directly into the R console; you should then see the output in the boxes labelled **R console output**. Of course, you are encouraged to edit the code and experiment.

> **Note**: These notes are evolving as the `gstat` package is increasingly linked to the `sp` package and its spatial classes, and as I learn more about using `sp` effectively. These notes may not always present the most efficient way to complete each task; suggestions are welcome. In particular, many of the graphics using the `levelplot` and `xyplot` methods of the `lattice` package can probably be done with `spplot` method of the `sp` package.

1.1 When to use co-kriging?

Co-kriging allows samples of an *auxiliary* variable (also called the *co-variable*), besides the *target* value of interest, to be used when predicting the target value at unsampled locations. The co-variable may be measured at the same points as the target (*co-located* samples), at other points, or both.

The most common application of co-kriging is when the co-variable is cheaper to measure, and so has been more densely sampled, than the target variable. In this example we show this typical situation, where there is a small sample set where both variables have been measured (co-located measurements) and a larger sample set where only the co-variable has been measured.

Co-kriging requires that both target and co-variable have a spatial structure that can be modelled, and in addition a spatially-dependent covariance.

> If the target and co-variable samples are co-located, and in addition the values of the co-variable are known at each prediction location

(e.g. an interpolation grid), more appropriate techniques are Kriging with External Drift (KED) or Regression Kriging (RK); these only requires a significant feature-space relation between target and co-variable.

1.2 R and gstat

The monograph "Introduction to the R Project for Statistical Computing for use at ITC" [13] explains how to use R on the ITC network, how to obtain and install your own (free and legal) copy, and how to use the documentation and tutorials. R is set up as a *base system* with many *contributed libraries*, among which are several for geostatistics, including `spatial` [16], `geoR` [11], and `gstat` [10], the subject of these notes. Each package has its own way of approaching geostatistics. This is one of the strengths of R: it is open to any practising statisticians who can write code to express their computational methods. Recently, several of these practicioners have together developed a common package `sp` to facilitate spatial statistics [9]; `gstat` depends on this package and loads it as necessary.

As with all R packages, `gstat` has on-line help: `library(help=gstat)` to get a list of methods in the package, `?variogram` to get help for the `variogram` method, and similarly for the other methods. The user's manual [8] refers to the stand-alone version, not the R package, but explains the algorithms in detail.

1.3 Example Data Set

This example uses the Meuse soil pollution dataset [12] and an interpolation grid of the study area in the south of the Netherlands, both provided with the `sp` R library as sample data sets. A brief explanation of the data fields is given in the on-line help (`?meuse`, `?meuse.grid`) once this library is loaded.

Task 1 : Load the `gstat` library, the dataset of sample points, and the interpolation grid. Examine the structure of the point set and grid. •

R code:
```
library(gstat)
?meuse
data(meuse)
str(meuse)
?meuse.grid
data(meuse.grid)
str(meuse.grid)
```

R console output:

```
Loading required package: sp

‘data.frame’: 155 obs. of  14 variables:
 $ x      : num  181072 181025 181165 181298 181307 ...
 $ y      : num  333611 333558 333537 333484 333330 ...
 $ cadmium: num  11.7 8.6 6.5 2.6 2.8 3 3.2 2.8 2.4 1.6 ...
 $ copper : num  85 81 68 81 48 61 31 29 37 24 ...
 $ lead   : num  299 277 199 116 117 137 132 150 133 80 ...
 $ zinc   : num  1022 1141  640  257  269 ...
 $ elev   : num  7.91 6.98 7.80 7.66 7.48 ...
 $ dist   : num  0.00136 0.01222 0.10303 0.19009 0.27709 ...
 $ om     : num  13.6 14 13 8 8.7 7.8 9.2 9.5 10.6 6.3 ...
 $ ffreq  : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
 ...

‘data.frame’: 3103 obs. of  5 variables:
 $ x     : num  181180 181140 181180 181220 181100 ...
 $ y     : num  333740 333700 333700 333700 333660 ...
 ...
```

Note that `gstat` automatically loads the `sp` package if necessary; the `meuse` and `meuse.grid` datasets are included in `sp`.

The data frame of sample points contains 155 observations of twelve variables[1], each with its coördinates in the Dutch national grid. The prediction grid has 3103 locations, spaced every 40 m in both the E and N grid directions, covering the irregularly-shaped study area.

2 Target and co-variables

In a mapping exercise we are often interested in one or more *target* variables, i.e. those to be mapped. When co-kriging we also need to select one or more *co-variables* to assist in the mapping of the target variables.

When mapping with co-kriging it is possible to use the same set of variables as target and co-variables; however this is typically used in the case of more or less equal sampling density. In these notes we consider the case of under-sampling of a target varible, using additional samples from co-variables.

2.1 The target variable

We select lead (abbreviation “Pb”) as the *target variable*, i.e. the one we want to map. This metal is a serious human health hazard. It can be inhaled as dust from disturbed soil or taken up by plants and ingested.

[1] although there are a few missing values of organic matter

The critical value for Pb in agricultural soils, according to the Berlin Digital Environmental Atlas[2], is 600 mg kg^{-1} for agricultural fields: above this level grain crops can not be grown for human consumption. Above 300 mg kg^{-1} crops must be tested; above 100 mg kg^{-1} consumption of leafy vegetables should be avoided. Levels below 200 mg kg^{-1} are required for sports fields or parks where soil may become bare from over-use. Natural levels in most unpolluted soils are on the order of 30 mg kg^{-1}.

Task 2 : Display histograms of the target variable and its $\log_{10}$ transform. What proportion of the samples are above the five thresholds? •

We will use the `lattice` graphics package throughout this case study; this is an R implementation by Sarkar [14] of the Trellis framework for data visualization developed at Bell Labs, based on the ideas in Cleveland [1]. It must be loaded before use. The `histogram` method is part of this package, not the base `graphics` package.[3]

R code:
```
library(lattice)
histogram(meuse$lead, nint=12)
histogram(log10(meuse$lead), nint=12)
        # proportions higher than various thresholds
ph <- function(level) round(100 * sum(meuse$lead > level)/length(meuse$lead), 1)
p <- NULL; lvls <- c(600, 300, 200, 100, 30)
for (l in lvls) p <- c(p, ph(l))
        # display a table of thresholds and proportions
(data.frame(cbind(level=lvls, percent.higher=p)))
rm(ph, l, lvls, p)
```

The `nint=` optional parameter is used to specify the number of histogram bins; if it is omitted, `lattice` will compute it as $(\log_2 n)+1$, in this case 8.

Note the use of a temporary function and a `for` loop to save typing and to ensure consistency in the output format, and the use of a temporary data frame to format the output as a table and give meaningful labels. This task could have been done more simply by repeating commands like

```
round(100*sum(meuse$lead > 600)/length(meuse$lead), 1)
```

[2] http://www.stadtentwicklung.berlin.de/umwelt/umweltatlas/ed103103.htm

[3] The `hist` method in the base graphics package gives similar output.

R console output:

```
  level percent.higher
1   600            0.6
2   300            8.4
3   200           27.1
4   100           54.8
5    30          100.0
```

Not many samples are too polluted for grain crops, but over a quarter are too polluted for use as playgrounds and half are too polluted for garden allotments. All are above background levels.

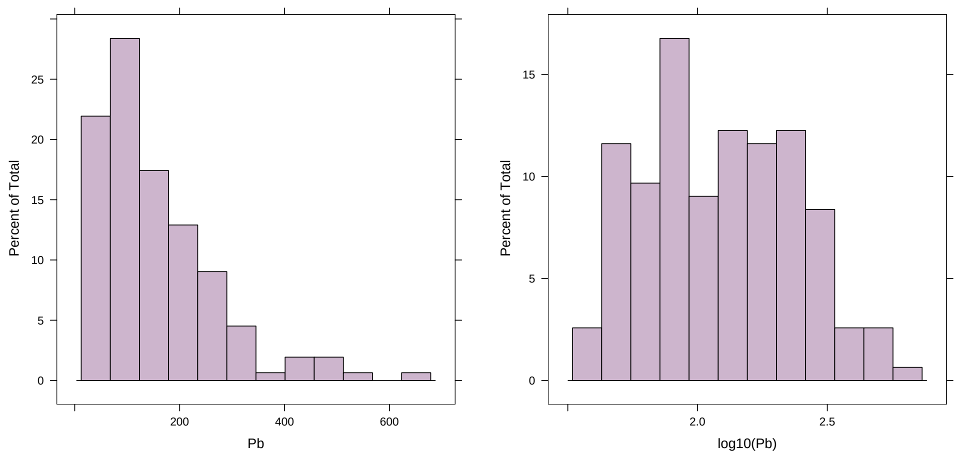

Figure 1: Histograms of target variable, original and log-transformed

Because of the right-skew and wide numerical range of the Pb values (Fig. 1), we will work with the log-transformed target variable; to allow easy interpretation of the results we will use base-10 logarithms.

2.2 Selecting the co-variables

A *co-variable* for use in co-kriging is correlated to the target variable in both feature and geographic space, and has usually been more densely sampled than the target variable.

Candidates for co-variables must have:

1. a *feature-space correlation* with the target variable;
2. a *spatial structure* (i.e. be modelled as a regional variable);
3. a *spatial co-variance* with the target variable.

There are two main ways to select a co-variable:

1. *theoretically*, from knowledge of the spatial process that caused the observed spatial (co-)distribution;
2. *empirically*, by examining the feature-space correlations (scatterplots) and then the spatial co-variance (cross-correlograms or cross-variograms).

We usually have some idea of the possible co-variables from theory (i.e. an *a priori* hypothesis) based on our knowledge of the process that (we think) gave rise to the data. The modelling process reveals if our hypothesis is supported by this particular data set. The second option, exploring the data (a sort of "data mining") is appropriate when there is no theory.

In this example we will compare two possible co-variables based on theory, which turn out to have different strengths of relation with the target variable.

First, we select **organic matter content** (abbreviation "OM"). It is cheaper than the heavy metal to measure in the laboratory, and may bind heavy metals in the soil [6, 15]; thus there may be a positive correlation in feature space between this co-variable and target variable. However, it is mostly formed in place in the soil and not transported with flood waters, as is Pb, so it is unclear if there will prove to be a spatial co-variance with Pb in the study area.

Second, we select **zinc content** (abbreviation "Zn"). Many of the industrial processes that result in lead pollution also produce zinc, and it is transported similarly, both as dust and sediment. It is generally measured along with lead at no additional cost (i.e. the same soil sample serves for both) so this example is somewhat artificial, but it will illustrate the effect of a highly-correlated (in both feature and geographic spaces) co-variable.

Task 3 : Display histograms of the co-variables and their log10 transforms. •

R code:
```
histogram(meuse$om, nint=12)
histogram(log10(meuse$om), nint=12)
histogram(meuse$zinc, nint=12)
histogram(log10(meuse$zinc), nint=12)
```

The co-variable OM is symmetric with a fairly narrow range (Fig. 2), and does not need to be transformed for feature-space correlation; however

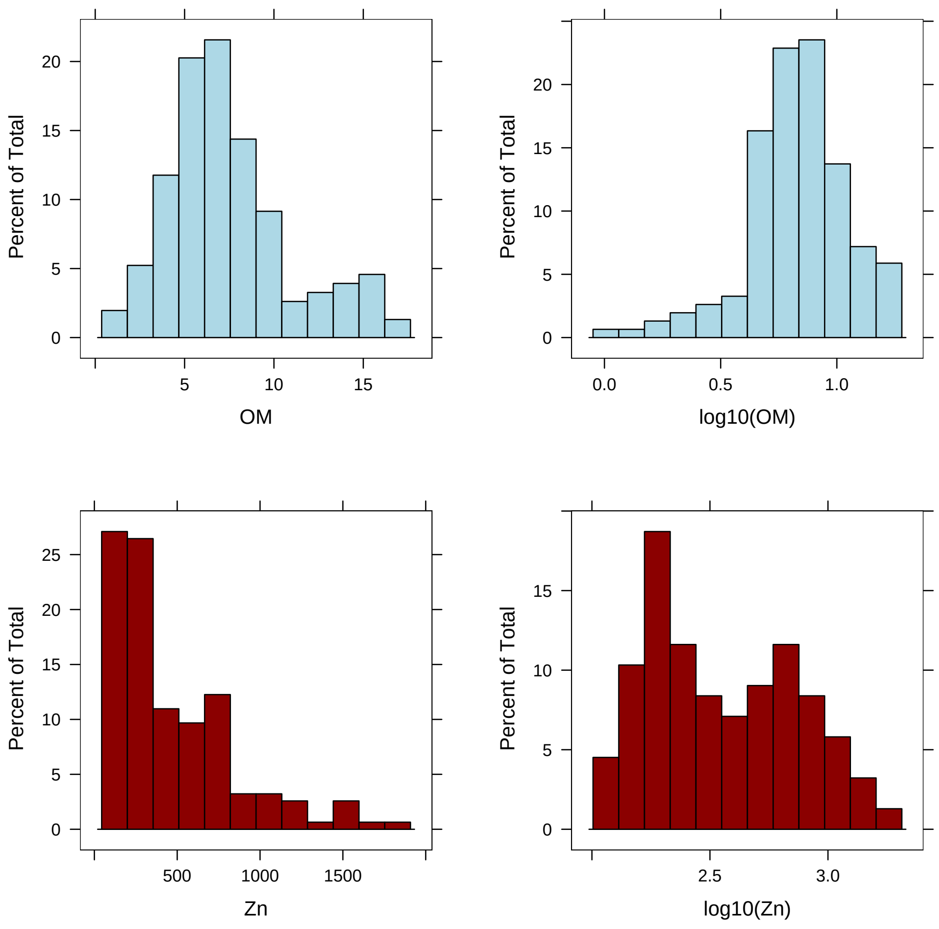

Figure 2: Histograms of co-variables, original and log-transformed

we will work with the transformed value to have comparable numbers for modelling co-regionalisation (§6). The co-variable Zn is skewed much like Pb, and so is log-transformed.

3 Simulation of under-sampling

To explore the value of co-kriging, we will simulate the situation where the target variable is under-sampled compared to the co-variable.

Task 4 : Make a subset of the observations of the target variable (lead) and the co-variables (organic matter and zinc), using every third sample from the full data set. •

R code:
```
meuse.pb <- meuse[ seq(1, length(meuse$lead), by=3),
                   c("x", "y", "lead", "om", "zinc")]
str(meuse.pb)
```

Note the use of the `seq` method to select every third row, and the `c` method to select only the relevant columns. Another possibility is the `sample` method to select a random sample of rows; however each sample is different, so you would not be able to compare your results with mine.

R console output:
```
'data.frame':   52 obs. of  4 variables:
 $ x   : num  181072 181298 181165 181232 180874 ...
 $ y   : num  333611 333484 333370 333168 333339 ...
 $ lead: num  299 116 132 80 285 240 207 123 75 67 ...
 $ om  : num  13.6 8 9.2 6.3 15.4 16.2 13.7 7.2 6.9 4.4 ...
 $ zinc: num  1022  257  346  183 1096 ...
```

We can see which observations were selected:

R code:
```
rownames(meuse.pb)
```

R console output:
```
  [1] "1"   "4"   "7"   "10"  "13"  "16"  "19"  "22"  "25"  "28"  "31"  "34"  "38"  "41"  "44"
[16] "47"  "50"  "53"  "56"  "59"  "62"  "65"  "69"  "79"  "82"  "85"  "88"  "123" "70"  "92"
[31] "95"  "98"  "101" "104" "108" "111" "114" "117" "120" "124" "127" "130" "133" "136" "137"
[46] "141" "144" "147" "150" "153" "156" "159"
```

Since we will work with the $\log_{10}$ transformed variables, for convenience we compute these once and add them to the subsample data frame.

Task 5 : Add fields with the $\log_{10}$ transformed target and co-variables to the data frame of the subsample. •

We name the new fields `ltpb` (stands for "log-ten-lead"), `ltom` (stands for "log-ten-organic-matter"), and `ltzn`:

R code:
```
meuse.pb <- cbind(meuse.pb,
                  ltpb = log10(meuse.pb$lead),
                  ltom = log10(meuse.pb$om),
                  ltzn = log10(meuse.pb$zinc))
str(meuse.pb)
```

The `cbind` method is used to add columns to a data frame. The column names may be specified on the left side of a = sign for each added column; otherwise they are named by the full expression, e.g. `log10(meuse.pb$zinc)` instead of `ltzn`.

R console output:
```
'data.frame':  52 obs. of  5 variables:
 $ x   : num  181072 181298 181165 181232 180874 ...
 $ y   : num  333611 333484 333370 333168 333339 ...
 $ lead: num  299 116 132 80 285 240 207 123 75 67 ...
 $ om  : num  13.6 8 9.2 6.3 15.4 16.2 13.7 7.2 6.9 4.4 ...
 $ ltpb: num  2.48 2.06 2.12 1.90 2.45 ...
 $ ltom: num  1.134 0.903 0.964 0.799 1.188 ...
 $ ltzn: num  3.01 2.41 2.54 2.26 3.04 ...
```

We have a set of 103 points at which lead was measured, but which we didn't use in the subset, either for modelling or interpolation. This can be used as a *validation dataset* to assess the performance of the interpolation; see §5.1. Here we set up the validation data set.

Task 6 : Make a data frame of the lead values at the extra points that were not included in the subsample and compare the descriptive statistics of the three sets: sample, extra, and full. •

Since we already have the subsample in frame `meuse.pb`, we can extract the samples we didn't choose for the subsample by comparing the sample with the full sample, using the `setdiff` method on the row names.

R code:
```
meuse.extra <- meuse[setdiff(rownames(meuse), rownames(meuse.pb)),
                 c("x", "y", "lead")]
meuse.extra <- cbind(meuse.extra, ltpb = log10(meuse.extra$lead))
str(meuse.extra)

summary(log10(meuse$lead)); sd(log10(meuse$lead))
summary(meuse.pb$ltpb); sd(meuse.pb$ltpb)
summary(meuse.extra$ltpb); sd(meuse.extra$ltpb)
```

R console output:

```
'data.frame': 103 obs. of  4 variables:
 $ x   : num  181025 181165 181307 181390 181027 ...
 $ y   : num  333558 333537 333330 333260 333363 ...
 $ lead: num  277 199 117 137 150 133 86 97 183 130 ...
 $ ltpb: num  2.44 2.30 2.07 2.14 2.18 ...

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.57    1.86    2.09    2.09    2.32    2.82
[1] 0.28944
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.68    1.83    2.14    2.12    2.39    2.68
[1] 0.30808
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.57    1.86    2.07    2.07    2.29    2.82
[1] 0.27991
```

The subsample has very similar statististics to the full sample. As expected, the range is a bit narrower but the standard deviation a bit higher; by chance the median and mean are a bit larger. The validation set has almost identical statistics to the full set.

4 Modelling the spatial structure of the target variable

We now begin *spatial* analysis. The first step is to convert the data sets to explictly-spatial, using the `sp` package.

As loaded from the `sp` package, the sample data `meuse` is a `data.frame`. Notice that the coördinates of each point are listed as fields. However, `sp` has defined some classes to make the spatial nature of the data explicit. It also provides a `coordinates` method to set the spatial coördinates and thereby create explict spatial data.

Task 7 : Convert the full and subset data frames, and the interpolation grid, to `sp` classes. •

R code:

```
class(meuse)
coordinates(meuse) <- ~ x + y
    # alternate command format: coordinates(meuse) <- c("x", "y")
coordinates(meuse.pb) <- ~ x + y
coordinates(meuse.extra) <- ~ x + y
coordinates(meuse.grid) <- ~ x + y
class(meuse)
```

R console output:
```
[1] "data.frame"

[1] "SpatialPointsDataFrame"
attr(,"package")
[1] "sp"
```

The `coordinates` method is used to inform `sp` about the spatial objects. The objects are now spatially-explicit; so for example they have a known bounding box, projection (here not defined), and attributes:

R code:
```
summary(meuse.pb)
```

R console output:
```
Object of class SpatialPointsDataFrame
Coordinates:
     min    max
x 178786 181298
y 329822 333611
Is projected: NA
proj4string : [NA]
Number of points: 52
Data attributes:
      lead             om             zinc           ltpb           ltom            ltzn
 Min.   : 48.0   Min.   : 1.90   Min.   : 117   Min.   :1.68   Min.   :0.279   Min.   :2.07
 1st Qu.: 67.8   1st Qu.: 5.45   1st Qu.: 190   1st Qu.:1.83   1st Qu.:0.736   1st Qu.:2.28
 Median :138.0   Median : 7.00   Median : 401   Median :2.14   Median :0.845   Median :2.60
 Mean   :167.1   Mean   : 7.78   Mean   : 517   Mean   :2.12   Mean   :0.847   Mean   :2.59
 3rd Qu.:243.2   3rd Qu.: 9.15   3rd Qu.: 738   3rd Qu.:2.39   3rd Qu.:0.961   3rd Qu.:2.87
 Max.   :482.0   Max.   :17.00   Max.   :1672   Max.   :2.68   Max.   :1.230   Max.   :3.22
                 NA's   : 1.00                                 NA's
                 :1.000
```

Many R methods are not yet aware of `sp` classes; for these the original data frame can be recovered when necessary with the `as.data.frame` method:

R code:
```
str(as.data.frame(meuse))
```

R console output:
```
str(as.data.frame(meuse))
‘data.frame’: 155 obs. of  14 variables:
 $ x      : num  181072 181025 181165 181298 181307 ...
 $ y      : num  333611 333558 333537 333484 333330 ...
 $ cadmium: num  11.7 8.6 6.5 2.6 2.8 3 3.2 2.8 2.4 1.6 ...
...
 $ dist.m : num  50 30 150 270 380 470 240 120 240 420 ...
```

Task 8 : Display a postplot of the subsample superimposed on the full sample and compare their geographic distribution. •

R code:
```
xyplot(y ~ x, as.data.frame(meuse), asp="iso",
       panel = function(x, ...) {
         panel.points(coordinates(meuse),
                 cex=1.8*(log10(meuse$lead) - 1.3),
                 pch=1, col="blue");
         panel.points(coordinates(meuse.pb),
                 cex=1.8*(meuse.pb$ltpb - 1.3),
                 pch=20, col="red");
         panel.grid(h=-1, v=-1, col="darkgrey")
       })
```

Note: The `xyplot` method is part of the `lattice` graphics package. It allows the use of *panel functions* to superimpose different kinds of plots, in this case two point sets and the grid lines on top of the main x-y (scatter) plot. Each `panel.*` method can have optional graphics parameters; in this case we used `cex=` to specify maximum point size, `pch=` to specify the printing character shape, and `col=` to specify the character colour. Colours may be given by an RGB code or by name, see `?colors` and `?col2rgb`. The `aspect="iso"` argument scales the map so that the same distance in the N–S and E–W directions is represented by the same distance on the map.

The subsample has a similar but sparser geographic distribution to the full sample (Fig. 3).

