A minimal introduction to geostatistics with R/gstat

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

October 23, 2012

Contents

Version 1.4. Copyright © 2010–2012 D G Rossiter. All rights reserved. Reproduction and dissemination of the work as a whole (not parts) freely permitted if this original copyright notice is included. Sale or placement on a web site where payment must be made to access this document is strictly prohibited. To adapt or translate please contact the author (http://www.itc.nl/personal/rossiter).

This document is a brief introduction to exploratory and inferential geostatistical analysis. At the same time, it introduces the **R environment for statistical computing and visualisation** [7] and the `gstat` [11] and `sp` R packages. Sections 1–7 were designed to be used in a two-day interactive tutorial course[1]; sections 10–12 were added for a third and fourth day, to give some appreciation for more sophisticated geostatistical analysis.

The exercise assumes no prior knowledge of either geostatistics nor the R environment. It is organized as a set of **discussions**, **tasks**, **R code** to complete the tasks, and self-study **questions** (with answers) to test your understanding.

After completing the exercise, you will have seen just the barest minimum of the R environment, and the simplest concepts of geostatistics. There are many resources for going futher, see §13, which you should consult before undertaking your own geostatistical analyses.

> **Note:** The same analyses can be performed in other (geo-)statistical computer programs; we choose R (1) to introduce students to this extremely powerful environment; (2) to allow the student to continue with this open-source, freely-available, multi-platform enviroment; (3) to show the analysis step-by-step, thereby leading to deeper understanding than in a "press the button" computing environment.

1 Starting R

R is an open-source environment for data manipulation, statistical analysis, and visualization. There are versions for MS-Windows, Mac OS/X, and various flavours of Unix. On MS-Windows, the program is named `RGui.exe`; start it in the usual way.

The simplest way to interact with the R environment is by typing **commands** at the "R console" **command line**.

You can use R as a simple calculator, by typing an **expression**:

Task 1 : Compute the number of radians in one circular degree. •

In this document, input and output are shown as follows:

```
> 2 * pi/360

[1] 0.017453
```

This means: when R prompts for input:

```
>
```

you type an expression, in this case `2*pi/360`. Then R shows the result; in this case it is a one-element vector (shown by the `[1]`), with the value 0.017453. Note that `pi` is a constant value known to R.

[1] Originally given at Institut Teknologi Bandung, Indonesia in August 2010

R is a **modular** system: there is a **base package** and some **standard** packages that are always loaded when R is started. In addition, there are several thousand **contributed packages** to perform specific tasks. In this exercise we will use the `gstat` package for geostatistical modelling, prediction and simulation, contributed by Pebesma [10].

Task 2 : Load the `gstat` package. •

The `require` function loads a package into the **R workspace**:

```
> require(gstat)
```

Note that `gstat` also loaded a package that it needs, `sp`. This package provides support for **spatial data**, and is described in the text of Bivand *et al.* [1].

2 Loading and examining the Meuse dataset

Most R packages include **sample datasets** that illustrate the functionality of the package[2]. For this exercise we will use the Meuse soil pollution data set distributed with the `sp` package. This dataset is described in Appendix §A; for now it is sufficient to know that the dataset consists of soil samples which were analyzed for their concentration of toxic heavy metals, along with the sample **location**.

Task 3 : Load the `meuse` dataset into the workspace. •

The `data` function loads a dataset. We show the contents of the workspace before and after with the `ls` "list objects" function:

```
> ls()

character(0)

> data(meuse)
> ls()

[1] "meuse"
```

Q1 : *What objects were in the workspace before and after loading the `meuse` dataset?* *Jump to A1* •

Task 4 : Examine the structure of the Meuse dataset. •

The `str` "structure" function shows the structure of an R object:

```
> str(meuse)
```

[2] R also provides functions for loading data from external sources, e.g., `read.table`

```
'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 7.8 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 ...
 $ soil    : Factor w/ 3 levels "1","2","3": 1 1 1 2 2 2 2 1 1 2 ...
 $ lime    : Factor w/ 2 levels "0","1": 2 2 2 1 1 1 1 1 1 1 ...
 $ landuse: Factor w/ 15 levels "Aa","Ab","Ag",..: 4 4 4 11 4 11 4 2 2 15 ...
 $ dist.m : num  50 30 150 270 380 470 240 120 240 420 ...
```

This is the typical R **data frame** structure: a **matrix** with **named columns** which are database **fields** ("variables"), and **rows** which are database **records**.