The first step in any kriging prediction is to model the regionalised target variable. In this case the sample set is small for variogram modelling; it is certainly too small to model anisotropy.

Task 9 : Compute the omnidirectional spatial variogram of $\log_{10}$Pb and

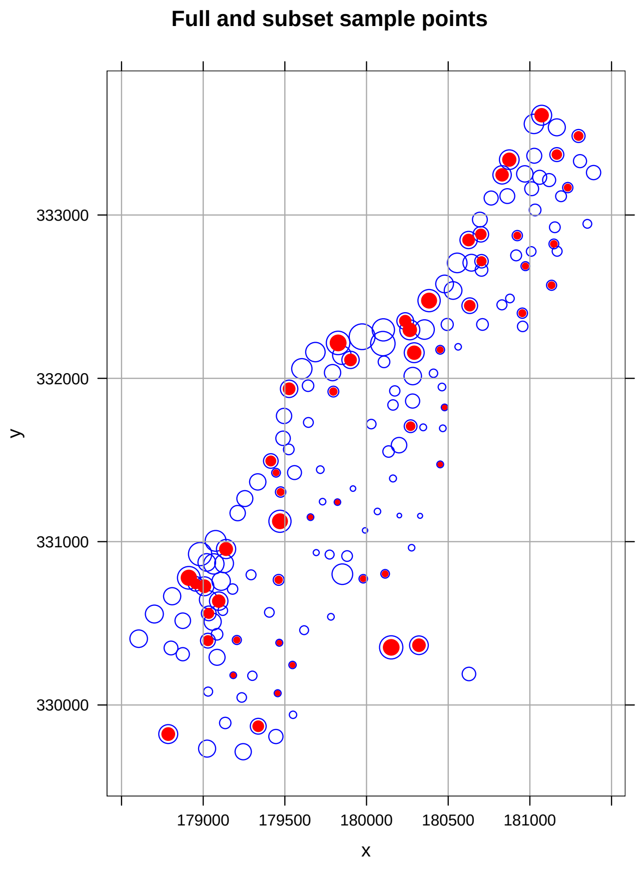

Figure 3: Sampling points in the full data set (blue) and the subset (red)

fit a variogram model to it. •

`Gstat` provides methods for each of these steps:

1. The `variogram` method computes the experimental variogram as a variogram object;
2. The `vgm` method creates a variogram model object;
3. The `fit.variogram` method adjusts a variogram model object to a variogram object.

First we examine the variogram cloud (Fig. 4 left) to see all the point-pairs; then we compute the experimental variogram. Then we specify a variogram model and its initial parameters (determined by eye looking at the experimental variogram); finally we fit it (Fig. 4 right).

> The `variogram` method has optional arguments to specify maximum distance, bin widths, directions, directional tolerances, and whether the robust Cressie estimator should be used. In this case

we use the default estimator but specify the bin widths and cutoff.

R code:
```
     # variogram cloud
v.ltpb.c <- variogram(ltpb  ~ 1, data=meuse.pb, cutoff=1800, cloud=T)
     # experimental variogam
v.ltpb <- variogram(ltpb  ~ 1, data=meuse.pb, cutoff=1800, width=200)
plot(v.ltpb, pl=T)
     # estimate variogram model form and parameters by eye
m.ltpb <- vgm(0.08,"Sph",800,0.03)
plot(v.ltpb, pl=T, model=m.ltpb)
     # fit model parameters by weighted least-squares
(m.ltpb.f <- fit.variogram(v.ltpb, m.ltpb))
plot(v.ltpb, pl=T, model=m.ltpb.f)
rm(v.ltpb.c)
```

R console output:
```
  model      psill    range
1   Sph 0.08879807 876.1765
2   Nug 0.02395032   0.0000
```

The initial variogram parameters (Spherical form, total sill 1.1, range 800, nugget 0.03) are estimated by eye. The form is suggested not only by this experimental variogram but also by previous experience with soil chemical values. The automatic fit did not change the initial parameter estimates much. For a small sample, the variogram has a good structure, and was successfully modelled.

5 Ordinary Kriging interpolation of the target variable

First, we predict Pb without the co-variables, from the small subset. This simulates the situation where we have a small sample of the expensive-to-measure target variable.

Task 10 : Use the fitted model of regionalization to interpolate the $\log_{10}$Pb content with OK on the prediction grid. Summarize the prediction and error; we will use this as a baseline to evaluate the benefits of co-kriging. •

`Gstat` provides the `krige` method for univariate kriging. We must supply a *formula* for the spatial dependence (here, `ltpb ~ 1` for a stationary field), the object containing the sample data and from which the coordinates can be extracted (here, `meuse.pb`), an object with locations to

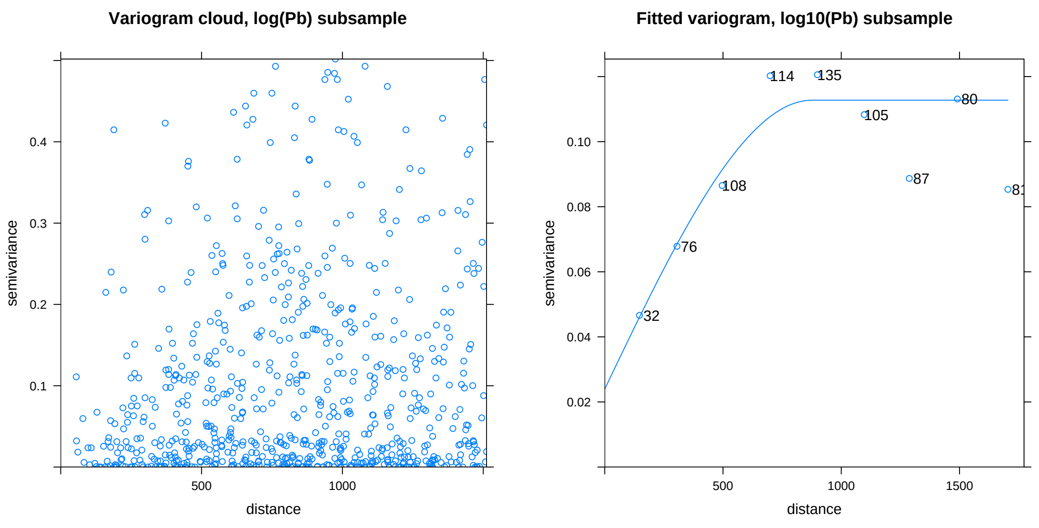

Figure 4: Variogram cloud and experimental variogram with fitted model, log10(Pb)

be predicted (here, `meuse.grid`) and named coordinates as in the sample data object, and a model of spatial dependence (variogram) (here, `m.ltpb.f`):

R code:

```
        # interpolate
k.o <- krige(ltpb ~1, locations=meuse.pb, newdata=meuse.grid, model=m.ltpb.f)
        # summary statistics
summary(k.o)
```

R console output:
```
Object of class SpatialPointsDataFrame
Coordinates:
     min    max
x 178460 181540
y 329620 333740
Is projected: NA
proj4string : [NA]
Number of points: 3103
Data attributes:
   var1.pred       var1.var
 Min.   :1.71   Min.   :0.0357
 1st Qu.:1.90   1st Qu.:0.0519
 Median :2.03   Median :0.0608
 Mean   :2.06   Mean   :0.0638
 3rd Qu.:2.23   3rd Qu.:0.0738
 Max.   :2.53   Max.   :0.1119
```

The kriged object has both the predictions (field `var1.pred`) and variances of the predictions (field `var1.var`).

Task 11 : Display a map of the predictions and their errors. •

A grid map can be displayed with the `levelplot`[4] method of the `lattice` package. Here we use panel functions to superimpose the sample points, as well as the extra points not included in the sample. For the predictions, we draw a postplot (size of symbols proportional to value); for the errors only the positions, since kriging error depends only on point configuration. The production of these plots has been encapsulated in several plotting functions which make use of `levelplot`; they are presented in §A.1 and also provided as source code in file `ck_plotfns.R`, which must be loaded at this point:

R code:
```
source("ck_plotfns.R")
plot.kresults(k.o, "var1", meuse, meuse.pb, "lead", "log10(Pb), OK")
```

The OK map (Fig. 5 left) shows the main features of the spatial distribution; however it is clear that some of the sample points that were not included in the sub-sample are poorly predicted. (See for example the extra point near $(179750E, 330800N)$; the prediction is low, but the large green point shows that the value is large.) The OK prediction error map (Fig. 5 right) shows the expected low errors near the sample points. There are extra points near most high-error areas; using information

[4] and `contourplot()`

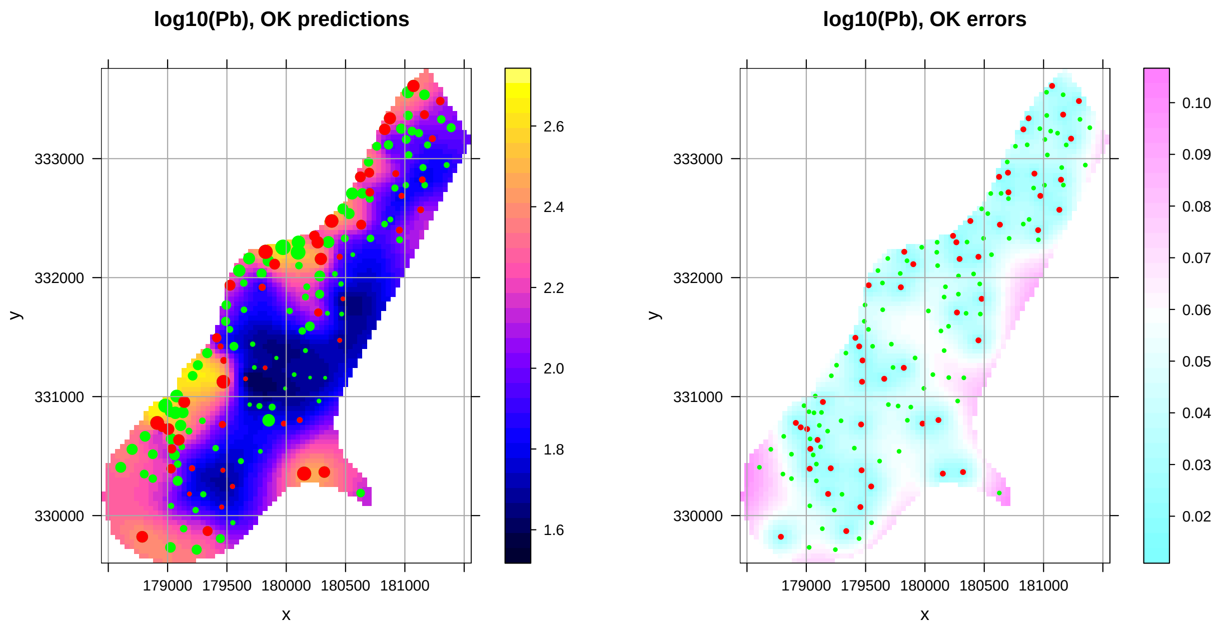

Figure 5: OK predictions and error, log10(Pb); subsample points red, extra points green

on the co-variable from these points should improve the prediction and lower its error.

5.1 Validation

The `meuse.extra` object has the coördinates and lead levels of the 103 points at which lead was measured, but which we didn't use in the subset, either for modelling or interpolation. This can be used as a *validation dataset* to assess the performance of the interpolation.

The procedure is to predict (by kriging) at the points in this validation set `meuse.extra`, using the sample points only. Then we compare the predictions with the measured values. We work with the log10-transformed values.

Task 12 : Interpolate at the extra points, using OK from the sample points, and determine the actual prediction bias and precision. •

R code:

```
        # predict at the extra points
k <- krige(ltpb  ~ 1, meuse.pb, meuse.extra, m.ltpb.f)
        # compute and summarize validation errors
summary(k)
diff <- k$var1.pred - meuse.extra$ltpb
summary(diff)
sqrt(sum(diff^2)/length(diff))   # RMSE (precision)
sum(diff)/length(diff)           # mean error (bias)
median(meuse.extra$ltpb)         # median error
```

R console output:

```
Object of class SpatialPointsDataFrame
Coordinates:
     min    max
x 178605 181390
y 329714 333558
Is projected: NA
proj4string : [NA]
Number of points: 103
Data attributes:
   var1.pred       var1.var
 Min.   :1.74   Min.   :0.0402
 1st Qu.:1.92   1st Qu.:0.0526
 Median :2.11   Median :0.0580
 Mean   :2.10   Mean   :0.0606
 3rd Qu.:2.26   3rd Qu.:0.0665
 Max.   :2.46   Max.   :0.1033

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.6630 -0.0647  0.0215  0.0250  0.1270  0.3720

[1] 0.16597
[1] 0.02502
[1] 2.0682
```

There is a slight positive bias: the median and mean predictions at the extra points by OK is higher than the true value, as is the mean error (+0.025). The overall precision is good (RMSE = 0.166).

Task 13 : Display a postplot of the errors. •

We can use the `bubble` method of the `gstat` library to plot spatial points. It requires two arguments: the spatial object (here, a kriging interpolation) and the field in the object to plot, in this case the difference. Positive values are plotted in green and negative in red, with the size of the "bubble" proportional to the distance from zero.

Note: This has also been included in the `plot.valids` function listed in §A.1; this also plots the cross-validation (see next §§ and Figure 7). Since we haven't yet computed the cross-validation, we just look at the validation residuals. The `bubble` method requires a spatial object, so we first have to convert the differences, assigning them the coördinates from the validation points:

R code:
```
diff <- as.data.frame(diff)
coordinates(diff) <- coordinates(meuse.extra)
bubble(diff, zcol="diff", main="OK validation errors at undersampled points, log10(Pb)")
```

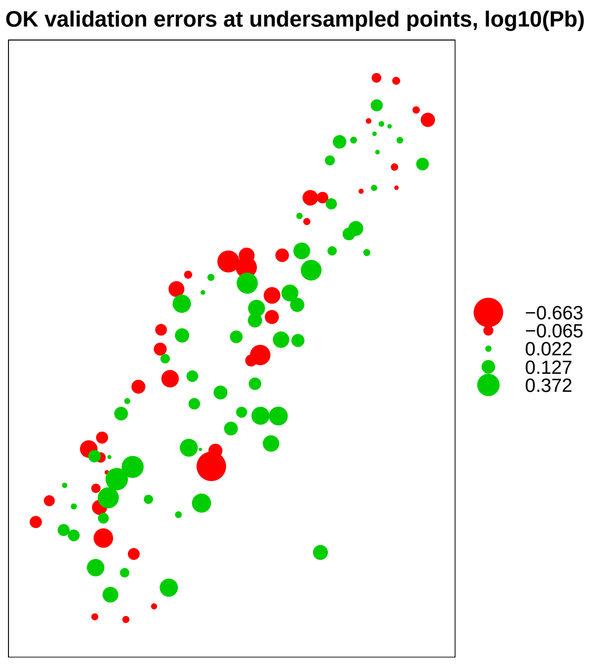

Figure 6: OK validation errors of log10(Pb)

Figure 6 shows the validation errors at the extra points; There are several large relative errors (from -0.66 to $+0.37$) compared to the median value for the extra points of 2.07. Note the very large under-prediction at the extra point near $(179750E, 330800N)$ which we noticed in the OK map (Fig. 5 left).

5.2 Cross-validation

We can also evaluate the success of OK by *cross-validation* at the 52 points in the subset. Each point is held out in turn, and the prediction at that point is made from the remaining 51 points, using the common variogram model.

Diagnostic measures are the ME (bias), RMSE (precision), and Mean Squared Deviation Ratio (MSDR) of the residuals to the prediction errors; this should be 1 because the residuals from cross-validation should equal to prediction errors at each point that was held out. For the univariate case (OK) we can use the `krige.cv` cross-validation method of the `gstat` package.

Task 14 : Cross-validate the OK predictions of $\log_{10}$Pb and compute the diagnostic measures. •

R code:
```
cv.o <- krige.cv(ltpb ~ 1, meuse.pb, model=m.ltpb.f, nfold=nrow(meuse.pb))
summary(cv.o)
res <- as.data.frame(cv.o)$residual
sqrt(mean(res^2))
mean(res)
mean(res^2/as.data.frame(cv.o)$var1.var)
rm(res)
```

The `nfold` argument specifies the number of times to do the cross-validation, in this case once for each point.

R console output:
```
Object of class SpatialPointsDataFrame
Coordinates:
     min    max
x 178786 181298
y 329822 333611
Is projected: NA
proj4string : [NA]
Number of points: 52
Data attributes:
   var1.pred       var1.var          observed       residual            zscore
 Min.   :1.78   Min.   :0.0412   Min.   :1.68   Min.   :-0.46412   Min.   :-1.86785
 1st Qu.:1.96   1st Qu.:0.0559   1st Qu.:1.83   1st Qu.:-0.15405   1st Qu.:-0.57469
 Median :2.12   Median :0.0621   Median :2.14   Median :-0.05570   Median :-0.22692
 Mean   :2.11   Mean   :0.0643   Mean   :2.12   Mean   : 0.00367   Mean   : 0.00707
 3rd Qu.:2.27   3rd Qu.:0.0729   3rd Qu.:2.39   3rd Qu.: 0.16694   3rd Qu.: 0.69951
 Max.   :2.49   Max.   :0.1131   Max.   :2.68   Max.   : 0.65826   Max.   : 2.68334
      fold
 Min.   : 1.0
 1st Qu.:13.8
 Median :26.5
 Mean   :26.5
 3rd Qu.:39.2
 Max.   :52.0

[1] 0.23691
[1] 0.0036705
[1] 0.88457
```

The RMSE is 0.237, the mean error is almost zero, and the Mean Squared Deviation Ratio (MSDR) of the predictions vs. the sample is a bit lower than the ideal 1 (0.88), meaning that the predictions are somewhat less variable than reality; this is to be expected, as kriging is a smoothing estimator.

Task 15 : Display a postplot of the errors. •

This is included in the `plot.valids` function, which also shows the validation errors from the previous § on the left side plot:

R code:
```
plot.valids(k, "var1", meuse.extra, "ltpb", cv.o, "OK")
```

Figure 7 (right) shows the cross-validation errors at the sample points. There are some very large errors, from −0.46 to +0.66.

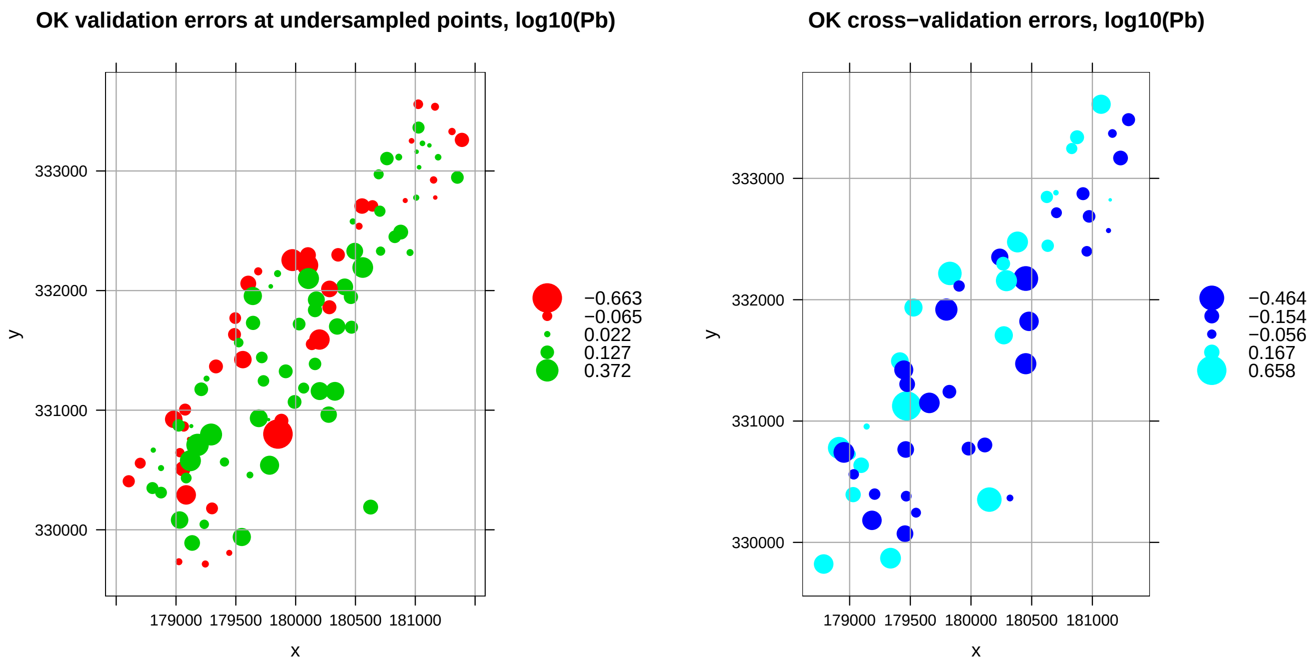

Figure 7: OK validation (left) and cross-validation (right) errors of log10(Pb)

5.3 Evaluating errors in absolute terms

The maximum absolute residual for both validations was 0.66 $\log_{10}$Pb; one might naïvely back-transform this as $10^{0.66} = 4.6$ mg kg^{-1} Pb, an order of magnitude smaller than the smallest data values. However, this conversion is not correct, because the error is *in addition to* the predicted value, which is of another order of magnitude. For example, the mean predictions are about 2.1 $\log_{10}$Pb, i.e. about $10^{2.1} = 125.9$ mg kg^{-1} Pb; at this level an error of 0.66 $\log_{10}$Pb corresponds to about $10^{2.1+0.66} - 10^{2.1} = 450$ mg kg^{-1} Pb.

So, to evaluate the severity of the errors, we must compare original Pb values with predicted Pb values, *not* log-transformed values (although these were used for interpolation).