Q2 : *How many observations (cases) and fields (variables) are there?* *Jump to A2* •

Notice that some variables are **continuous** (e.g., the metal concentrations) and some are **classified** (e.g., the flood frequency `ffreq`); these are called R **factors**.

Q3 : *Which fields show that this is **spatial** data?* *Jump to A3* •

All R functions and built-in datasets have **on-line help**.

Task 5 : View the on-line help information for the Meuse dataset. •

The `?` "help" function displays help on a function, method or built-in dataset.

```
> `?`(meuse)
```

On most systems this will display in a browser window.

Q4 : *What are the units of measure of the metals?* *Jump to A4* •

3 Univariate EDA and transformation

Before considering the **spatial** aspects of the data using `gstat`, we briefly look at the data as **non-spatial** dataset, i.e., considering **feature** (or, attribute) **space**.

Task 6 : Display the actual data values for zinc content, both in sample and sort order. •

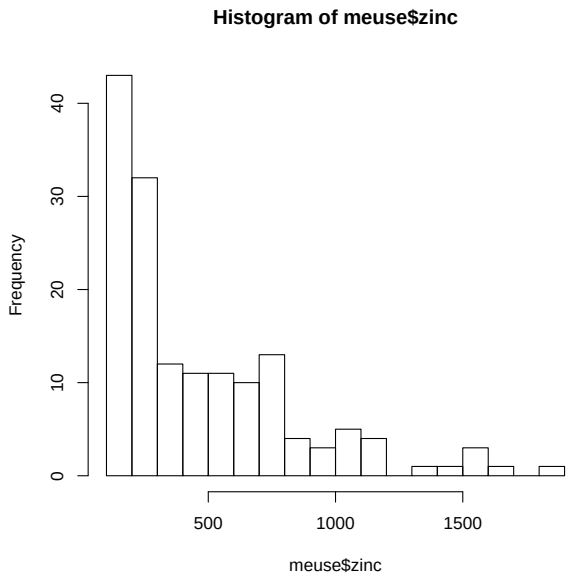
To reference a field in a dataframe, use the notation `dataframe$fieldname`:

```
> meuse$zinc

  [1] 1022 1141  640  257  269  281  346  406  347  183  189  251
 [13] 1096  504  326 1032  606  711  735 1052  673  402  343  218
 [25]  200  194  207  180  240  180  208  198  250  192  213  321
 [37]  569  833  906 1454  298  167  176  258  746  746  464  365
 [49]  282  375  222  812 1548 1839 1528  933  432  550 1571 1190
 [61]  907  761  659  643  801  784 1060  119  778  703  676  793
 [73]  685  593  549  680  539  560 1136 1383 1161 1672  765  279
 [85]  241  317  545  505  420  332  400  553  577  155  224  180
 [97]  226  186  198  187  199  157  203  143  136  117  113  130
[109]  192  240  221  140  128  166  191  232  203  722  210  198
[121]  139  253  703  832  262  142  119  152  415  474  126  210
[133]  220  133  141  158  129  206  451  296  189  154  169  403
[145]  471  612  601  783  258  214  166  496  342  162  375

> sort(meuse$zinc)

  [1]  113  117  119  119  126  128  129  130  133  136  139  140
 [13]  141  142  143  152  154  155  157  158  162  166  166  167
 [25]  169  176  180  180  180  183  186  187  189  189  191  192
 [37]  192  194  198  198  198  199  200  203  203  206  207  208
 [49]  210  210  213  214  218  220  221  222  224  226  232  240
 [61]  240  241  250  251  253  257  258  258  262  269  279  281
 [73]  282  296  298  317  321  326  332  342  343  346  347  365
 [85]  375  375  400  402  403  406  415  420  432  451  464  471
 [97]  474  496  504  505  539  545  549  550  553  560  569  577
[109]  593  601  606  612  640  643  659  673  676  680  685  703
[121]  703  711  722  735  746  746  761  765  778  783  784  793
[133]  801  812  832  833  906  907  933 1022 1032 1052 1060 1096
[145] 1136 1141 1161 1190 1383 1454 1528 1548 1571 1672 1839
```