Task 16 : Compute and summarize the absolute validation and cross-validation errors in units of mg kg^{-1} Pb. •

R code:

```
diff <- 10^(k$var1.pred) - 10^(meuse.extra$ltpb)
summary(diff)
histogram(diff, col="darkseagreen2", nint=12, type="count",
    main="Validation errors", xlab="Pb, ppm")
diff <- 10^(cv.o$var1.pred) - 10^(meuse.pb$ltpb)
summary(diff)
histogram(diff, col="lightblue2", nint=12, type="count",
    main="Cross-Validation errors", xlab="Pb, ppm")
rm(diff)
```

R console output:

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-368.00  -23.30    6.66   -7.20   27.20  110.00

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -311.0   -63.4    12.3   -24.6    34.4   158.0
```

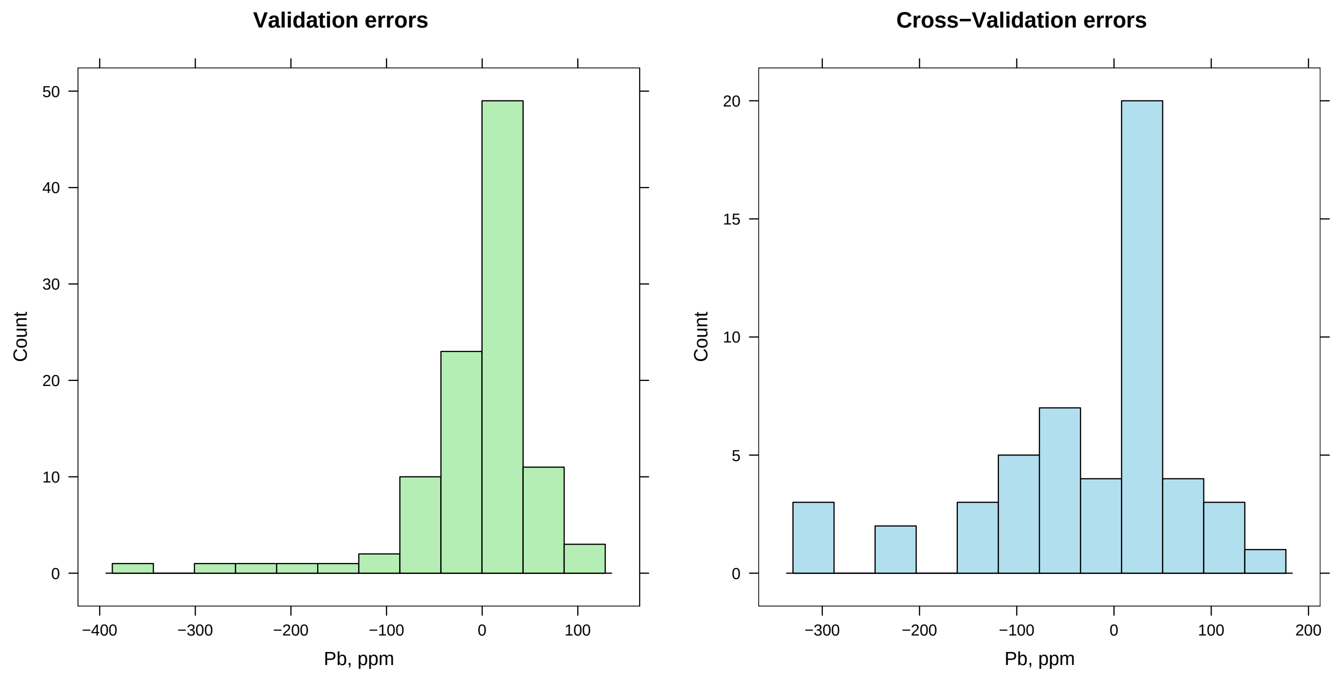

Figure 8: Histograms of OK validation (left) and cross-validation (right) errors of Pb

Figure 8 shows the histograms of the absolute validation and cross-validation errors. There are some serious errors; in particular the high negative residuals are points with high Pb which are seriously underpredicted; recall that 100 mg kg^{-1} is the level at which agricultural crops must be tested.

In summary, OK with this small dataset did not perform so well; this is the motivation for trying to improve the predictions with co-kriging.

6 Modelling a bivariate co-regionalisation

In order to perform co-kriging, we must first *model the spatial structure* of a co-variable and its covariance with the target variable. This is called a *co-regionalisation*. It is an extension of the theory of a single regionalised variable used for ordinary kriging. This is the same sequence we followed for OK: we first modelled the spatial structue of the target variable (§4) and then used this model to interpolate by OK (§5).

In this section we use the organic matter content as the co-variable. This is cheap to sample and determine in the laboratory, and in practice would be a good choice if it turns out to give better predictions. We will see that this isn't the case, so in a later section (§9) a different co-variable, zinc content, is used.

6.1 Modelling the co-variable

First we examine the feature-space correlation.

Task 17 : Display a scatterplot of the co-variable (organic matter) vs. the target variable (Pb) and compute the correlation coefficient. Describe the feature-space relation. •

For this task we can only use the points from the subsample, since we are simulating the situation where the target variable is under-sampled.

R code:

```
attach(as.data.frame(meuse.pb))
xyplot(ltpb ~ ltom,  pch=20, cex=1.2,
       col="blue", ylab="log10(Pb)", xlab="log10(OM)")
cor(ltom, ltpb)
sum(is.na(om))
cor(ltom, ltpb, use = "complete")
```

R console output:

```
Error in cor(om, ltpb) : missing observations in cov/cor

[1] 1

[1] 0.61277
```

The `cor` method does not, by default, compute the correlation coefficient if there are any missing values in either vector. Here there is one missing value of organic matter in the sub-sample (detected by the `is.na` method). The `use = "complete"` optional argument to `cor` specifies that only complete observations (with values for both variables) be used to compute the correlation.

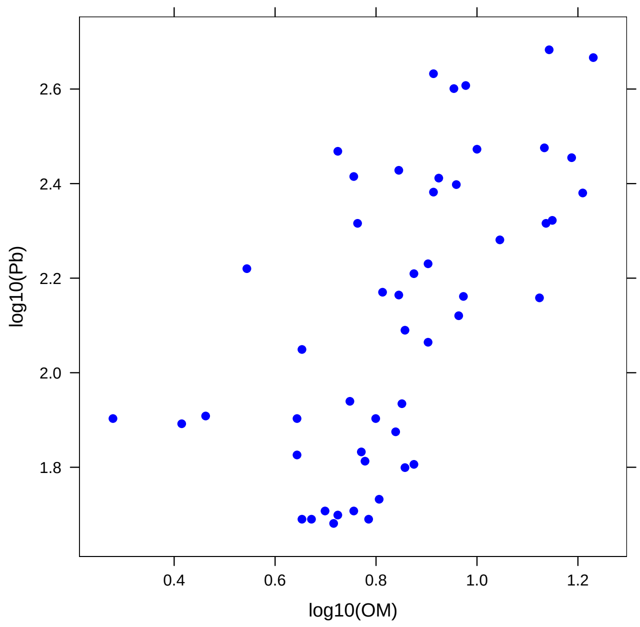

Figure 9: Feature-space relation between Pb and OM, subsample

The scatterplot (Fig. 9) and numerical correlation both show that there is a positive feature-space relation between the target and co-variable. However, the relation only explains $0.61277^2 = 0.37549$ or about 38% of the total variability; this can be seen in the “cloud”-like form of the scatterplot. We will see if this is too weak for a successful co-kriging.

Task 18 : Model the omnidirectional spatial variogram of the $\log_{10}$-transformed covariable. We do not intend to interpolate it; however we need to know its spatial structure for co-kriging. Compare the structure with that of the target variable. •

First we have to make a data frame without the missing values of organic matter, then we can compute the variogram. The `subset` method is convenient for this; its second argument is the condition (in this case, to omit the missing values) and its third argument is the list of fields to include. All available samples are used for this model, not just the subset. This simulates the realistic situation where the target variable is under-sampled but the co-variable has a full sample.

R code:

```
        # all valid covariable observations, with coordinates
meuse.co <- subset(as.data.frame(meuse), !is.na(om), c(x, y, om))
        # add log10-transformed variables for convenience
meuse.co <- cbind(meuse.co, ltom = log10(meuse.co$om))
str(meuse.co)
        # convert to spatial object
coordinates(meuse.co) <- ~ x + y
        # experimental variogram
v.ltom <- variogram(ltom ~ 1, meuse.co, cutoff=1800)
plot(v.ltom, pl=T)
        # model by eye
m.ltom <- vgm(.035, "Sph", 800, .015)
        # fit
(m.ltom.f <- fit.variogram(v.ltom, m.ltom))
plot(v.ltom, pl=T, model=m.ltom.f)
        # compare variogram structure to target variable
m.ltom.f$range[2]; m.ltpb.f$range[2]
round(m.ltom.f$psill[1]/sum(m.ltom.f$psill),2)
round(m.ltpb.f$psill[1]/sum(m.ltpb.f$psill),2)
```

R console output:

```
'data.frame': 153 obs. of  5 variables:
 $ x   : num  181072 181025 181165 181298 181307 ...
 $ y   : num  333611 333558 333537 333484 333330 ...
 $ om  : num  13.6 14 13 8 8.7 7.8 9.2 9.5 10.6 6.3 ...
 $ ltom: num  1.134 1.146 1.114 0.903 0.940 ...

  model    psill  range
1   Nug 0.010672   0.00
2   Sph 0.038599 641.62

[1] 641.62
[1] 876.18

[1] 0.22
[1] 0.21
```

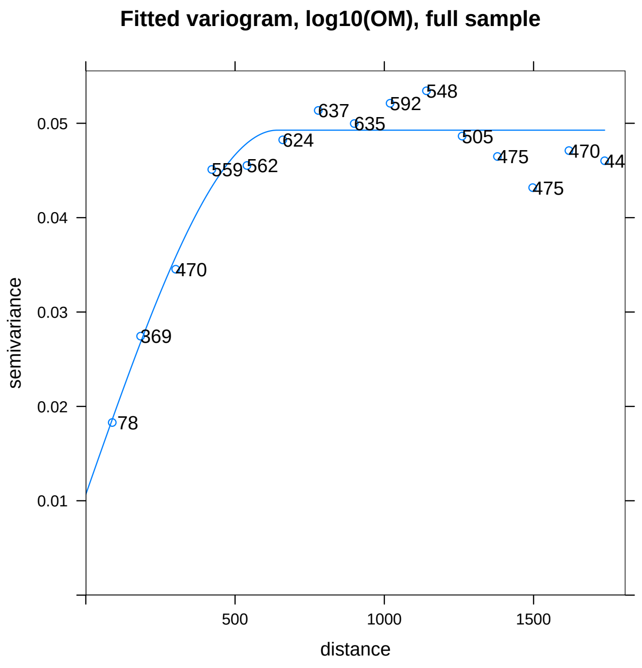

Figure 10: Experimental variogram with fitted model, log10(OM), full data set

The structure for $\log_{10}$OM (Fig. 10) is quite similar to that for $\log_{10}$Pb (Fig 4). A spherical model could be fitted in both cases, with an almost identical nugget effect (about 1/5 of the total sill). However, the range for OM is only about 3/4 of that for Pb (641 vs. 876 m); it has a more local structure. The linear model of co-regionalisation requires the ranges of the target and co-variable be the same; here we see they are not too different.

6.2 Building a data structure to model co-regionalisation

Now comes the hardest part of co-kriging: modelling the co-regionalisation. We have to fit models to both the direct and cross-variograms simultaneously, and these models must lead to a positive definite cokriging system. The easiest way to ensure this is to fit a *linear model of co-regionalisation*: all models (direct and cross) have the same shape and range, but may have different partial sills and nuggets.

It is possible to model any number of variables together; in this case we have three (one target and two co-variables). The modelling and subsequent co-kriging in fact treat all the variables equally: they are all modelled (direct and cross) and predicted.

We will explore the simplest case: one target variable (here, Pb) and one co-variable (here, OM). In a later section (§9.2) we will use the other co-variable (Zn).

Task 19 : Build a gstat structure containing the two sample sets: subsample for Pb and full sample for OM. •

Up till now we've worked with data frames, variograms, and variogram models as separate objects, each with its own class. This is sufficient for univariate geostatistics. For multivariate analysis we must work with the more general gstat object; this allows all the flexibility of the stand-alone gstat program but within the R environment.

First, we build gstat objects to specify the two experimental variograms that we computed above. The object has named subframes, and is added to as we go. In the ordinary kriging case we used a 'wrapper' method, variogram, for the simple case of a single variogram. Here we have to explicitly build the gstat model piece-by-piece, using the gstat method. This allows us to build complicated structures both for modelling and interpolation. It works on an object of class gstat that is a *list* of *frames*; each frame has a separate model or interpolation result.

We fill the first frame within the gstat object with the $\log_{10}$Pb sample observations, only at the subsample points, and the second for the $\log_{10}$OM sample observations, at all the points where it was measured.

> Note how the first command declares that there is no existing object, whereas the second command begins by naming the existing object, so that gstat will add onto it. Note also how each command uses the id= argument to name the frame within the object. Here we have one frame for Pb (id="ltpb") and one for OM (id="ltom").

R code:
```
(g <- gstat(NULL, id = "ltpb", form = ltpb ~ 1, data=meuse.pb))
(g <- gstat(g, id = "ltom", form = ltom ~ 1, data=meuse.co))
```

R console output:
```
data:
ltpb : formula = ltpb`~`1 ; locations = ~x + y ; data dim = 52 x 6

data:
ltpb : formula = ltpb`~`1 ; locations = ~x + y ; data dim = 52 x 6
ltom : formula = ltom`~`1 ; locations = ~x + y ; data dim = 153 x 2
```

The gstat object now has two frames, both of which are data observation with their associated formulas and locations. From these, both direct and cross-variograms can be computed and plotted.

Task 20 : Compute and display the two direct variograms and one cross-variogram. •

R code:
```
v.cross <- variogram(g)
str(v.cross)
plot(v.cross, pl=T)
```

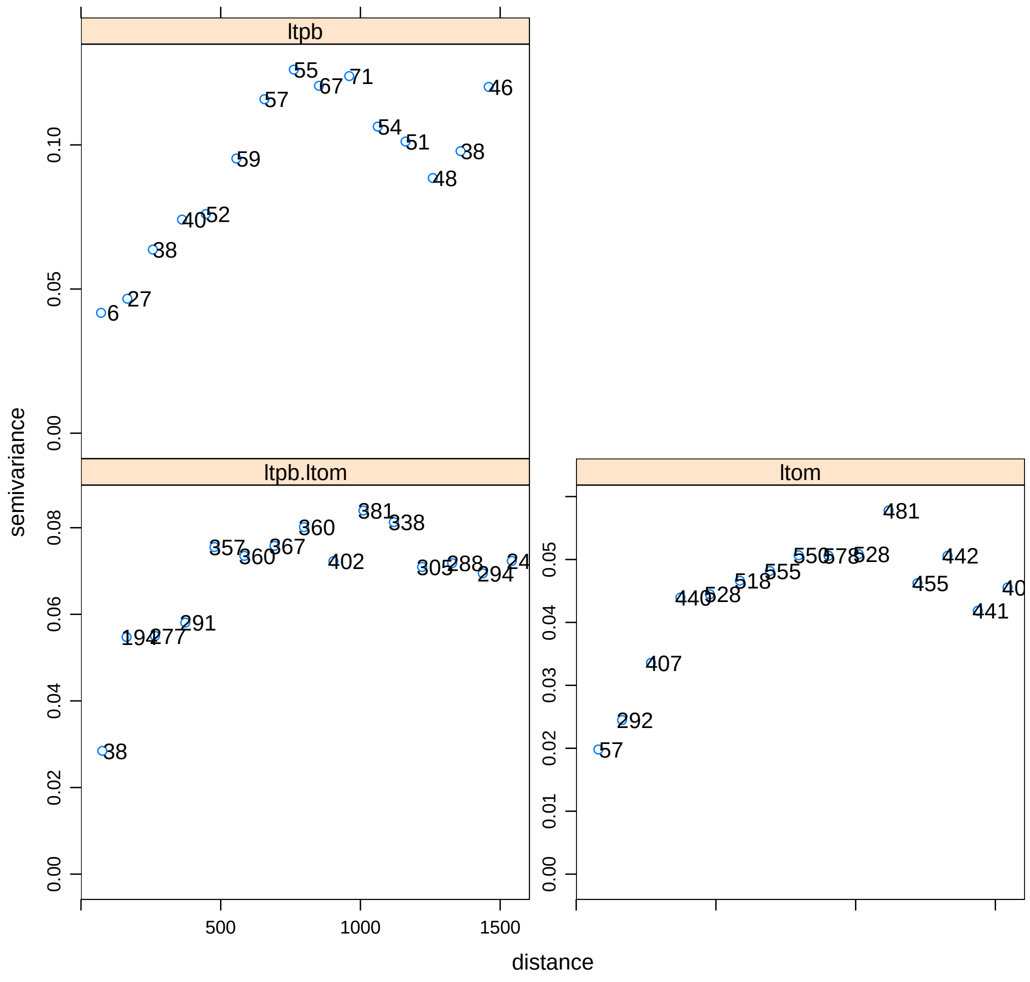

Figure 11: Direct and cross variograms, log10(Pb) and log10(OM)

R console output:

```
Classes gstatVariogram  and ‘data.frame’:      45 obs. of  6 variables:
 $ np     : num  38 194 277 291 357 360 367 360 402 381 ...
 $ dist   : num   76.4 163.1 264.9 373.1 477.4 ...
 $ gamma  : num  0.0285 0.0548 0.0550 0.0581 0.0755 ...
 $ dir.hor: num  0 0 0 0 0 0 0 0 0 0 ...
 $ dir.ver: num  0 0 0 0 0 0 0 0 0 0 ...
 $ id     : Factor w/ 3 levels "ltpb.ltom","ltom",..: 1 1 1 1 1 1 1 1 1 1 ...
 - attr(*, "direct")=‘data.frame’:      3 obs. of  2 variables:
  ..$ id       : Factor w/ 3 levels "ltom","ltpb",..: 3 1 2
  ..$ is.direct: logi  FALSE  TRUE  TRUE
```

The variogram object has a field `id`, which identifies the variogram (direct or cross, using the same identifiers as the `gstat` object used to specify the variogram) to which the semi-variance refers; we can also see whether the variogram estimate is from a direct or cross-variogram. Note in Figure 11 the larger number of point pairs for each lag for the co-variable compared to the target variable; the cross-variogram is intermediate between these.

> `Gstat` does not compute the cross-variogram as strictly defined, but rather the *pseudo-cross-variogram* [18, p. 217-8]. The cross-variogram only uses co-located points, whereas the pseudo-cross-variogram uses all point pairs which can be formed from the target and co-variables. This gives many more point pairs with which to estimate the co-regionalisation.

We have already computed the variograms for $\log_{10}$Pb (§4) and $\log_{10}$OM (§6.1) and determined that their structures and ranges were similar. The cross-variogram in this case also has a similar structure to the direct variograms; the nugget effect appears to be about 1/5 of the total sill. The range seems comparable to the range of the direct variogram of the target variable (about 850 m). Since the linear model of coregionalization requires equal ranges, this is good.

6.3 Fitting a linear model of co-regionalisation

The next step is to add variogram models to the `gstat` object and then fit these.

Task 21 : Add variogram models to the `gstat` object and fit a them using the linear model of co-regionalisation. •

We use the fitted model for the target value as a starting point for all three variogram models. This is because we will use the linear model of co-regionalisation, which requires a single range and structure. By filling all the frames with one model (using the `fill.all = T` argument), these conditions are automatically met.

R code:
```
(g <- gstat(g, id = "ltpb", model = m.ltpb.f, fill.all=T))
```

The `gstat` object now contains both the data and the models:

R console output:
```
data:
ltpb : formula = ltpb`~`1 ; locations = ~x + y ; data dim = 52 x 4
ltom : formula = ltom`~`1 ; locations = ~x + y ; data dim = 153 x 3
variograms:
             model    psill  range
ltpb[1]        Nug 0.023950   0.00
ltpb[2]        Sph 0.088798 876.18
ltom[1]        Nug 0.023950   0.00
ltom[2]        Sph 0.088798 876.18
ltpb.ltom[1]   Nug 0.023950   0.00
ltpb.ltom[2]   Sph 0.088798 876.18
```

Models can be specified individually for each direct and cross-variogram; however, it is difficult to ensure that the resulting CK system is positive-definite. For example:

```
g <- gstat(g, id=c("ltpb","ltom"), model=vgm(0.02,"Sph",800,0.08))
```

could be used to specify the cross-variogram. Note how the two variables whose cross-variogram is to be specified are named in a list.

Now we fit all three variograms together, ensuring they lead to a positive definite co-kriging system. For this we use the `fit.lmc` method ("fit linear model of co-regionalization"). This takes the initial estimate, fits all the variograms, and then each of the partial sills is adjusted (by least squares) to the closest value that will result in a positive definite matrices.

R code:
```
(g <- fit.lmc(v.cross, g))
plot(variogram(g), model=g$model)
```

R console output:

```
data:
ltpb : formula = ltpb`~`1 ; locations = ~x + y ; data dim = 52 x 4
ltom : formula = ltom`~`1 ; locations = ~x + y ; data dim = 153 x 3
variograms:
                model     psill   range
ltpb[1]           Nug 0.031606    0.00
ltpb[2]           Sph 0.084308  876.18
ltom[1]           Nug 0.021307    0.00
ltom[2]           Sph 0.038028  876.18
ltpb.ltom[1]      Nug 0.025951    0.00
ltpb.ltom[2]      Sph 0.051444  876.18
```

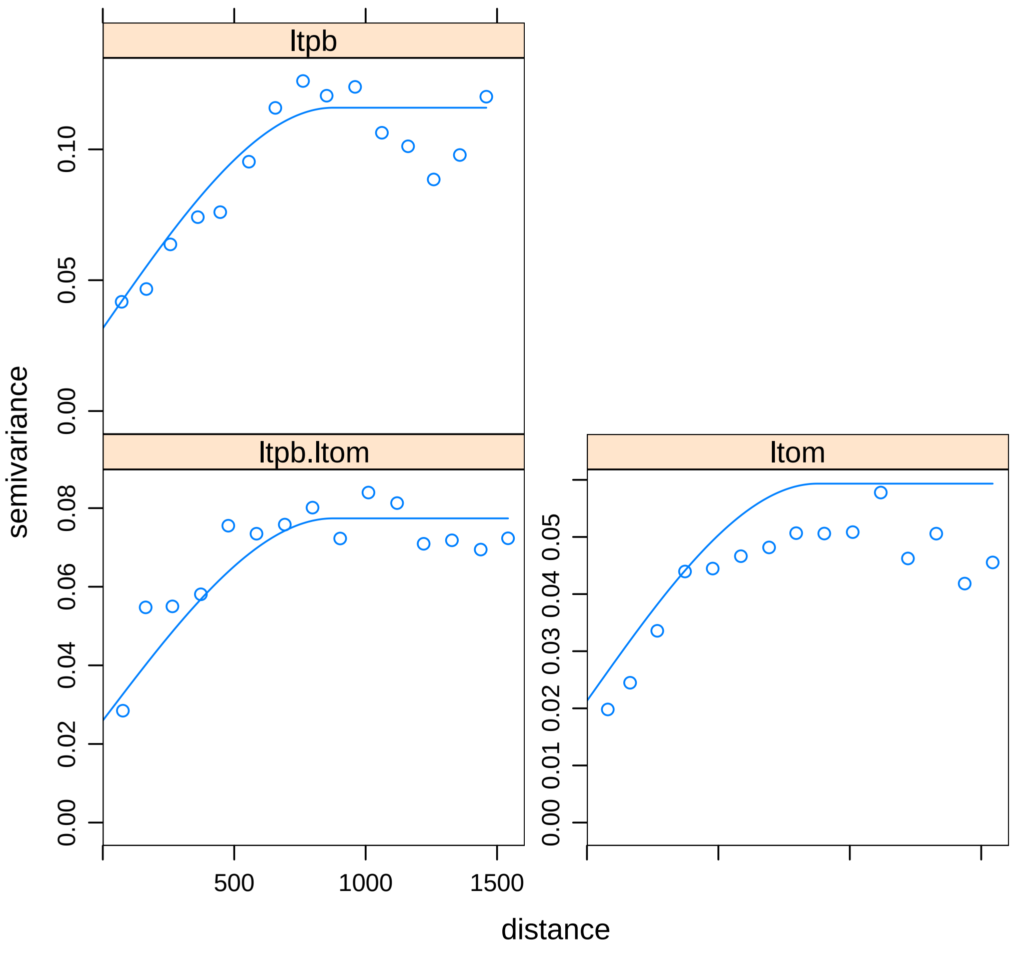

Figure 12: Fitted direct and cross variograms, log10(Pb) and log10(OM)

The model fits fairly well (Fig. 12), although the sill for $\log_{10}$OM seems a bit high and its range too long. Note that the ranges are not adjusted by `fit.lmc`, however all the partial sills (both for the spherical model and for the nugget) of the co-variable and cross-variogram were adjusted. So, the fitted variograms have the same range but different sills and nuggets.

6.4 Comparing models of regionalisation and co-regionalisation

We can compare the two direct variograms (for subset Pb, §4, and full set OM, §6.1) with the parameters that we determined for each one separately, to see how much fitting the linear model of co-regionalisation adjusted the fit to each variable separately.

First, we have to examine the structure of the model objects (the `gstat` object `g` and the variogram model objects `m.ltom.f` and `m.ltpb.f`) to see where the parameters are stored; then we can compute their differences.

We begin with the single variogram model for OM:

R code:
```
str(m.ltom.f)
```

R console output:
```
Classes variogramModel  and ‘data.frame’: 2 obs. of  9 variables:
$ model: Factor w/ 17 levels "Nug","Exp","Sph",..: 1 3
 $ psill: num  0.0107 0.0386
 $ range: num    0 642
 $ kappa: num  0.5 0
 $ ang1 : num  0 0
 $ ang2 : num  0 0
 $ ang3 : num  0 0
 $ anis1: num  1 1
 $ anis2: num  1 1
 - attr(*, "singular")= logi FALSE
 - attr(*, "SSErr")= num 6.69e-08
```

This object is of class `variogramModel` and has several fields of interest, including the ranges `range` and partial sills `psill` for the different components of the variogram model. In this case there are two (as shown in the `model` field): the first is a spherical model with partial sill 0.0373 and range 653, and the second is a nugget mode with partial sill 0.0121 (and by definition range 0).

We now examine the structure of the `gstat` object; this is so complex that we look at it a level at a time:

R code:
```
str(g, max.level = 1)
```

R console output:
```
List of 3
 $ data :List of 2
 $ model:List of 5
 $ call : language gstat(g = g, id = "ltom", formula = ltom ~ 1, data = meuse.co)
 - attr(*, "class")= chr [1:2] "gstat" "list"
```

At the first level this is a list of three items: the data frames (here, for the two variables), the models (here, two direct and three cross-variograms and their parameters), and the most recent call.

Now we look at the structure of the list components:

R code:
```
str(g$data, max.level = 1)
str(g$model, max.level = 1)
```

R console output:
```
List of 2
 $ ltpb:List of 11
 $ ltom:List of 11

List of 5
 $ ltpb     :Classes variogramModel  and ‘data.frame’: 2 obs. of  9 variables:
  ..- attr(*, "singular")= logi FALSE
 $ ltpb.ltom:Classes variogramModel  and ‘data.frame’: 2 obs. of  9 variables:
  ..- attr(*, "singular")= logi FALSE
 $ ltom     :Classes variogramModel  and ‘data.frame’: 2 obs. of  9 variables:
  ..- attr(*, "singular")= logi FALSE
 $ ltom.NA  :Classes variogramModel  and ‘data.frame’: 2 obs. of  9 variables:
  ..- attr(*, "singular")= logi FALSE
 $ ltom.ltom:Classes variogramModel  and ‘data.frame’: 2 obs. of  9 variables:
  ..- attr(*, "singular")= logi FALSE
```

The data has two sub-lists and the models five. There are two cross-variograms we didn’t ask for explicitly: `ltom.NA` for the missing values of OM, and `ltom.ltom` for the OM values measured at different sampling locations (subset vs. full set).

R code:
```
str(g$data$ltpb, max.level = 1)
```

R console output:
```
List of 5
 $ ltpb     :Classes variogramModel  and ‘data.frame’: 2 obs. of  9 variables:
  ..- attr(*, "singular")= logi FALSE
  ..- attr(*, "SSErr")= num 1.57e-07
 $ ltpb.ltom:Classes variogramModel  and ‘data.frame’: 2 obs. of  9 variables:
  ..- attr(*, "singular")= logi FALSE
  ..- attr(*, "SSErr")= num 1.48e-06
 $ ltom     :Classes variogramModel  and ‘data.frame’: 2 obs. of  9 variables:
  ..- attr(*, "singular")= logi FALSE
  ..- attr(*, "SSErr")= num 2.32e-07
 $ ltom.NA  :Classes variogramModel  and ‘data.frame’: 2 obs. of  9 variables:
  ..- attr(*, "singular")= logi FALSE
  ..- attr(*, "SSErr")= num 1.11e-07
 $ ltom.ltom:Classes variogramModel  and ‘data.frame’: 2 obs. of  9 variables:
  ..- attr(*, "singular")= logi FALSE
  ..- attr(*, "SSErr")= num 1.11e-07
```

The data list item is itself a list with 11 items. One is the data object used to estimate the variogram for the `model` list components which use this variable. Another is the formula used for modelling. The others control the modelling for this variable, e.g. is there a trend, should a limiting distance be used, are their weights etc.; all these are explained in the help `?gstat`.

R code:
```
str(g$model$ltpb, max.level = 1)
```

R console output:
```
Classes variogramModel  and ‘data.frame’: 2 obs. of  9 variables:
 $ model: Factor w/ 17 levels "Nug","Exp","Sph",..: 1 3
 $ psill: num  0.0316 0.0843
 $ range: num    0 876
 $ kappa: num  0 0.5
 $ ang1 : num  0 0
 $ ang2 : num  0 0
 $ ang3 : num  0 0
 $ anis1: num  1 1
 $ anis2: num  1 1
 - attr(*, "singular")= logi FALSE
 - attr(*, "SSErr")= num 1.57e-07
```

The structure for the model within the `gstat` object is exactly that same as for the single variogram; it is of the same class: `variogramModel`.

Now that we know the structure, we can compare the models:

R code:
```
g$model$ltom$psill - m.ltom.f$psill
sum(g$model$ltom$psill) - sum(m.ltom.f$psill)
sum(g$model$ltom$psill)

g$model$ltpb$psill - m.ltpb.f$psill
sum(g$model$ltpb$psill) - sum(m.ltpb.f$psill)
sum(g$model$ltpb$psill)
```

R console output:
```
[1]  0.01063506 -0.00057129
[1] 0.010064
[1] 0.059334

[1]  0.0076558 -0.0044897
[1] 0.0031661
[1] 0.11591
```

The total sills, 0.059 for $\log_{10}$OM and 0.116 for $\log_{10}$Pb, were hardly affected (in both cases raised slightly). The partial sills were adjusted even less. Thus the common starting point for the linear model of co-regionalisation was satisfactory. Note that the total sill for Pb is about double that for OM.

7 Co-kriging with one co-variable

Now that we have modelled the co-regionalisation, we can use the extra samples of OM to improve (we hope!) the predictions of the target variable.

Task 22 : Predict $\log_{10}$Pb on the interpolation grid using the modelled co-regionalisation. Summarize the predictions and their errors. •

The wrapper method `krige` that was used for OK (§5) can only be used for univariate kriging; here we must use the `predict.gstat` method. This takes a `gstat` object as the first argument and the prediction points data frame as the second argument,.

R code:

```
        # interpolate
k.c <- predict.gstat(g, meuse.grid)
str(k.c)
        # summarize predictions and their errors
summary(k.c$ltpb.pred); summary(k.c$ltpb.var)
```

R console output:
```
Linear Model of Coregionalization found. Good.
[using ordinary cokriging]

Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
  ..@ data       :Formal class 'AttributeList' [package "sp"] with 1 slots
  .. .. ..@ att:List of 5
  .. .. .. ..$ ltpb.pred    : num [1:3103] 2.34 2.36 2.34 2.31 2.39 ...
  .. .. .. ..$ ltpb.var     : num [1:3103] 0.0756 0.0663 0.0692 0.0723 0.0562 ...
  .. .. .. ..$ ltom.pred    : num [1:3103] 1.03 1.05 1.04 1.02 1.07 ...
  .. .. .. ..$ ltom.var     : num [1:3103] 0.0427 0.0387 0.0398 0.0411 0.0346 ...
  .. .. .. ..$ cov.ltpb.ltom: num [1:3103] 0.0542 0.0488 0.0503 0.0521 0.0430 ...
  ..@ coords.nrs : int [1:2] 1 2
  ..@ coords     : num [1:3103, 1:2] 181180 181140 181180 181220 181100 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:2] "x" "y"
  ..@ bbox       : num [1:2, 1:2] 178460 329620 181540 333740
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "x" "y"
  .. .. ..$ : chr [1:2] "min" "max"
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
  .. .. ..@ projargs: chr NA

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.59    1.82    2.01    2.04    2.26    2.67
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0415  0.0512  0.0559  0.0591  0.0640  0.1030
```

Both variables and their covariance are all predicted at the same time. We are interested in the target variable, but the linear model also predicts the co-variable, in both cases using all sample information.

Task 23 : Display the predictions and their errors as maps. •

R code:
```
plot.kresults(k.c, "ltpb", meuse, meuse.pb, "lead", "CK with OM covariable, ")
```

Figure 13 shows the results. There are still some obvious discrepancies between the predictions and extra Pb points, although not so glaring as with OK (Fig. 5 left).

7.1 Validation

We have a set of 103 points at which lead was measured, but which we didn't use in the subset, either for modelling or interpolation. We did use these points for CK prediction, but only their OM values. Thus, as in

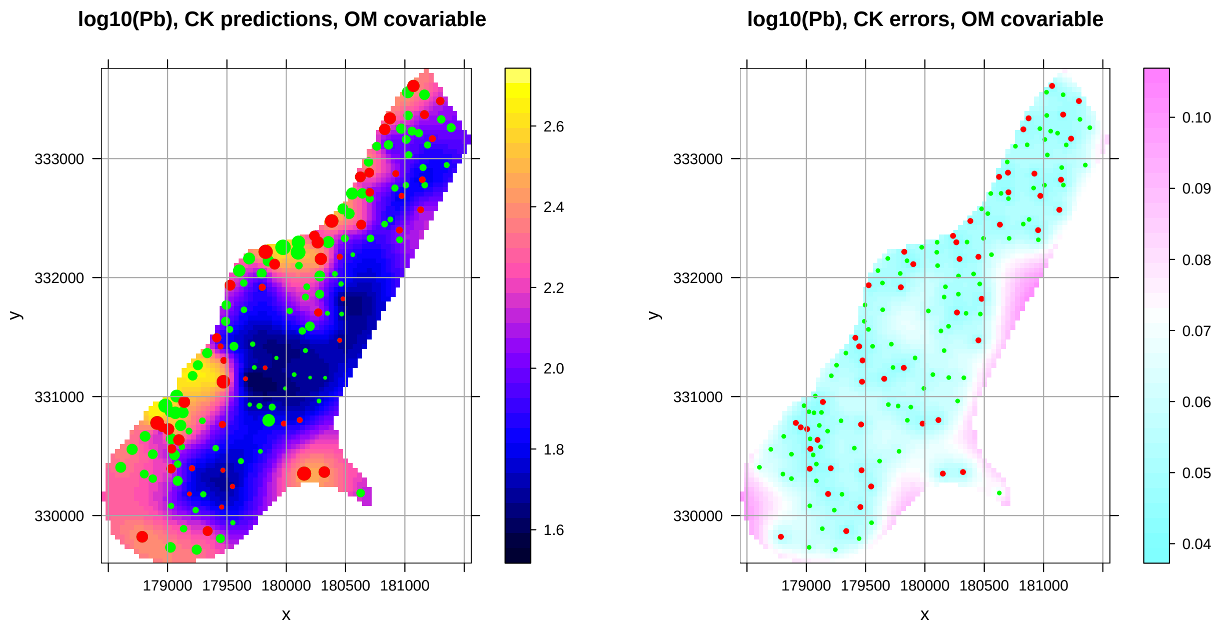

Figure 13: Co-kriging predictions and error, log10(Pb); subsample points red, extra points green

§5.1, the lead values at these points can be used as a *validation dataset* to assess the performance of the interpolation. We already have the data frame of extra points `meuse.extra` from §5.1.

Task 24 : Interpolate at the extra points, using CK from the sample points, and determine the actual prediction bias and precision. •

R code:

```
        # predict at the extra points
k <- predict.gstat(g, meuse.extra)
        # compute and summarize prediction errors
diff <- k$ltpb.pred - meuse.extra$ltpb
summary(diff)
sqrt(sum(diff^2)/length(diff))   # RMS error (precision)
sum(diff)/length(diff)               # mean error (bias)
```

R console output:

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.8110 -0.0559  0.0379  0.0148  0.1060  0.5550

[1] 0.22263
[1] 0.014763
```

There is a slight positive bias: the median and mean predictions at the extra points by CK is higher than the true value, as is the mean error (+0.015). The overall precision is good (RMSE = 0.222). There are several large relative errors (from −0.81 to +0.55) compared to the median value for the extra points of 2.07. This validation will be compared with the OK validation (§5.1) in §8.3, below.

Task 25 : Display the validation errors as a bubble plot. •

R code:

```
plot.valids(k, "ltpb", meuse.extra, "ltpb", cv.c, "CK, OM co-variable")
```

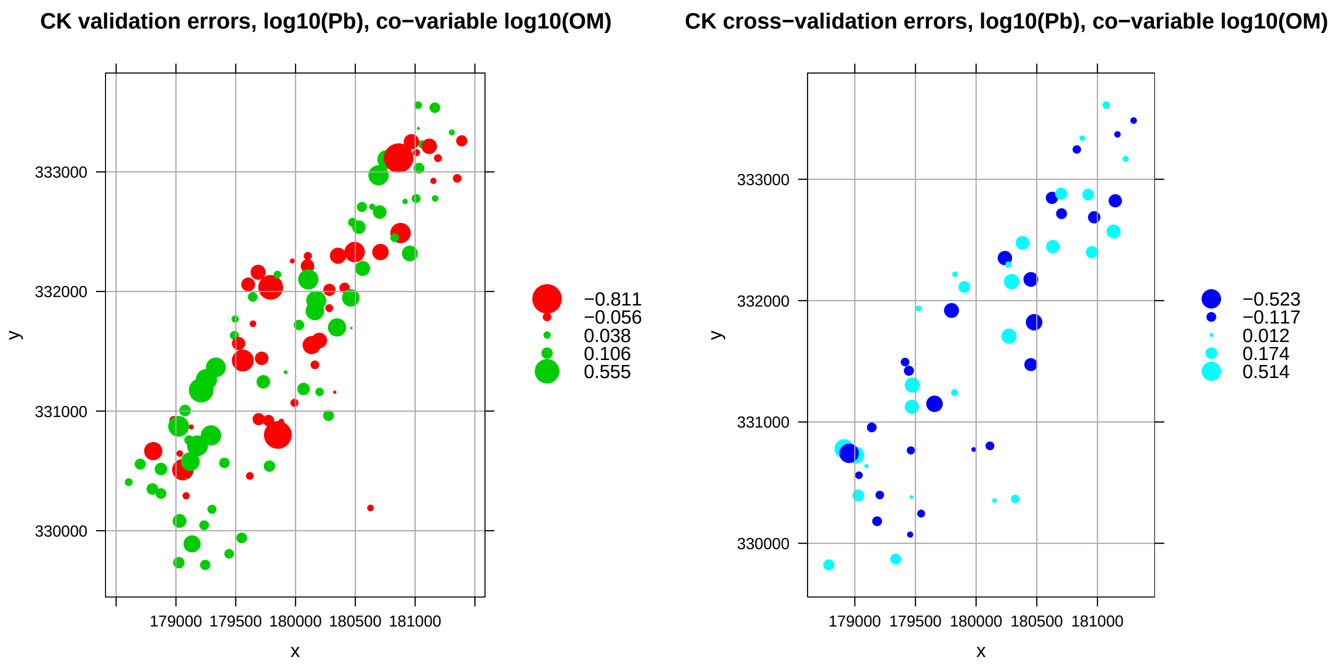

Figure 14: CK validation (left) and cross-validation (right) errors of log10(Pb)

Figure 14 (left) shows the validation errors at the extra points; this is comparable to Fig. 7 (left) for the OK predictions.

7.2 Cross-validation

We can also evaluate the success of CK from its cross-validation, as we did for OK (§5.2). As with OK, the cross-validatation is at the 52 points with data for Pb in the subset, since these were used for the interpolation. Each point is held out in turn, and the prediction at that point is made from the remaining 51 points. However, for CK cross-validation, all 153 samples of the co-variable are used to predict the target variable; these points are not held out. For almost all points in the subset there is a measurement of the co-variable (OM) at that point; this should give a decided advantage to CK if the feature-space correlation is good and there is a good spatial cross-correlation.

For the univariate case (OK) we were able to use the `krige.cv` cross-validation method, but for the more general case of CK we must use the `gstat.cv` cross-validation method that can be applied to arbitrary `gstat` objects. We do not have to specify the model, since this is already stored as part of the `gstat` object.

Task 26 : Cross-validate the CK predictions of $\log_{10}$Pb and compute the diagnostic measures. •

R code:

```
cv.c <- gstat.cv(g)
str(cv.c)
summary(cv.c$residual)
sqrt(mean(cv.c$residual^2))
mean(cv.c$residual)
mean(cv.c$residual^2/cv.c$ltpb.var)
```

R console output:

```
'data.frame': 52 obs. of  8 variables:
 $ x        : num  181072 181298 181165 181232 180874 ...
 $ y        : num  333611 333484 333370 333168 333339 ...
 $ ltpb.pred: num  2.41 2.11 2.16 1.86 2.42 ...
 $ ltpb.var : num  0.00827 0.00663 0.00463 0.00695 0.00402 ...
 $ observed : num  2.48 2.06 2.12 1.90 2.45 ...
 $ residual : num   0.0647 -0.0458 -0.0438  0.0390  0.0315 ...
 $ zscore   : num   0.712 -0.563 -0.644  0.467  0.496 ...
 $ fold     : int  1 2 3 4 5 6 7 8 9 10 ...

    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.52300 -0.11700  0.01230  0.00591  0.17400  0.51400

[1] 0.20931
[1] 0.0059087
[1] 12.619
```

The RMSE is 0.209, the mean error is almost zero, but the Mean Squared Deviation Ratio (MSDR) of the prediction residuals vs. the kriging error is very high (12.6) meaning that the actual residuals are an order of magnitude greater than what CK predicts; this is very poor performance, and shows that the CK prediction errors are too low.

Task 27 : Display a postplot of the errors. •

R code:

```
plot.valids(k, "ltpb", meuse.extra, "ltpb", cv.c, "CK, OM co-variable")
```

Figure 14 (right) shows the cross-validation errors at the sample points; this is comparable to Fig. 7 (right) for the OK predictions. There are some very large errors, from −0.52 to +0.51; this is similar to the OK result.

8 Comparison of co-kriging and ordinary kriging

We have now used both OK and CK to predict the lead distribution across the study area. In this section we compare the predictions and their errors, and investigate the reasons for any discrepancies.

8.1 Predictions

First we compare the predictions of the target variable from OK and CK.

Task 28 : Determine the differences between co-kriging (with OM) and

OK predictions of $\log_{10}$Pb, and display these as maps. •

These can be compared graphically on the same scale by use of the `range` method on the list of kriging predictions to find the extremes and then by dividing this range into a number (here, 32) of break points. The differences between predictions are computed, placed into a temporary data frame, and displayed with `levelplot`, using a different colour scheme than for the predictions. This has been encapsulated in the `plot.cf` function which is listed in §A.1 and also provided as source code in file `ck_plotfns.R`.

R code:

```
plot.cf(k.o, k.c, "var1", "ltpb", meuse, meuse.pb, "pred",
            "log10(Pb), OK", "log10(Pb), CK with log10(OM)")
```

There are some regions of the map where the predictions are quite different; these are in the undersampled regions where the OM sample provides new information. Now we compare them numerically:

R code:

```
summary(k.o$var1.pred); summary(k.c$ltpb.pred)
diff <- k.c$ltpb.pred - k.o$var1.pred
summary(diff); rm(diff)
```

R console output:

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.71    1.90    2.03    2.06    2.23    2.53
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.59    1.82    2.01    2.04    2.26    2.67
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.3040 -0.0885 -0.0332 -0.0188  0.0440  0.3660
```

The CK predictions are on average slightly lower than the OK predictions; there are some fairly large differences, both positive and negative.

8.2 Prediction errors

Second, we compare the prediction errors for the target variable from OK and CK.

A major aim of using co-kriging is to reduce the prediction variances. With so much new information (103 extra points, plus the 52 colocated points) and a fairly good model of co-regionalisation, we expect lower predication errors.

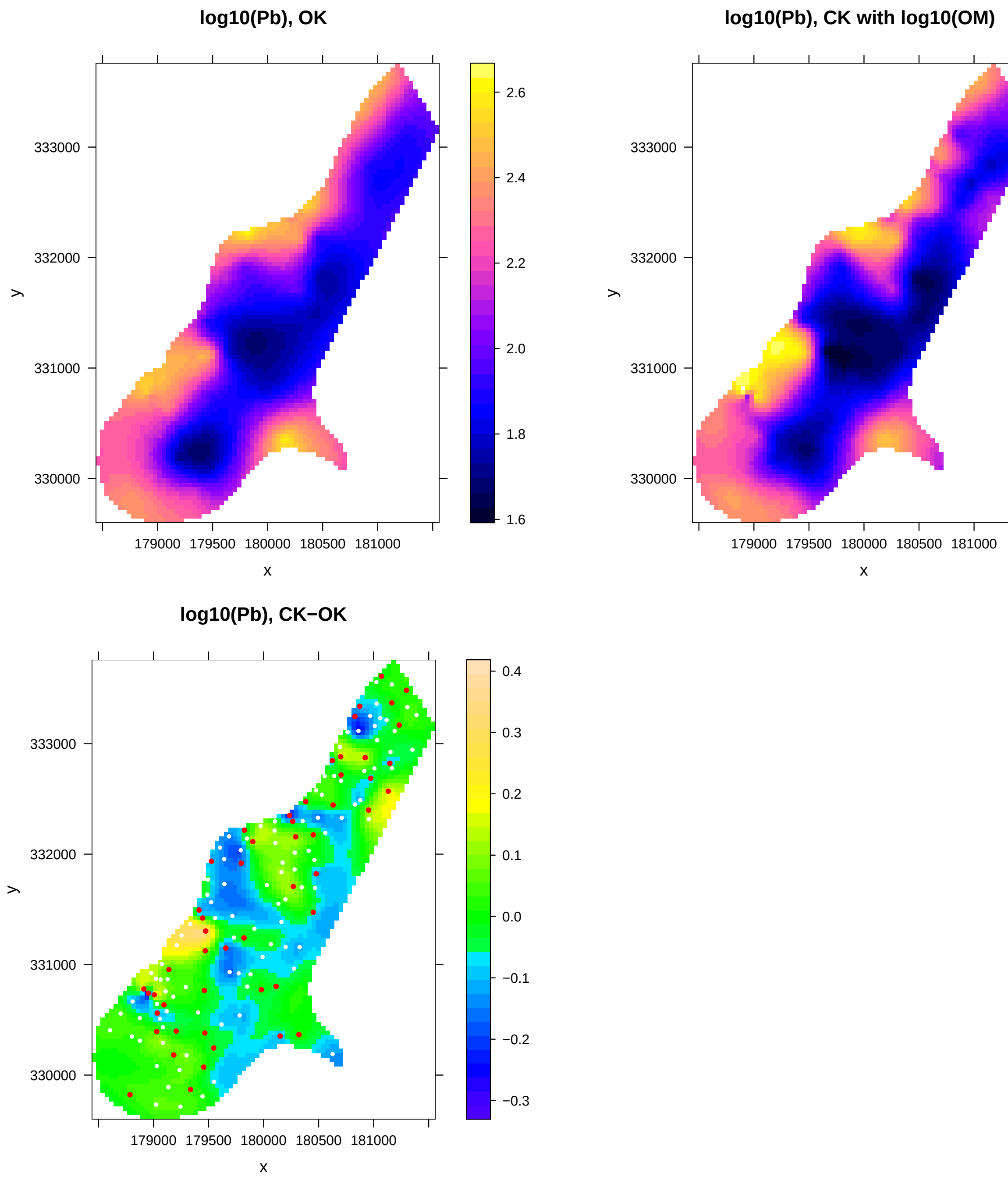

Figure 15: Cokriging with OM vs. ordinary kriging, log(Pb); subset points in red, extra points in white; yellow: CK prediction higher, blue: OK prediction higher

Task 29 : Determine the differences between the prediction errors of CK with OM and OK and display these as maps. •

We compare these graphically on the same scale; in addition we show the differences with a different colour scheme.

R code:
```
plot.cf(k.o, k.c, "var1", "ltpb", meuse, meuse.pb, "var",
            "log10(Pb), OK", "log10(Pb), CK with log10(OM)")
```

Clearly, the cokriging errors are in general larger, especially near the sample points of the subset. This is because the co-kriging predictions are based not only on the target variable but also the co-variable at these points as well as their covariance. Both of these have a significant nugget effect.

However, some areas of the map have much improved predictions; these are the areas far from the subsample points and relatively nearer the extra samples with the co-variable.

Now we compare the prediction errors numerically:

R code:
```
summary(k.o$var1.var); summary(k.c$ltpb.var)
diff.var <- (k.c$ltpb.var - k.o$var1.var)
summary(diff.var)
round(100*sum(diff.var < 0)/length(diff.var))
rm(diff.var)
```

R console output:
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0357  0.0519  0.0608  0.0638  0.0738  0.1120
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0415  0.0512  0.0559  0.0591  0.0640  0.1030
    Min.   1st Qu.    Median      Mean   3rd Qu.     Max.
-0.03750 -0.00914 -0.00286 -0.00469  0.00125  0.00792

[1] 67
```

The OK prediction variances are higher than the CK prediction variances over 67% of the map.

8.3 Validations

We used the extra sample points to evaluate the performance of OK (§5.1) and CK (§7.1). Here we compare these two validations.

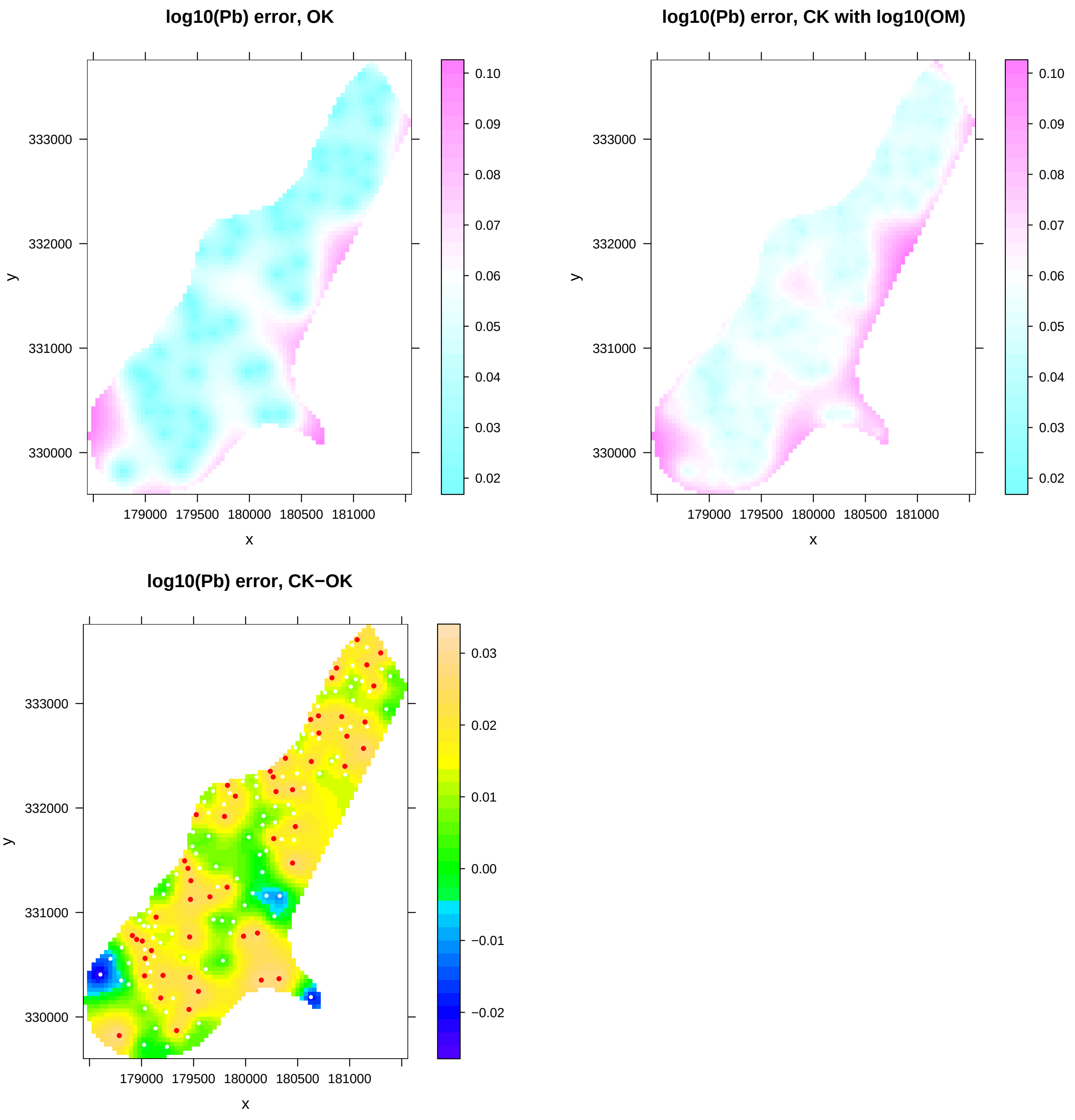

Figure 16: Cokriging with OM vs. OK prediction errors, log10(Pb); subset points shown in red, extra points in white; yellow: CK error higher, blue: OK error higher

	ME	RMSE	MinE	MaxE
OK	0.0250	0.1660	-0.663	+0.372
CK with OM	0.0148	0.2226	-0.811	+0.555

The OK validation is superior in all measures except bias: the mean error of CK is closer to zero than that of OK. However, the overall precision is greater for OK, and the extreme errors are smaller.

Fig. 17 shows where the prediction differences are located. There are many large changes in predictions between the two validations.

R code:
```
kv.ok <- krige(ltpb ~ 1, meuse.pb, meuse.extra, m.ltpb.f)
kv.ck <- predict.gstat(g, meuse.extra)
kv.diff <- (kv.ck$ltpb.pred - kv.ok$var1.pred)
summary(kv.diff)
d <- SpatialPointsDataFrame(coordinates(kv.ok), data=as.data.frame(kv.diff))
bubble(d, col = c("plum","seagreen"),
          panel = function(x, ...) {
            panel.xyplot(x, ...);
            panel.grid(h=-1, v=-1, col="darkgrey")})
rm(kv.ok, kv.ck, kv.diff, d)
```

R console output:
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.8380 -0.1360  0.0193 -0.0103  0.1150  0.4840
```

8.4 Cross-validations

We summarize the cross-validation performance of OK (§5.2) and CK (§7.2):

	ME	RMSE	MSDR
OK	0.0037	0.2369	00.88
CK with OM	0.0059	0.2093	12.62

The cross-validation of $\log_{10}$Pb at the 52 subsample points shows that CK had slightly lower RMSE; however the MSDR for CK was very high, meaning that the actual residuals are an order of magnitude greater than what CK predicts. By contrast, the MSDR for OK was close to the ideal 1. Both had low bias, OK being a bit lower.

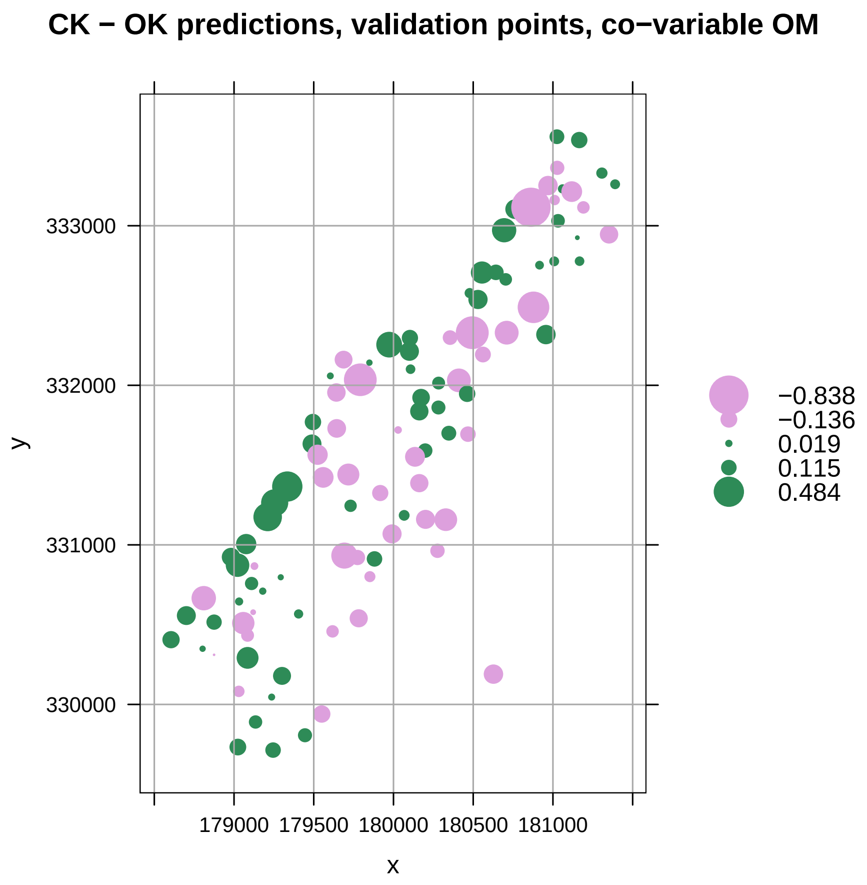

Figure 17: Difference between CK and OK predictions, log10(Pb) at validation points, co-variable OM

R code:

```
cv.diff <- data.frame(diff = cv.c$residual -
               cv.o$residual, better = (abs(cv.c$residual) < abs(cv.o$residual)))
summary(cv.diff$diff)
summary(cv.diff$better)
d <- SpatialPointsDataFrame(coordinates(cv.c), data=as.data.frame(cv.diff$diff))
bubble(d,
        col=c("lightblue", "red4"),
          panel = function(x, ...)
            panel.xyplot(x, ...);
            panel.grid(h=-1, v=-1, col="darkgrey"))
rm(cv.diff)
```

R console output:
```
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.43300 -0.12800  0.00523  0.00224  0.14300  0.48500
   Mode    FALSE     TRUE
logical       23       29
```

Slightly more than half (56%) of the individual cross-validation predictions are better with CK than OK. Fig. 18 shows the difference between the cross-validation residuals for the two methods. There are large differences between the residuals, showing the large effect of CK on predictions at the sample points.

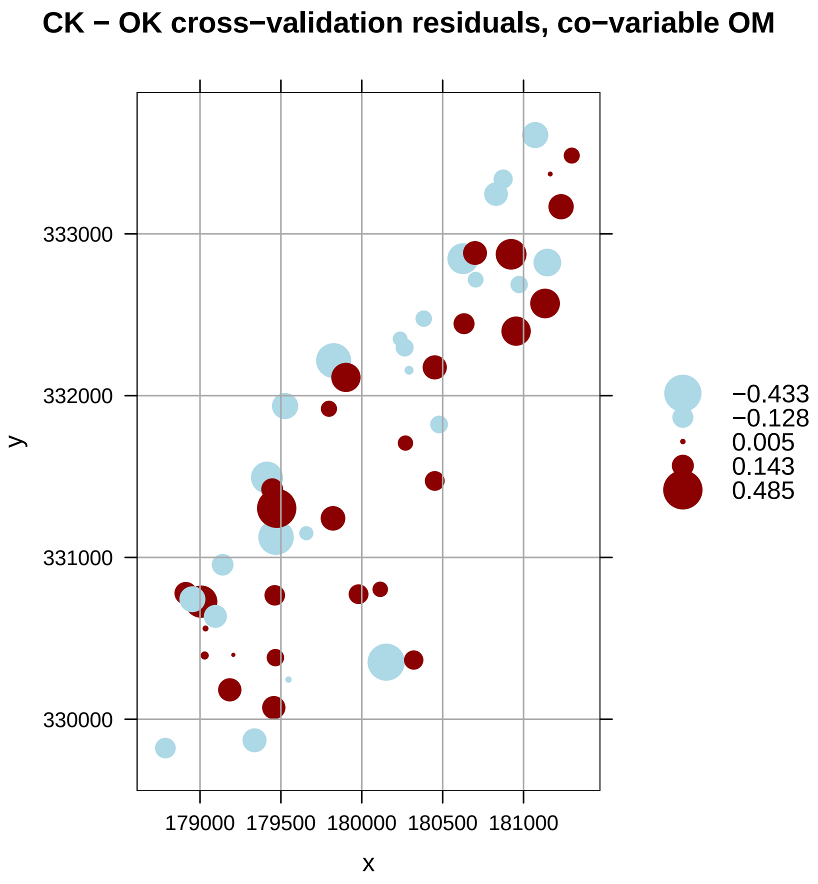

Figure 18: Difference between CK and OK cross-validation residuals, log10(Pb), co-variable OM

9 Co-kriging with a different co-variable

The use of organic matter as a co-variable gave disappointing results (§8.3, §8.4). Now we try the other co-variable (Zn), to see if it is a better co-variate. We follow the same procedure as explained in detail in §6 (modelling the co-regionalisation) and §7 (co-kriging with one co-variable), so here we present the code and results without explanation, concentrating on the discussion.

9.1 Modelling the co-variable

As with the other co-variable (§6.1), we first we examine the feature-space correlation.

Task 30 : Display a scatterplot of the co-variable (Zn) vs. the target variable (Pb) and compute the correlation coefficient. Describe the feature-space relation. •

Note that we should only use the points from the subsample, since we are simulating the situation where the target variable is under-sampled.

R code:
```
xyplot(ltpb ~ ltzn, pch=20, cex=1.2, col="red4",
      ylab="log10(Pb)", xlab="log10(Zn)")
cor(ltzn, ltpb)
```

R console output:
```
[1] 0.9738
```

The scatterplot (Fig. 19) and numerical correlation both show that there is a very strong positive feature-space relation between the target and co-variable. However, there is a big difference between the co-variables: in the case of OM (§6.1) the relation only explained about 38% of the total variability, whereas in this case the relation explains $0,9738^2 = 0.9483$ or about 95% of the total variability; this should give better results.

This close relation is evidence that the processes by which soils in this area are polluted with the two metals are similar.

Task 31 : Model the omnidirectional spatial variogram of the log10-transformed covariable. Compare the structure with that of the target variable. •

We use all the available samples for the co-variable, not just the subset.

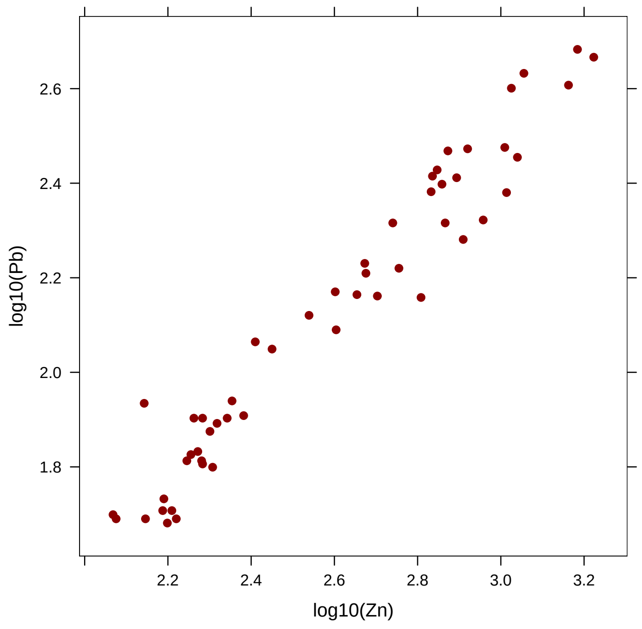

Figure 19: Feature-space relation between Pb and Zn, subsample

R code:

```
        # all valid covariable observations, with coordinates
meuse.co <- as.data.frame(meuse)[, c("x", "y", "zinc")]
        # add log10-transformed variables for convenience
meuse.co <- cbind(meuse.co, ltzn = log10(meuse.co$zinc))
coordinates(meuse.co) <- ~ x + y
str(meuse.co)
        # experimental variogram
v.ltzn <- variogram(ltzn ~ 1, meuse.co, cutoff=1800)
plot(v.ltzn, pl=T)
        # model by eye
m.ltzn <- vgm(.11, "Sph", 1000, .02)
        # fit
(m.ltzn.f <- fit.variogram(v.ltzn, m.ltzn))
plot(v.ltzn, pl=T, model=m.ltzn.f)
        # compare variogram structure to target variable
m.ltzn.f$range[2]; m.ltpb.f$range[2]
round(m.ltzn.f$psill[1]/sum(m.ltzn.f$psill),2)
round(m.ltpb.f$psill[1]/sum(m.ltpb.f$psill),2)
```

R console output:

```
Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
  ..@ data       :Formal class 'AttributeList' [package "sp"] with 1 slots
  .. .. ..@ att:List of 2
  .. .. .. ..$ zinc: num [1:155] 1022 1141  640  257  269 ...
  .. .. .. ..$ ltzn: num [1:155] 3.01 3.06 2.81 2.41 2.43 ...
  ..@ coords.nrs : int [1:2] 1 2
  ..@ coords     : num [1:155, 1:2] 181072 181025 181165 181298 181307 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:2] "x" "y"
  ..@ bbox       : num [1:2, 1:2] 178605 329714 181390 333611
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "x" "y"
  .. .. ..$ : chr [1:2] "min" "max"
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
  .. .. ..@ projargs: chr NA

  model    psill  range
1   Nug 0.015521   0.00
2   Sph 0.106151 976.62

[1] 976.62
[1] 876.18

[1] 0.13
[1] 0.21
```

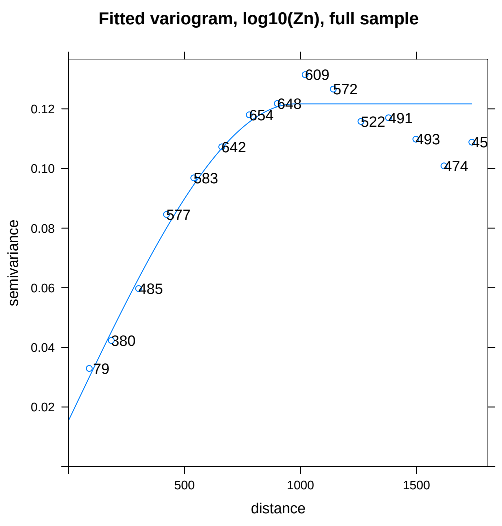

Figure 20: Experimental variograms with fitted model, log10(Zn), full data set

The structure for $\log_{10}$Zn (Fig. 20) is quite similar to that for $\log_{10}$Pb (Fig 4). A spherical model could be fitted in both cases, but with a much lower nugget effect (about 1/8 of the total sill) and a somewhat longer range (976 vs. 876 m) for $\log_{10}$Zn. Thus Zn is more structured than Pb.

9.2 Building a data structure to model co-regionalisation

As with the other co-variable (§6), the next step is to model its co-regionalisation with the target variable.

Task 32 : Build a `gstat` structure containing the two sample sets: subsample for Pb and full sample for Zn. •

R code:
```
(g2 <- gstat(NULL, id = "ltpb", form = ltpb ~ 1, data = meuse.pb))
(g2 <- gstat(g2, id = "ltzn", form = ltzn ~ 1, data = meuse.co))
```

R console output:
```
data:
ltpb : formula = ltpb‘~‘1 ; data dim = 52 x 6

data:
ltpb : formula = ltpb‘~‘1 ; data dim = 52 x 6
ltzn : formula = ltzn‘~‘1 ; data dim = 155 x 2
```

Task 33 : Compute and display the two direct variograms and one cross-variogram. •

R code:
```
v.cross <- variogram(g2)
plot(v.cross, pl=T)
```

9.3 Fitting a linear model of co-regionalisation

Task 34 : Add variogram models to the `gstat` object and fit a linear model of co-regionalisation to them. •

R code:
```
g2 <- gstat(g2, id = "ltpb", model = m.ltpb.f, fill.all=T)
(g2 <- fit.lmc(v.cross, g2))
plot(variogram(g2), model=g2$model)
```

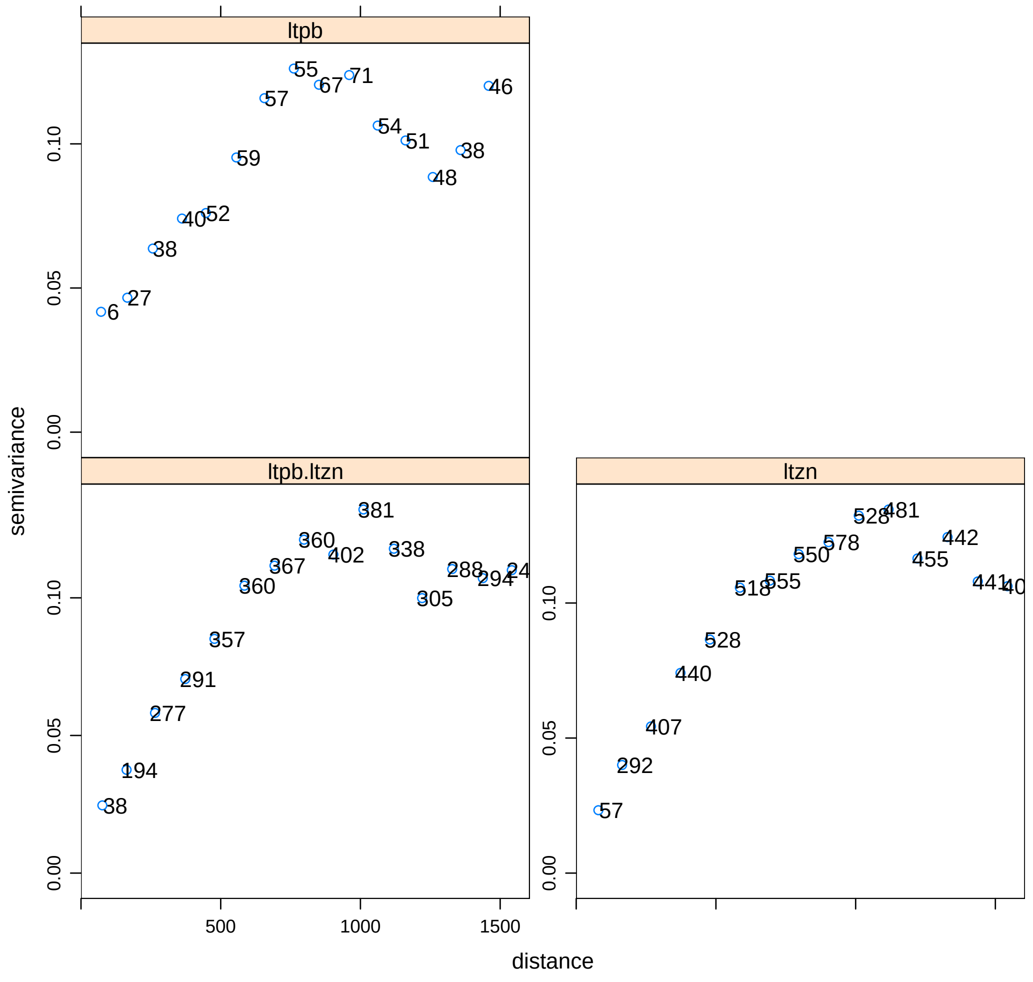

Figure 21: Direct and cross variograms, log10(Pb) and log10(Zn)

R console output:

```
data:
ltpb : formula = ltpb‘~‘1 ; data dim = 52 x 6
ltzn : formula = ltzn‘~‘1 ; data dim = 155 x 2
variograms:
             model     psill  range
ltpb[1]        Nug 0.0274357   0.00
ltpb[2]        Sph 0.0897199 876.18
ltzn[1]        Nug 0.0090083   0.00
ltzn[2]        Sph 0.1150807 876.18
ltpb.ltzn[1]   Nug 0.0104469   0.00
ltpb.ltzn[2]   Sph 0.1016122 876.18
```

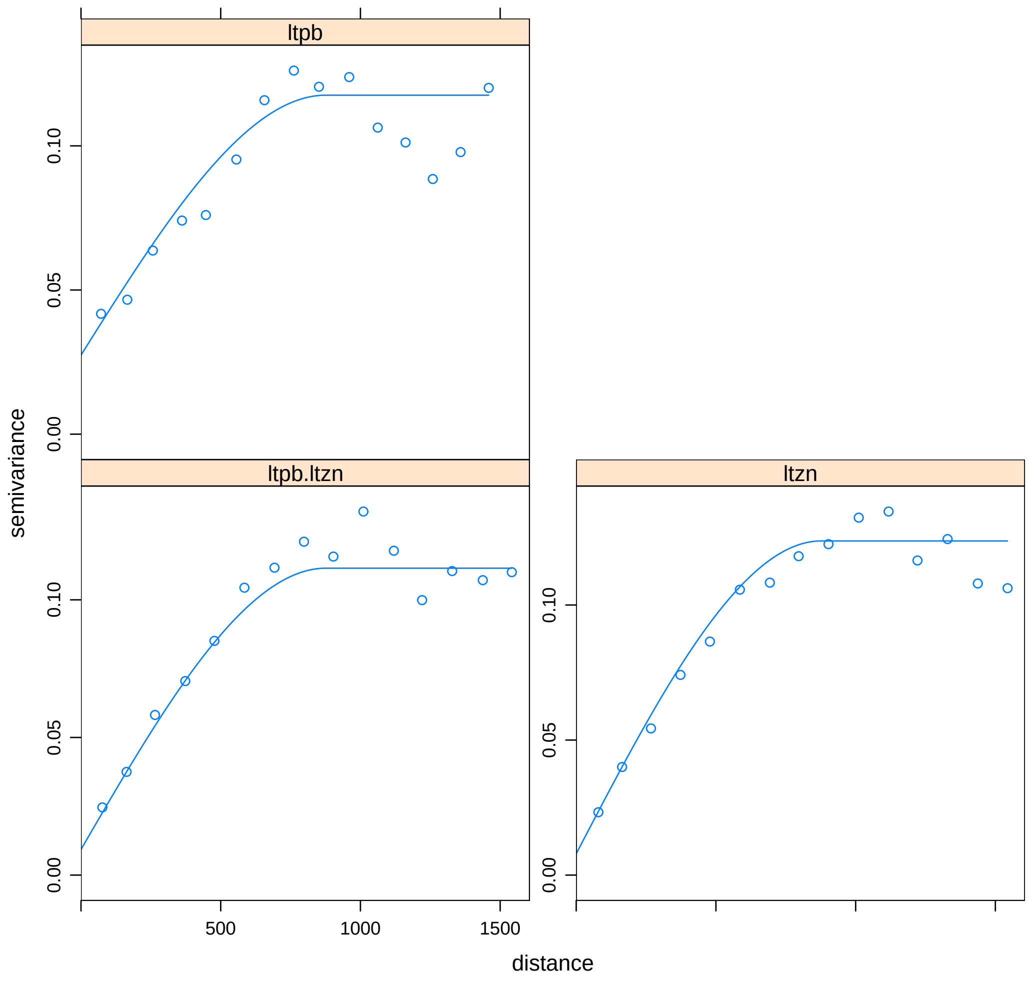

Figure 22: Fitted direct and cross variograms, log10(Pb) and log10(Zn)

The model fits fairly well (Fig. 22), although the sill for the cross-variogram seems a bit low. The enforced common range (876 m) does not seem unreasonable.

Task 35 : Compare these fits to the fitted models for the direct variograms. •

R code:
```
        # zinc
g2$model$ltzn$psill - m.ltzn.f$psill
sum(g2$model$ltzn$psill) - sum(m.ltzn.f$psill)
sum(g2$model$ltzn$psill)
        # lead
g2$model$ltpb$psill - m.ltpb.f$psill
sum(g2$model$ltpb$psill) - sum(m.ltpb.f$psill)
sum(g2$model$ltpb$psill)
```

R console output:
```
[1] -0.0065131  0.0089298
[1] 0.0024168
[1] 0.12409

[1] 0.00348539 0.00092185
[1] 0.0044072
[1] 0.11716
```

The total sills, 0.124 for $\log_{10}$Zn and 0.117 for $\log_{10}$Pb, were hardly affected (in both cases raised slightly). The partial sills were adjusted even less. Thus the common starting point for the linear model of co-regionalisation was satisfactory. The partial and total sills are quite similar between the two models.

Task 36 : Compare the fitted variogram for Pb in the two co-regionalisations; this one with Zn and the previous with OM (§6). •

R code:
```
g2$model$ltpb$psill - g$model$ltpb.$psill
sum(g2$model$ltpb$psill) - sum(g$model$ltpb$psill)
```

R console output:
```
[1] 0.0014852 0.0382759
[1] 0.0012411
```

The models are somewhat different: the model co-fitted with Zn has slightly higher partial sill of the spherical model, nugget, and therefore total sill. This shows that the linear model of co-regionalisation depends on the variables included in the model.

9.4 Co-kriging with one co-variable

Task 37 : Predict $\log_{10}$Pb on the interpolation grid using the modelled co-regionalisation with $\log_{10}$Zn. Summarize the predictions and their errors. •

R code:
```
k.c2 <- predict.gstat(g2, meuse.grid)
summary(k.c2$ltpb.pred); summary(k.c2$ltpb.var)
```

R console output:
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.66    1.84    1.97    2.02    2.20    2.69
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0329  0.0409  0.0448  0.0484  0.0525  0.0972
```

The prediction errors are somewhat lower than for the CK with OM (§7).

Task 38 : Display the predictions and their errors as maps. •

R code:
```
plot.kresults(k.c2, "ltpb", meuse, meuse.pb, "lead", "CK with Zn covariable, ")
```

Figure 23 shows the results. There are no glaring discrepancies between the predictions and extra Pb points, unlike the CK predictions using OM as co-variable (Fig. 13 left) and OK (Fig. 5 left). In particular, the extra point near $(179750E, 330800N)$ is well-predicted; this is because of the high Zn value at that point in the full data set.

9.5 Validation

Task 39 : Interpolate at the extra (validation) points, using CK from the sample points, and determine the actual prediction bias and precision. •

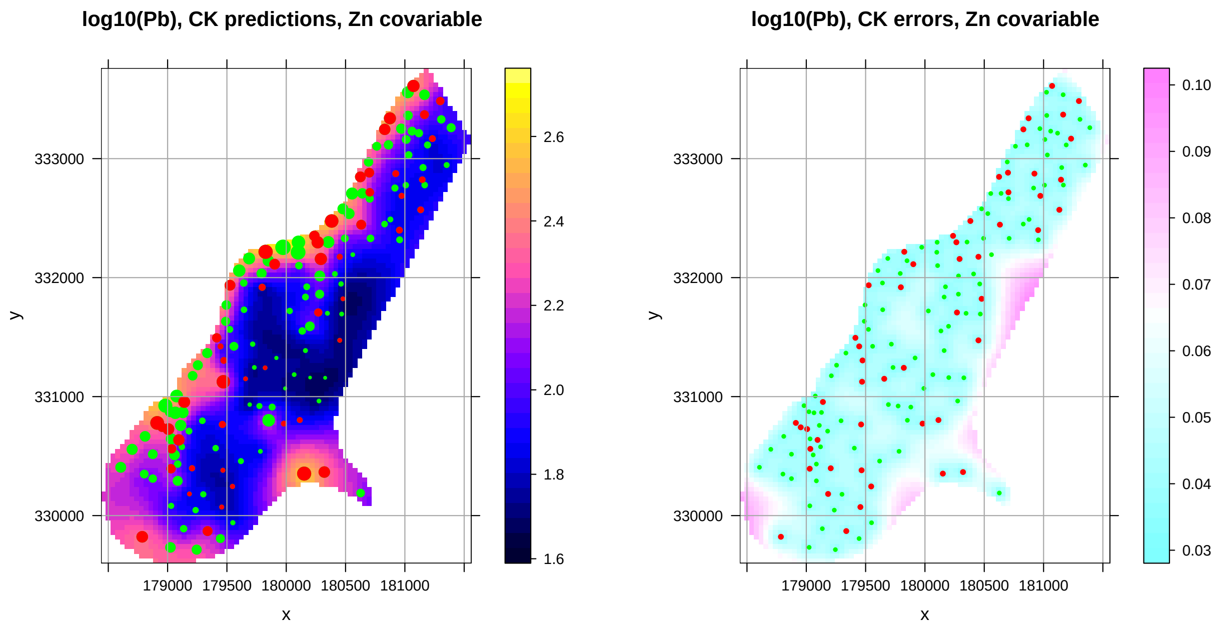

Figure 23: Co-kriging predictions and error, log10(Pb), co-variable Zn; subsample points red, extra points green

R code:

```
        # predict at the extra points
k <- predict.gstat(g2, meuse.extra)
        # compute and summarize prediction errors
diff <- k$ltpb.pred - meuse.extra$ltpb
summary(diff)
sqrt(sum(diff^2)/length(diff))   # RMS error (precision)
sum(diff)/length(diff)               # mean error (bias)
```

R console output:

```
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.13500 -0.05210 -0.00575  0.00539  0.05070  0.19200
[1] 0.075133
[1] 0.0053896
```

There is a very slight positive bias: the mean predictions at the extra points by CK is higher than the true value, as is the mean error (+0.005); however the median is slightly lower (−0.006). The overall precision is very good (RMSE = 0.075). There are no large relative errors (only from −0.14 to +0.19) compared to the median value for the extra points of

2.07.

Task 40 : Display the validation errors as a bubble plot. •

R code:
```
plot.valids(k, "ltpb", meuse.extra, "ltpb", cv.o, "CK, Zn co-variable")
```

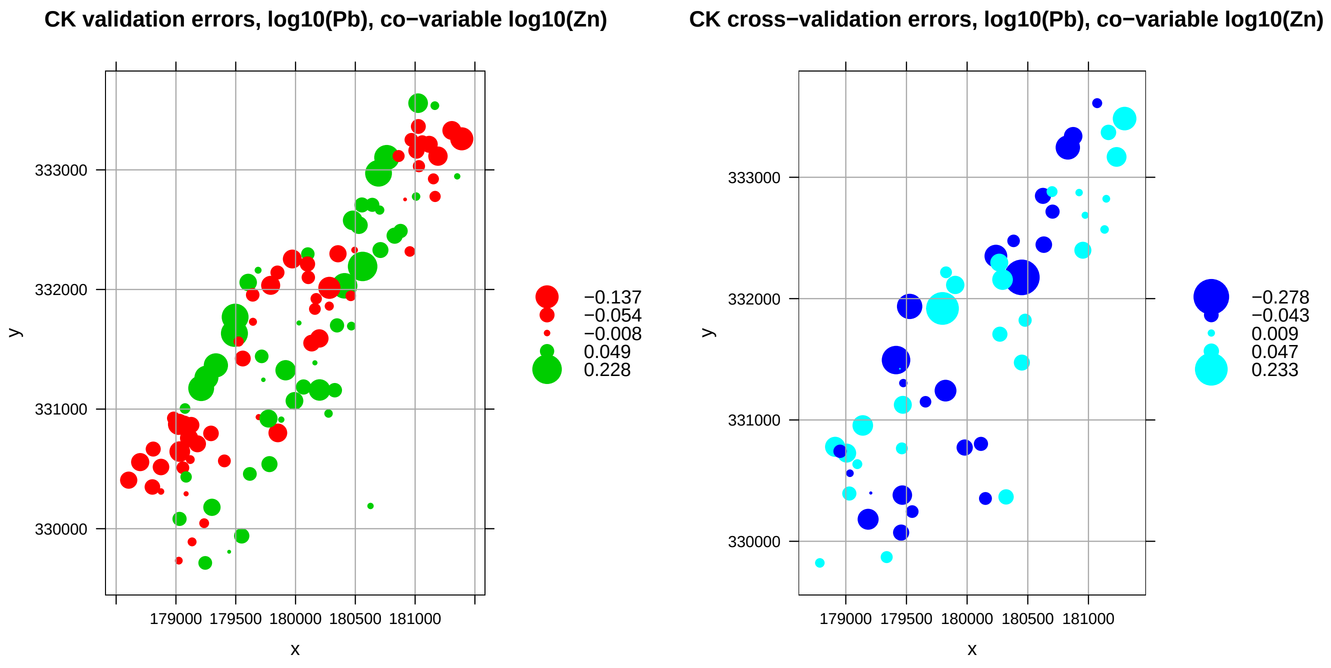

Figure 24: CK validation (left) and cross-validation (right) of log10(Pb), covariable Zn

Figure 24 (left) shows the prediction errors at the validation points; this is much better than the validation of CK with OM as covariable (Fig. 14 (left)) (note the different scales).

9.6 Cross-validation

Task 41 : Cross-validate the CK predictions of $\log_{10}$Pb and compute the diagnostic measures. •

R code:
```
cv.c2 <- gstat.cv(g2)
summary(cv.c2$residual)
sqrt(mean(cv.c2$residual^2))
mean(cv.c2$residual)
mean(cv.c2$residual^2/cv.c2$ltpb.var)
```

R console output:
```
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-1.77e-01 -4.11e-02  7.88e-03 -2.87e-06  4.49e-02  2.32e-01
[1] 0.070995
[1] -2.8697e-06
[1] 0.31228
```

The RMSE is 0.07, which is much lower than when using OM as the co-variable (0.21, §7.2). The mean error is again almost zero, but the Mean Squared Deviation Ratio (MSDR) lower than the ideal 1 (0.31), meaning that the CK prediction errors are less variable than reality, as expected with a smooth predictor.

Task 42 : Display a postplot of the errors. •

R code:
```
plot.valids(k, "ltpb", meuse.extra, "ltpb", cv.o, "CK, Zn co-variable")
```

Figure 24 (right) shows the cross-validation errors at the sample points; this is comparable to Fig. 7 (right) for the OK predictions and Fig. 14 (right) for the CK predictions with OM as the co-variable. The maximum errors, from −0.18 to +0.23 are about half those for OK or for CK with co-variable OM.

9.7 Comparison of co-kriging and ordinary kriging

In this section we compare the CK and OK predictions and their errors, and investigate the reasons for any discrepancies for CK with co-variable Zn, just as we did for CK with co-variable OM (§8).

9.7.1 Predictions

Task 43 : Determine the differences between co-kriging (with Zn) and OK predictions of $\log_{10}$Pb, and display these as maps. •

R code:
```
plot.cf(k.o, k.c2, "var1", "ltpb", meuse, meuse.pb, "pred",
            "log10(Pb), OK", "log10(Pb), CK with log10(Zn)")
```

There are some regions of the map where the predictions are quite different; these are in the undersampled regions where the Zn measurement provides new information. The CK map shows finer detail. This is similar to CK using OM as the co-variable (§8.1). Now we compare them numerically:

R code:
```
summary(k.o$var1.pred); summary(k.c2$ltpb.pred)
diff <- k.c2$ltpb.pred - k.o$var1.pred
summary(diff); rm(diff)
```