Task 7 : Display a stem-and-leaf plot, a histogram, and a five-number summary of the zinc content. •

```
> stem(meuse$zinc)

  The decimal point is 2 digit(s) to the right of the |

   1 | 122233333444444556666777788888999999999
   2 | 00000001111111222223344455566667888
   3 | 0022334455788
   4 | 00012235677
   5 | 001455556789
   6 | 0114467889
   7 | 0012455678889
   8 | 0133
   9 | 113
  10 | 2356
  11 | 04469
  12 |
  13 | 8
  14 | 5
```

```
  15 | 357
  16 | 7
  17 |
  18 | 4

> hist(meuse$zinc, breaks = 16)
> summary(meuse$zinc)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    113     198     326     470     674    1840
```



Note in the `hist` "histogram" graphics function the use of the optional `breaks` **argument** to specify the number of histogram bins.

Q5 : *Describe the distribution of this variable. Is it symmetric or skewed? Does it appear to come from one population? Do there appear to be any unusual values ("outliers")?* *Jump to A5* •

Q6 : *What are the minimum, first quartile, median, third quartile, and maximum lead concentrations in the sample set?* *Jump to A6* •

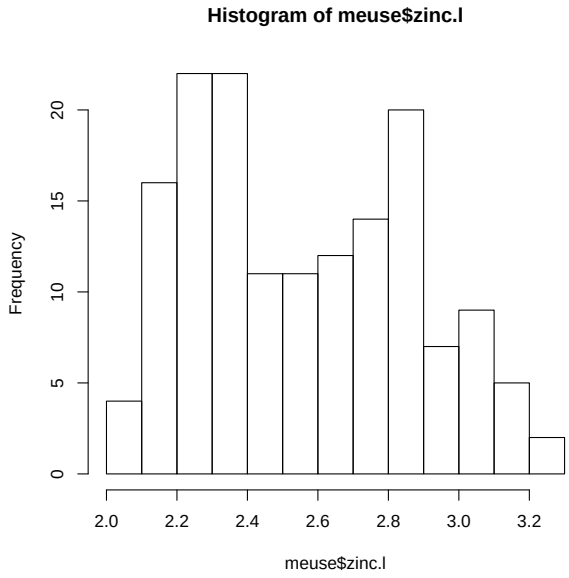
Q7 : *Compare the mean and median. What does this imply about the distribution of the variable?* *Jump to A7* •

3.1 Transformation

It's clear that the distribution this variable is far from symmetrical. A common transform for a highly-skewed distribution is the logarithm. The transformed variable is easier to visualise and is better behaved in various types of models.

Task 8 : Log-transform the variable `zinc` and repeat the above analyses and questions. Use base-10 logarithms, as they are easy to understand in the original units (e.g. if $\log_{10}(\text{zinc}) = 3$, zinc = $10^3 = 1000$). Repeat the summary and histogram. •

```
> meuse$zinc.l <- log10(meuse$zinc)
> hist(meuse$zinc.l, breaks = 16)
> summary(meuse$zinc.l)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2.05    2.30    2.51    2.56    2.83    3.26
```



Note the use of the `<-` **assignment** operator. This computes the expression on its right-hand side (here, `log10(meuse$zinc)`) and then places it in the object on its left-hand side (here, `meuse$zinc.l`). This object is the field named `zinc.l` in the `meuse` data frame; it doesn't exist yet, so R creates it, and puts the results of the expression in it.