R console output:
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.71    1.90    2.03    2.06    2.23    2.53
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.66    1.84    1.97    2.02    2.20    2.69
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.3370 -0.0897 -0.0368 -0.0377  0.0121  0.3510
```

The CK predictions are on average slightly lower than the OK predictions; there some substantial differnces, both positive and negative, although half the differences are small, between -0.089 and $+0.012$; this is a narrower range than for CK with OM as the co-variable (§8.1).

9.7.2 Prediction variances

Task 44 : Determine the differences between the prediction errors of CK with Zn and OK and display these as maps. •

R code:
```
plot.cf(k.o, k.c2, "var1", "ltpb", meuse, meuse.pb, "var",
            "log10(Pb), OK", "log10(Pb), CK with log10(Zn)")
```

The cokriging errors are in general larger, especially near the sample points of the subset. However, some areas of the map have much improved predictions; these are the areas far from the subsample points

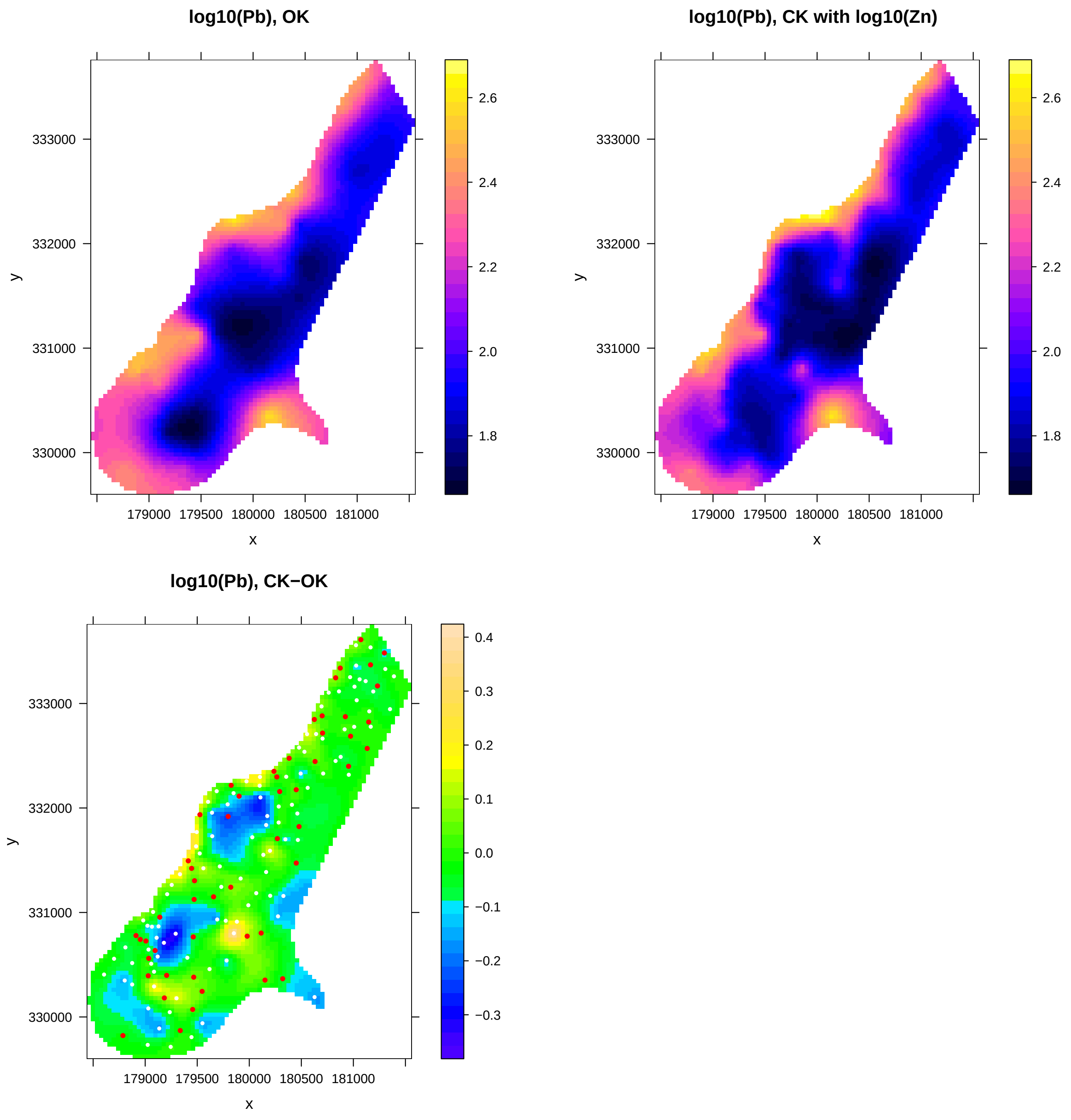

Figure 25: Cokriging with Zn vs. OK, log10(Pb); subset points in red, extra points in white; yellow: CK prediction higher, blue: OK prediction higher

and relatively nearer the extra samples with the co-variable. This is the same pattern we saw when using CK with OM as the co-variable (§8.2), but here the differences are less marked.

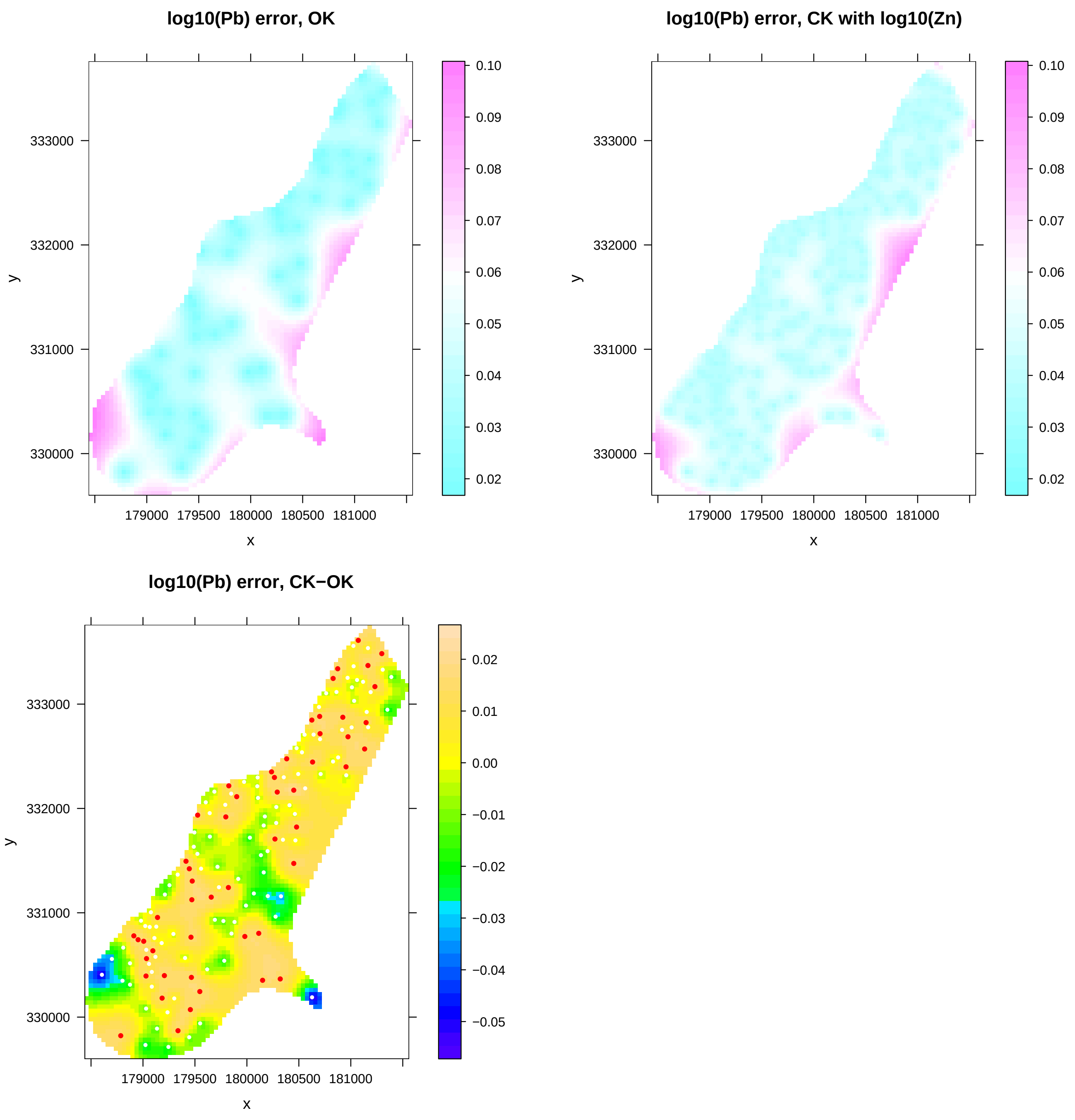

Figure 26: Cokriging with Zn vs. OK prediction errors, log10(Pb); subset in red, extra points in white; yellow: CK error higher, blue: OK error higher

Now we compare the prediction errors numerically:

R code:
```
summary(k.o$var1.var); summary(k.c2$ltpb.var)
diff.var <- (k.c2$ltpb.var - k.o$var1.var)
summary(diff.var)
```

R console output:
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0357  0.0519  0.0608  0.0638  0.0738  0.1120
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0329  0.0409  0.0448  0.0484  0.0525  0.0972
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-0.065500 -0.021400 -0.012500 -0.015400 -0.007000 -0.000809
```

The CK prediction variances are all lower than the OK prediction variances. This is a much better result than when using OM as the covariable (§8.2).

9.7.3 Validations

We used the extra sample points to evaluate the performance of OK (§5.1) and CK with Zn as co-variable (§9.5). Here we compare these two validations; this is the same as we did in §8.3 for CK with OM as co-variable.

	ME	RMSE	MinE	MaxE
OK	0.0250	0.1660	-0.663	+0.372
CK with Zn	0.0043	0.0775	-0.137	+0.228

The CK validation is well superior in all measures; this is in contrast to the result for CK with OM as the co-variable. Fig. 27 shows where the prediction differences are located. There are many large changes in predictions between the two validations, but not as extreme as when OM was used as the co-variable (Fig. 17).

R code:

```
kv.ok <- krige(ltpb ~ 1, meuse.pb, meuse.extra, m.ltpb.f)
kv.ck <- predict.gstat(g2, meuse.extra)
kv.diff <- data.frame(x = coordinates(kv.ck)[,"x"],
              y = coordinates(kv.ck)[,"y"],
              diff = kv.ck$ltpb.pred - kv.ok$var1.pred)
summary(kv.diff$diff)
coordinates(kv.diff) <- ~ x +y
bubble(kv.diff, z="diff",
        main="CK - OK predictions, validation points, co-variable Zn",
        col = c("plum","seagreen"),
          panel = function(x, ...) {
            panel.xyplot(x, ...);
            panel.grid(h=-1, v=-1, col="darkgrey")})
rm(kv.ok, kv.ck, kv.diff)
```

R console output:

```
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.4430 -0.1040 -0.0457 -0.0207  0.0738  0.5750
```

9.7.4 Cross-validations

We summarize the cross-validation performance of OK (§5.2) and CK with Zn as co-variable (§9.6); this is the same as we did in §8.4 for CK with OM as co-variable.

	ME	RMSE	MSDR
OK	+0.0037	0.2369	0.88
CK with Zn	-0.0035	0.0807	0.32

CK had much lower overall error than OK. The bias of both cross-validations was very low. The MSDR for CK was quite a bit further from the ideal 1 than OK, evidence of over-smooth predictions.

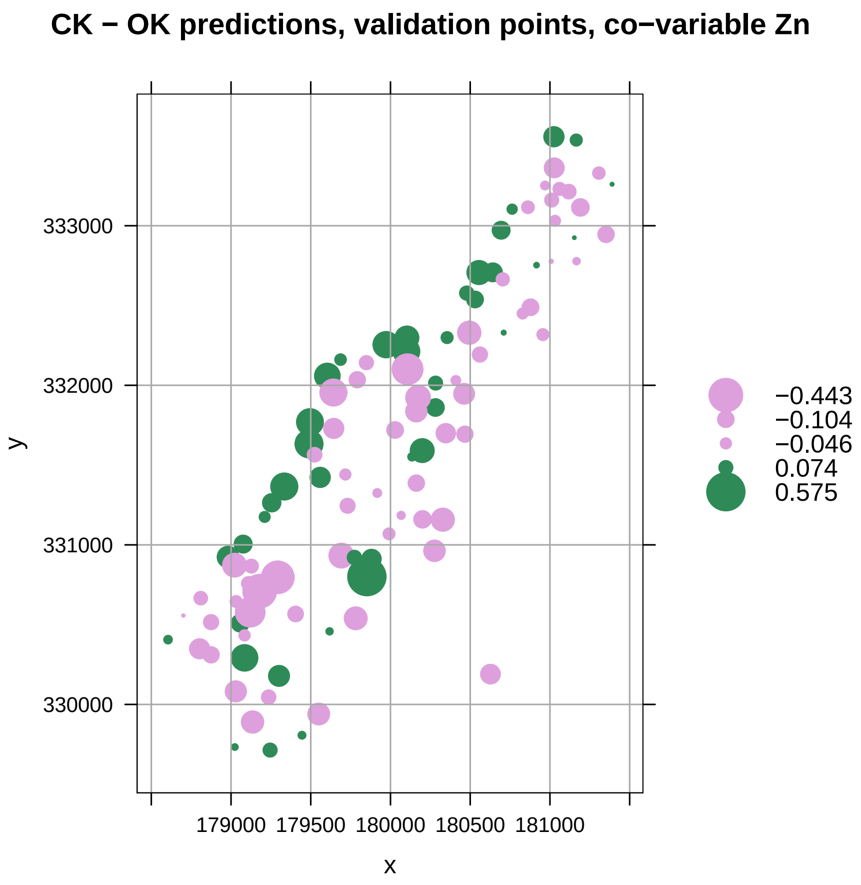

Figure 27: Difference between CK and OK predictions, log10(Pb) at validation points, co-variable Zn

---

R code:

```
cv.diff <- data.frame(x = coordinates(cv.c2)[,"x"],
              y = coordinates(cv.c2)[,"y"],
              diff = cv.c2$residual -
               cv.o$residual, better = (abs(cv.c2$residual) < abs(cv.o$residual)));
str(cv.diff);
summary(cv.diff$better);
coordinates(cv.diff) <- ~ x +y
bubble(cv.diff, z="diff",
        main="CK - OK cross-validation residuals, co-variable Zn",
        col = c("lightblue", "red4"),
          panel = function(x, ...)
            panel.xyplot(x, ...);
            panel.grid(h=-1, v=-1, col="darkgrey"))
rm(cv.diff)
```

R console output:

```
'data.frame': 52 obs. of  4 variables:
 $ x     : num  181072 181298 181165 181232 180874 ...
 $ y     : num  333611 333484 333370 333168 333339 ...
 $ diff  : num  -0.2930  0.2340  0.0935  0.2327 -0.2094 ...
 $ better: logi   TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE ...

   Mode   FALSE    TRUE
logical       6      46
```

About 85% of the individual cross-validation predictions are better with CK than OK. Fig. 28 shows the difference between the cross-validation residuals for the two methods. There are large differences between the residuals; this shows that CK with co-variable Zn had a large effect on predictions at the sample points. This effect is stronger than that for CK with co-variable OM (Fig. 18).

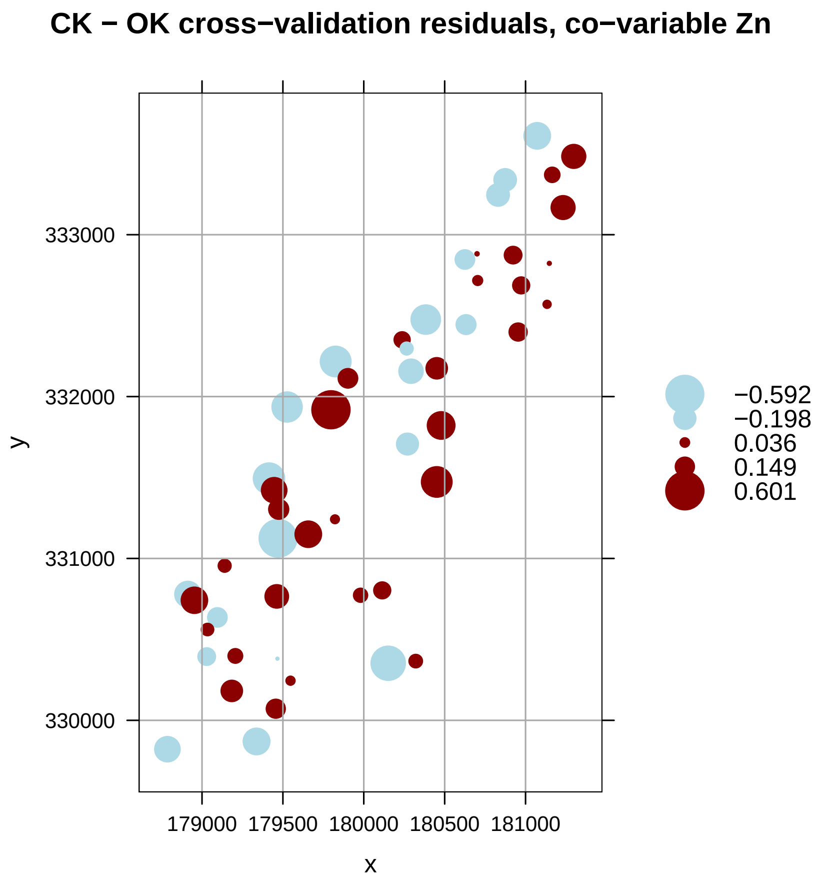

Figure 28: Difference between CK and OK cross-validation residuals, log10(Pb), co-variable Zn

10 Comparison of the three interpolations

Here we compare all three interpolations side-by-side.

10.1 Predictions

Task 45 : Compare the predictions of $\log_{10}$Pb for OK (§5), CK with OM as co-variable (§7), and CK with Zn as co-variable (§9.4). •

R code:

```
        # common colour scale
range <- range(k.o$var1.pred, k.c$ltpb.pred, k.c2$ltpb.pred)
breaks <- seq(range[1], range[2], length=32)
levelplot(var1.pred ~ x+y, as.data.frame(k.o), aspect="iso",
    col.regions=bpy.colors(64), at=breaks,
    main="log10(Pb), OK predictions",
         panel = function(x, ...) {
            panel.levelplot(x, ...);
            panel.grid(h=-1, v=-1, col="darkgrey")
})
levelplot(ltpb.pred ~ x+y, as.data.frame(k.c), aspect="iso",
    col.regions=bpy.colors(64), at=breaks,
    main="log10(Pb), CK predictions, OM covariable",
         panel = function(x, ...) {
            panel.levelplot(x, ...);
            panel.grid(h=-1, v=-1, col="darkgrey")
})
levelplot(ltpb.pred ~ x+y, as.data.frame(k.c2), aspect="iso",
    col.regions=bpy.colors(64), at=breaks,
    main="log10(Pb), CK predictions, Zn covariable",
         panel = function(x, ...) {
            panel.levelplot(x, ...);
            panel.grid(h=-1, v=-1, col="darkgrey")
})
rm(range, breaks)
```

Fig. 29 (left) shows large differences in predictions. The most intricate pattern is for CK with co-variable Zn; this respects the full sample points much better than the other co-variable, because both the feature-space correlation and the spatial co-regionalisation is much stronger for Zn than OM.

10.2 Prediction errors

Task 46 : Compare the kriging predictions errors of $\log_{10}$Pb for OK (§5), CK with OM as co-variable (§7), and CK with Zn as co-variable (§9.4). •

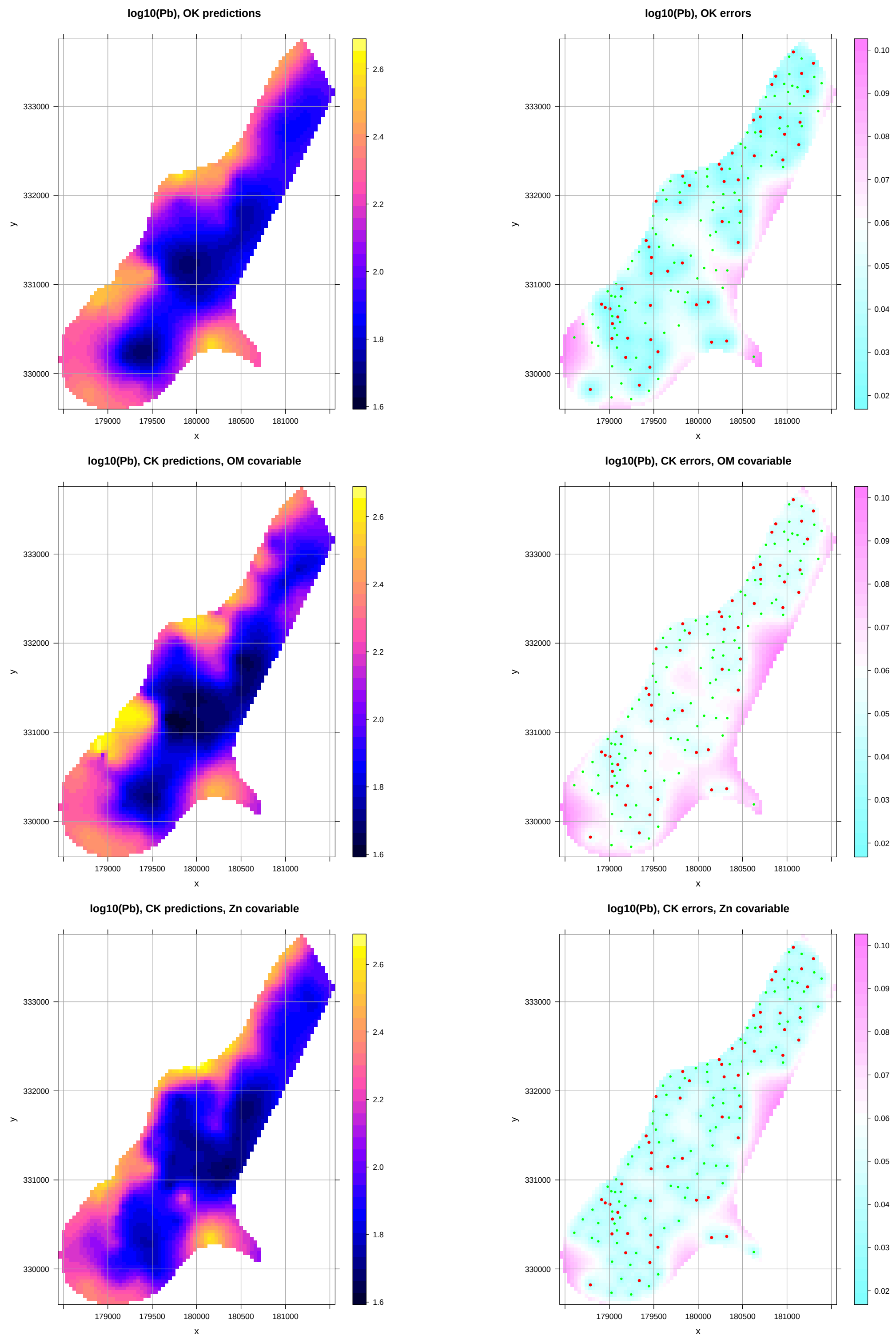
log10(Pb), OK predictions
log10(Pb), OK errors
log10(Pb), CK predictions, OM covariable
log10(Pb), CK errors, OM covariable
log10(Pb), CK predictions, Zn covariable
log10(Pb), CK errors, Zn covariable
x
y
179000
179500
180000
180500
181000
330000
331000
332000
333000
1.6
1.8
2.0
2.2
2.4
2.6
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.10

Figure 29: Kriging predictions and their errors compared, log10(Pb)

R code:

```
        # common colour scale
range <- range(k.o$var1.var, k.c$ltpb.var, k.c2$ltpb.var)
breaks <- seq(range[1], range[2], length=32)
levelplot(var1.var ~ x+y, as.data.frame(k.o), aspect="iso",
col.regions=cm.colors(64), at=breaks, colorkey=T,
main="log10(Pb), OK errors",
         panel = function(x, ...) {
            panel.levelplot(x, ...);
            panel.points(coordinates(meuse), col="green", pch=20, cex=.6);
            panel.points(coordinates(meuse.pb), col="red", pch=20, cex=.8);
            panel.grid(h=-1, v=-1, col="darkgrey")
})
levelplot(ltpb.var ~ x+y, as.data.frame(k.c), aspect="iso",
col.regions=cm.colors(64), at=breaks, colorkey=T,
main="log10(Pb), CK errors, OM covariable",
         panel = function(x, ...) {
            panel.levelplot(x, ...);
            panel.points(coordinates(meuse), col="green", pch=20, cex=.6);
            panel.points(coordinates(meuse.pb), col="red", pch=20, cex=.8);
            panel.grid(h=-1, v=-1, col="darkgrey")
})
levelplot(ltpb.var ~ x+y, as.data.frame(k.c2), aspect="iso",
col.regions=cm.colors(64), at=breaks, colorkey=T,
main="log10(Pb), CK errors, Zn covariable",
         panel = function(x, ...) {
            panel.levelplot(x, ...);
            panel.points(coordinates(meuse), col="green", pch=20, cex=.6);
            panel.points(coordinates(meuse.pb), col="red", pch=20, cex=.8);
            panel.grid(h=-1, v=-1, col="darkgrey")
})
rm(range, breaks)
```

Fig. 29 (right) shows that the OK prediction errors are the smallest of the three near the sample points, whereas CK with co-variable Zn gives the lowest errors averaged across the map.

10.3 Validations

Task 47 : Compare the validations for OK (§5.1), CK with OM as co-variable (§7.1), and CK with Zn as co-variable (§9.5). •

	ME	RMSE	MinE	MaxE
OK	0.0250	0.1660	-0.663	+0.372
CK with OM	0.0148	0.2226	-0.811	+0.555
CK with Zn	0.0043	0.0775	-0.137	+0.228

The CK with Zn as co-variable is far superior to OK in all measures; in turn this is better than CK with OM as co-variable. We see that CK with Zn substantially narrows the range of errors, lowers the RMSE, and has almost no bias at these extra points. This is a very good result.

Task 48 : Plot all three validations and cross-validations. •

R code:

```
     # compute all the validations and x-validations, and differences
k.ok <- krige(ltpb  ~ 1, meuse.pb, meuse.extra, m.ltpb.f)
diff.ok <- k.ok$var1.pred - meuse.extra$ltpb
k.ck <- predict.gstat(g, meuse.extra)
diff.ck <- k.ck$ltpb.pred - meuse.extra$ltpb
k.ck.zn <- predict.gstat(g2, meuse.extra)
diff.ck.zn <- k.ck.zn$ltpb.pred - meuse.extra$ltpb
cv.o <- krige.cv(ltpb ~ 1, meuse.pb, model=m.ltpb.f, nfold=nrow(meuse.pb))
cv.c <- gstat.cv(g)
cv.c2 <- gstat.cv(g2)
        # value that corresponds to cex=3
extreme <- max(abs(range(diff.ok, diff.ck, diff.ck.zn,
        cv.o$residual, cv.c$residual, cv.c2$residual)))
        # max. cex in each case proportional to its largest
```

R code:

```
      # display all plots
d <- SpatialPointsDataFrame(coordinates(k.ok), data=as.data.frame(diff.ok))
b.ok <- bubble(d,
         main="OK validation errors, log10(Pb)",
         maxsize = 3 * (max(abs(range(diff.ok))))/extreme,
         panel = function(x, ...) {
            panel.xyplot(x, ...);
            panel.grid(h=-1, v=-1, col="darkgrey")})
d <- SpatialPointsDataFrame(coordinates(k.ck), data=as.data.frame(diff.ck))
b.ck <- bubble(d,
          main="CK validation errors, log10(Pb)~x+y, , co-variable log10(OM)",
         maxsize = 3 * (max(abs(range(diff.ck))))/extreme,
          panel = function(x, ...) {
            panel.xyplot(x, ...);
            panel.grid(h=-1, v=-1, col="darkgrey")})
d <- SpatialPointsDataFrame(coordinates(k.ck.zn), data=as.data.frame(diff.ck.zn))
b.ck.zn <- bubble(d,
          main="CK validation errors, log10(Pb), co-variable log10(Zn)",
         maxsize = 3 * (max(abs(range(diff.ck.zn))))/extreme,
          panel = function(x, ...) {
            panel.xyplot(x, ...);
            panel.grid(h=-1, v=-1, col="darkgrey")})
d <- SpatialPointsDataFrame(coordinates(cv.o), data=as.data.frame(cv.o$residual))
b.x.ok <- bubble(d,
         main="OK cross-validation errors, log10(Pb)",
         maxsize = 3 * (max(abs(range(diff.ok))))/extreme, col=c(4,5),
         panel = function(x, ...) {
            panel.xyplot(x, ...);
            panel.grid(h=-1, v=-1, col="darkgrey")})
d <- SpatialPointsDataFrame(coordinates(cv.c), data=as.data.frame(cv.c$residual))
b.x.ck <- bubble(d,
          main="CK cross-validation errors, log10(Pb), co-variable log10(OM)",
         maxsize = 3 * (max(abs(range(diff.ck))))/extreme, col=c(4,5),
          panel = function(x, ...) {
            panel.xyplot(x, ...);
            panel.grid(h=-1, v=-1, col="darkgrey")})
d <- SpatialPointsDataFrame(coordinates(cv.c2), data=as.data.frame(cv.c2$residual))
b.x.ck.zn <- bubble(d,
          main="CK cross-validation errors, log10(Pb), co-variable log10(Zn)",
         maxsize = 3 * (max(abs(range(diff.ck.zn))))/extreme, col=c(4,5),
          panel = function(x, ...) {
            panel.xyplot(x, ...);
            panel.grid(h=-1, v=-1, col="darkgrey")})
print(b.ok, split=c(1,1,2,3), more=T)
print(b.ck, split=c(1,2,2,3), more=T)
print(b.ck.zn, split=c(1,3,2,3), more=T)
print(b.x.ok, split=c(2,1,2,3), more=T)
print(b.x.ck, split=c(2,2,2,3), more=T)
print(b.x.ck.zn, split=c(2,3,2,3), more=F)
rm(k.ok, k.ck, k.ck.zn)
rm(diff.ok, diff.ck, diff.ck.zn, extreme)
rm(b.ok, b.ck, b.ck.zn, b.x.ok, b.x.ck, b.x.ck.zn)
```

Fig. 30 (left) shows bubble plots of the validation errors; this is a summary of Figs. 7, 14, and 24 (left), but with a single symbol size range for all three figures. The superiority of CK with Zn as co-variable is clear.

10.4 Cross-validations

Task 49 : Compare the cross-validations for OK (§5.2), CK with OM as co-variable (§7.2), and CK with Zn as co-variable (§9.6). •

	ME	RMSE	MSDR
OK	+0.0037	0.2369	00.88
CK with OM	+0.0059	0.2093	12.62
CK with Zn	-0.0035	0.0807	00.32

CK with Zn as co-variable performs much better on every measure than CK with OM as co-variable. CK with Zn is better than OK except that it underestimates the prediction error at the cross-validation points.

Fig. 30 (right) shows bubble plots of the cross-validation errors; this is a summary of Figs. 7, 14, and 24 (right), but with a single symbol size range for all three figures. The superiority of CK with Zn as co-variable is clear.

11 Conclusion

This technical note has shown how to perform co-kriging using the `gstat` package of the R environment for statistical computing and visualisation, and how to evaluate its success by validation and cross-validation.

In the particular case studied here, the use of organic matter as the co-variable for co-kriging the target variable did not in general improve predictions compared to univariate ordinary kriging. This was despite the fairly feature-space good correlation between the two variables and the fairly good linear model of co-regionalisation that could be fitted to the two variables. This shows that the more sophisticated technique is not always better. The underlying reasons for the poor relative performance of co-kriging in this case are (1) that the feature-space correlation between OM and Pb is too weak (§6.1), explaining only about 38% of the variabilty; (2) the spatial structures of the two variables were somewhat different, so that the linear model of co-regionalization was not completely appropriate.

Using Zn as the co-variable, with a feature-space correlation that explained about 95% of the variability and also had a good model of co-regionalisation with the target variable, did result in a much-improved

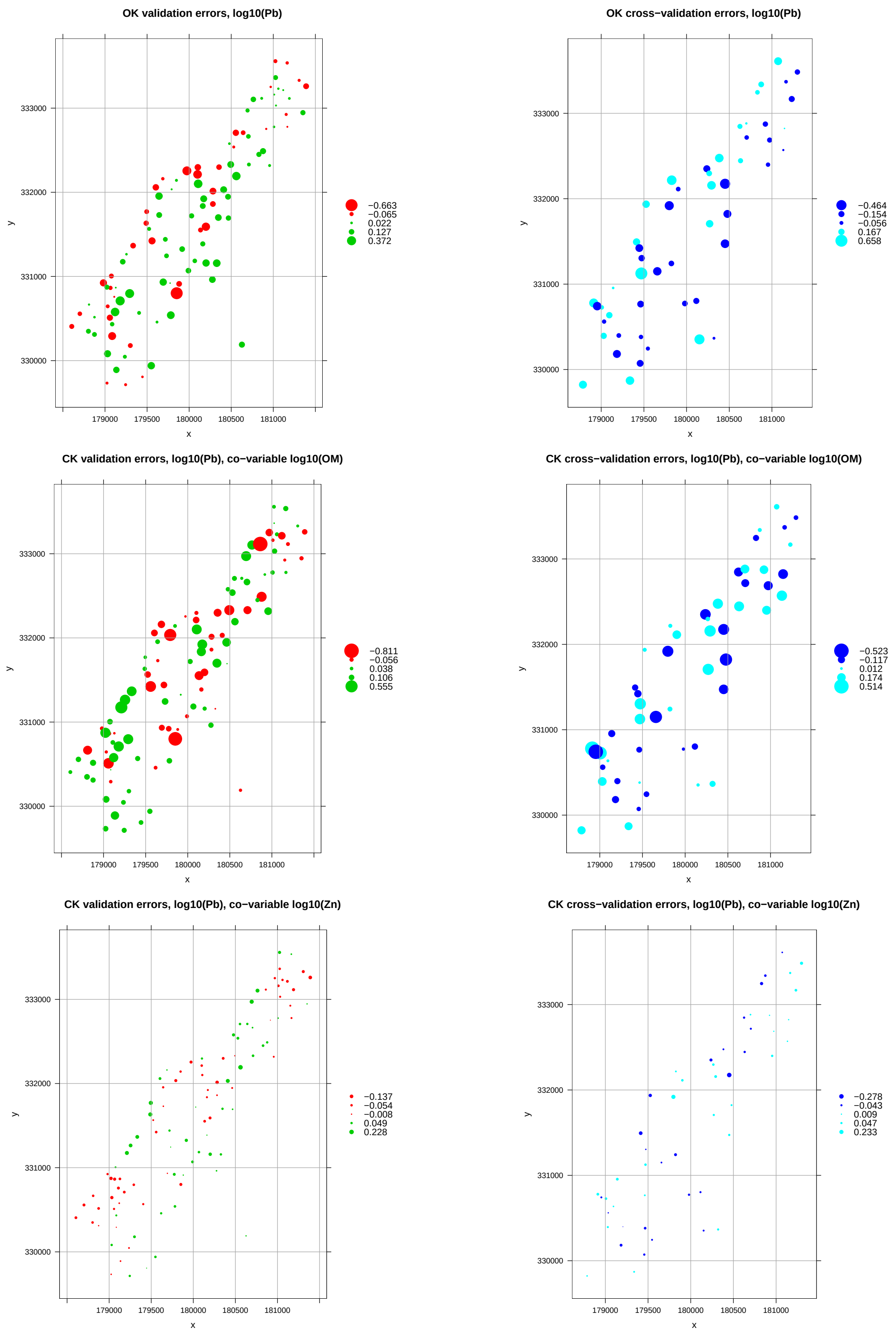
OK validation errors, log10(Pb)
OK cross−validation errors, log10(Pb)
CK validation errors, log10(Pb), co−variable log10(OM)
CK cross−validation errors, log10(Pb), co−variable log10(OM)
CK validation errors, log10(Pb), co−variable log10(Zn)
CK cross−validation errors, log10(Pb), co−variable log10(Zn)
x
y
179000
179500
180000
180500
181000
330000
331000
332000
333000
−0.663
−0.065
0.022
0.127
0.372
−0.464
−0.154
−0.056
0.167
0.658
−0.811
−0.056
0.038
0.106
0.555
−0.523
−0.117
0.012
0.174
0.514
−0.137
−0.054
−0.008
0.049
0.228
−0.278
−0.043
0.009
0.047
0.233

Figure 30: Validation and cross-validation errors compared, log10(Pb)

map, especially away from the sample points of the (undersampled) target variable (§9).

There is much more to co-regionalisation and co-kriging than is explained in these notes, for example indicator co-kriging, multivariate co-kriging, non-linear models of co-regionalisation, and universal co-kriging. The reader is encouraged to consult relevant texts [e.g. 17, 7, 2, 18, 4] and then experiment with `gstat`. Modelling the co-regionalisation is also tricky and should be approached with caution [e.g. 5]; remember that all kriging approaches are model-based: bad models lead to bad predictions.

References

[1] WS Cleveland. *Visualizing data*. AT&T Bell Laboratories; Hobart Press, Murray Hill, N.J., 1993.

[2] P Goovaerts. *Geostatistics for natural resources evaluation*. Applied Geostatistics. Oxford University Press, New York; Oxford, 1997.

[3] R Ihaka and R Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996.

[4] EH Isaaks and RM Srivastava. *An introduction to applied geostatistics*. Oxford University Press, New York, 1990.

[5] RM Lark. Two robust estimators of the cross-variogram for multivariate geostatistical analysis of soil properties. *European Journal of Soil Science*, 54(1):187–201, 2003.

[6] MB McBride. *Environmental chemistry of soils*. Oxford University Press, New York, 1994.

[7] A Papritz and A Stein. Spatial prediction by linear kriging. In A Stein, F van der Meer, and B G F Gorte, editors, *Spatial statistics for remote sensing*, pages 83–113. Kluwer Academic, Dordrecht, 1999.

[8] EJ Pebesma. *Gstat User's Manual*. Dept. of Physical Geography, Utrecht University, Utrecht, version 2.3.3 edition, 2001.

[9] EJ Pebesma and RS Bivand. Classes and methods for spatial data in R. *R News*, 5(2):9–13, November 2005. URL http://CRAN.R-project.org/doc/Rnews/.

[10] EJ Pebesma and CG Wesseling. Gstat: a program for geostatistical modelling, prediction and simulation. *Computers & Geosciences*, 24 (1):17–31, 1998.

[11] PJ Ribeiro, Jr. and PJ Diggle. geoR: A package for geostatistical analysis. *R News*, 1(2):14–18, June 2001.

[12] MGJ Rikken and RPG van Rijn. Soil pollution with heavy metals – an inquiry into spatial variation, cost of mapping and the risk evaluation of copper, cadmium, lead and zinc in the floodplains of the Meuse west of Stein, the Netherlands. Doctoraalveldwerkverslag, Dept. of Physical Geography, Utrecht University, 1993.

[13] DG Rossiter. *Introduction to the R Project for Statistical Computing for use at ITC*. International Institute for Geo-information Science & Earth Observation (ITC), Enschede (NL), 2.7 edition, 2006. URL http://www.itc.nl/personal/rossiter/teach/R/RIntro_ITC.pdf.

[14] D Sarkar. Lattice. *R News*, 2(2):19–23, June 2002.

[15] P Schachtschabel, H-P Blume, G Brümmer, KH Hartge, and U Schwertmann. *Lehrbuch der Bodenkunde (Scheffer, Schachtschabel)*. Enke, Stuttgart, 14 edition, 1998.

[16] WN Venables and BD Ripley. *Modern applied statistics with S*. Springer-Verlag, New York, fourth edition, 2002.

[17] H Wackernagel. *Multivariate geostatistics : an introduction with applications*. Springer, Berlin ; New York, 2nd edition, 1998.

[18] R Webster and MA Oliver. *Geostatistics for environmental scientists*. Wiley & Sons, Chichester, 2001.

A Producing the figures for this document

Many figures in this document combine several plots into one. This can easily be accomplished with the `lattice` graphics package. The basic idea is to create plots with any `lattice` method, including `levelplot()`, `xyplot()`, and `histogram()`, but instead of displaying them directly, *saving* them (using the assignment operator `<-`) as *trellis objects*; this is the object type created by `lattice` graphics methods[5]. They are then printed with the `print()` method for `trellis` objects, using the optional `split=` argument to split the display and position each saved plot within it.

Fig. 5 was created using the `plot.kresults` function shows in the next section §A.1; see the code listing.

First, two `trellis` objects (`plot.kr` and `plot.kre`) are created, using the `levelplot()` method. Then they are placed into a one-row, two-column split display: The `split = c(1,1,2,1)` optional argument says to place this figure in position `(1,1)` within a matrix with dimensions `(2,1)` in (x, y) i.e. (columns, rows). The `more=T` argument indicates that more saved objects will be added to the current plot; `more=F` indicates that the plot is finished.

Plots can be saved for later printing or incorporation into documents by writing them to a graphics "device" that represents a file. Here is an example of how to write a PDF file.

[5] The reason for this name is historical: S-PLUS has a 'trellis' graphics environment, on which the R package 'lattice' is based.

R code:
```
        # open the graphics device
pdf("graph/fig_pb_ok.pdf", h=6, w=12)
        # put plotting commands here
        # ...
        # close the graphics device
dev.off()
```

The optional h= and w= arguments specify the dimensions of the plot, in inches (1” = 2.54 cm exactly). In this case the plots are side-by-side so the plot is wider than tall.

A.1 Plotting functions

Note: These were written before sp package was developed; this package includes the spplot method which would probably make this code simpler and easier to understand.

These functions should be loaded into your workspace as follows:

R code:
```
source("ck_plotfns.R")
```

The code is reproduced here for your reference; however, rather than attempt to cut-and-paste from here, it is better to download file ck_plotfns.R from the author’s website where this tutorial is also found[6].

```
### R source code, file ck_plotfns.R
### author: DG Rossiter, ITC Enschede (NL) rossiter@itc.nl
###
### to accompany Technical Note "Co-kriging with the gstat package
###    of the R environment for statistical computing"
### plot Kriging results: predictions and variances
##     with postplot of sample/subsample points on predictions
##        colour: bpy, grey, red
##     and locations of sample/subsample points on prediction variance
##        colour: cm, green, red
##  arguments
##     kr.o     SpatialPointsDataFrame from kriging
##         coordinates must be named x, y
##     var1     prefix for field names *.pred, *.var, e.g. "var1"
##     samp.pts  SpatialPointsDataFrame sample points
##     subsamp.pts  SpatialPointsDataFrame subsample points
##     f       field name (quoted) or number for point data values
##     title
plot.kresults <- function(kr.o, var1, samp.pts, subsamp.pts,
                          f=1, title="") {
```

[6] http://www.itc.nl/personal/rossiter/pubs/list.html#pubs_m_R

```
  to.eval <- paste("plot.kr <- levelplot(",
                 paste(var1,"pred",sep="."),
                 " ~ x+y, as.data.frame(kr.o),
          aspect='iso',
          col.regions=bpy.colors(64), cut=32,
          main=paste(title, 'Prediction'),
              panel = function(x, ...) {
                  panel.levelplot(x, ...);
                  panel.points(coordinates(samp.pts), col='grey', pch=20,
                       # log scale, but still show minimum
                       cex=1.6 * (log10(samp.pts[[f]]) -
                              0.9 * min(log10(samp.pts[[f]]))));
                  panel.points(coordinates(subsamp.pts), col='red', pch=20,
                       cex=1.6 * (log10(subsamp.pts[[f]]) -
                              0.9 * min(log10(subsamp.pts[[f]]))));
                  panel.grid(h=-1, v=-1, col='')
           })"  );
    eval(parse(text=to.eval));
    to.eval <- paste("plot.kr.e <- levelplot(",
                   paste(var1,"var",sep="."),
                   "~ x+y, as.data.frame(kr.o),
      aspect='iso',
        col.regions=cm.colors(64), cut=32,
        main=paste(title, 'Prediction variance'),
             panel = function(x, ...) {
                panel.levelplot(x, ...);
                panel.points(coordinates(samp.pts), col='green', pch=20,
                     cex=.6);
                panel.points(coordinates(subsamp.pts), col='red', pch=20,
                     cex=.8); # subsample points slightly larger
                panel.grid(h=-1, v=-1, col='darkgrey')
    })"  );
    eval(parse(text=to.eval));
    print(plot.kr, split = c(1,1,2,1), more=T);
    print(plot.kr.e, split = c(2,1,2,1), more=F)
}

### plot Kriging validation and cross-validation errors
##       colours: validation: bubble() default: palette()[2:3]
##       colours: x-valid: palette()[4:5]
##  arguments
##     kv.o     SpatialPointsDataFrame from kriging to validation points
##     var1     prefix for kriging field name *.pred, e.g. "var1"
##     valid.pts  SpatialPointsDataFrame with validation points
##     f        field name (quoted) or number for point data values
##     cv.o     SpatialPointsDataFrame from x-validation kriging
##     title
plot.valids <- function(kv.o, var1, valid.pts, f, cv.o, title="") {
                                      # validation errors
  to.eval <- paste("diff <- kv.o$", paste(var1,"pred",sep="."),
                 " - valid.pts[[f]]")
  eval(parse(text=to.eval))
  extreme <- max(abs(range(diff, as.data.frame(cv.o)$residual)))
  d <- SpatialPointsDataFrame(kv.o, data=as.data.frame(diff))
  b1 <- bubble(d,
             main=paste(title,"Validation errors"),
             maxsize = 3 * (max(abs(range(diff))))/extreme,
             panel = function(x, ...) {
                panel.xyplot(x, ...);
                panel.grid(h=-1, v=-1, col="darkgrey")}
               )
  b2 <- bubble(cv.o, z="residual",
               main=paste(title,"Cross-validation errors"), col=c(4,5),
             maxsize = 3 * (max(abs(range(cv.o$residual))))/extreme,
             panel = function(x, ...) {
                panel.xyplot(x, ...);
                panel.grid(h=-1, v=-1, col="darkgrey")}
               )
  print(b1, split=c(1, 1, 2, 1), more=T)
  print(b2, split=c(2, 1, 2, 1), more=F)
}

### compare Kriging results: predictions or prediction variance
##     with locations of sample/subsample points on predictions
##        colour: bpy, grey, red
##     and locations of sample/subsample points on prediction variance
##        colour: cm, green, red
##  arguments
##     k1.o, k2.o     SpatialPointsDataFrame'sfrom kriging
##         coordinates must be named x, y
##     var1.1, var1.2 prefix for field names *.pred, *.var, e.g. "var1"
##         in the two objects
##     samp.pts  SpatialPointsDataFrame sample points
##     subsamp.pts  SpatialPointsDataFrame subsample points
##     type   what to compare: "pred" or "var"
##     title.1, title.2   titles for the two kriging objects
plot.cf <- function(k1.o, k2.o, var1.1, var1.2, samp.pts, subsamp.pts,
    type="pred", title.1="", title.2="") {
            # common scale
  to.eval <- paste("range <- range(k1.o$", paste(var1.1, type, sep="."),
                   ", k2.o$", paste(var1.2, type, sep="."), ")", sep="")
  eval(parse(text=to.eval));  # makes range
  breaks <- seq(range[1], range[2], length=32)
  to.eval <- paste("plot.k1 <- levelplot(",
                   paste(var1.1,type,sep="."),
                   " ~ x+y, as.data.frame(k1.o), aspect='iso',
        col.regions=', ifelse(type=='pred', 'bpy.colors', 'cm.colors'),
        '(64), at=breaks,
        main=title.1,
        panel = function(x, ...) {
          panel.levelplot(x, ...);
          panel.grid(h=-1, v=-1, col='darkgrey')})"
                   );
  eval(parse(text=to.eval));  # makes plot.k1
  to.eval <- paste("plot.k2 <- levelplot(",
                   paste(var1.2,type,sep="."),
                   " ~ x+y, as.data.frame(k2.o), aspect='iso',
        col.regions=', ifelse(type=='pred', 'bpy.colors', 'cm.colors'),
        '(64), at=breaks,
        main=title.2,
        panel = function(x, ...) {
          panel.levelplot(x, ...);
          panel.grid(h=-1, v=-1, col='darkgrey')})"
                  );
  eval(parse(text=to.eval));  # makes plot.k2
  to.eval <- paste("diff <- k2.o$", paste(var1.2, type, sep="."),
                   "- k1.o$", paste(var1.1, type, sep="."), sep="")
  eval(parse(text=to.eval));  # makes diff
  tmp <- data.frame(x = coordinates(k1.o)[,"x"],
            y = coordinates(k2.o)[,"y"], diff)
  plot.diff <-
         levelplot(diff ~ x+y, tmp, aspect="iso",
                   col.regions=topo.colors(64), cut=32,
                   main="Difference",
                   panel = function(x, ...) {
                      panel.levelplot(x, ...);
                      panel.points(coordinates(samp.pts), col="white",
                                   pch=20, cex=.6);
                      panel.points(coordinates(subsamp.pts), col="red",
                                   pch=20, cex=.8);
                      panel.grid(h=-1, v=-1, col="darkgrey")
    })
           # display the plots in one figure
    print(plot.k1, split = c(1,1,2,2), more=T)
    print(plot.k2, split = c(2,1,2,2), more=T)
    print(plot.diff, split = c(1,2,2,2), more=F)
}
```