This example illustrates another key feature of R: many operations are **vectorized**. This means that they apply to all elements of a vector in **parallel**. In this example the field `meuse$zinc` is a 155-element vector; so the `log10` function is applied to each element, resulting in a 155-element transformed vector. We can see this by looking at the first few, using the `head` function to show the "head" of a vector:

```
> head(meuse$zinc)

[1] 1022 1141  640  257  269  281

> head(meuse$zinc.l)

[1] 3.0095 3.0573 2.8062 2.4099 2.4298 2.4487
```

Q8 : *Does the transform make the variable more symmetric? Does it remove presumed outliers? Is there now evidence for more than one population?* *Jump to A8* •

All four metals have similar-shaped distributions, so they should all be transformed for further analysis.

Task 9 : Log-transform all four metals and attach them as new fields to the data frame. •

```
> meuse$cadmium.l <- log10(meuse$cadmium)
> meuse$copper.l <- log10(meuse$copper)
> meuse$lead.l <- log10(meuse$lead)
> str(meuse)

'data.frame':        155 obs. of  18 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 7.8 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 ...
 $ soil      : Factor w/ 3 levels "1","2","3": 1 1 1 2 2 2 2 1 1 2 ...
 $ lime      : Factor w/ 2 levels "0","1": 2 2 2 1 1 1 1 1 1 1 ...
 $ landuse   : Factor w/ 15 levels "Aa","Ab","Ag",..: 4 4 4 11 4 11 4 2 2 15 ...
 $ dist.m    : num  50 30 150 270 380 470 240 120 240 420 ...
 $ zinc.l    : num  3.01 3.06 2.81 2.41 2.43 ...
 $ cadmium.l: num  1.068 0.934 0.813 0.415 0.447 ...
 $ copper.l : num  1.93 1.91 1.83 1.91 1.68 ...
 $ lead.l    : num  2.48 2.44 2.3 2.06 2.07 ...
```

Notice the four new fields.

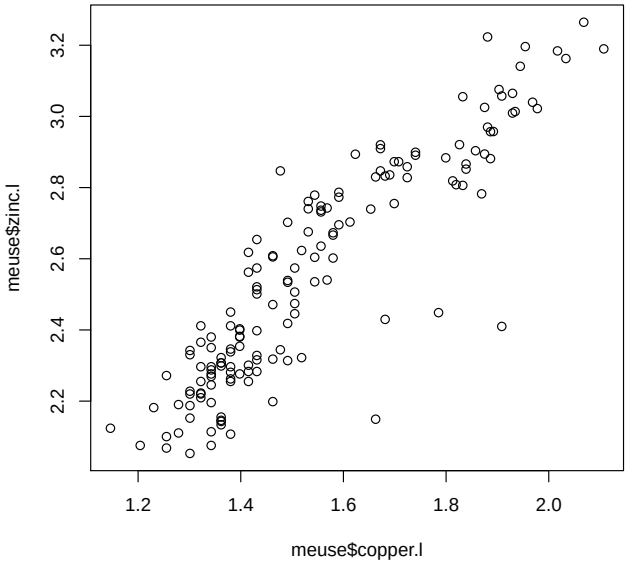
4 Non-spatial Bivariate EDA

We continue the analysis of **feature** (attribute) space by considering the relation between **two** variables.

Task 10 : Show a **scatterplot** of the relation between log-transformed Zn and Cu. •

The generic `plot` method produces a scatterplot if its argument is of the form `var.y ~ var.x`, the `~` ("tilde") formula operator symbolizing the **dependence** of the left-hand side on the right-hand side. This is a simple example of a **model formula**.

```
> plot(meuse$zinc.l ~ meuse$copper.l)
```



This graph looks like a 2D "map" . . . in fact it is, considering the range of the two variables as the "coordinates". In mathematical terms it is a "space", from whence the term **feature** (or, attribute; or variable) space.

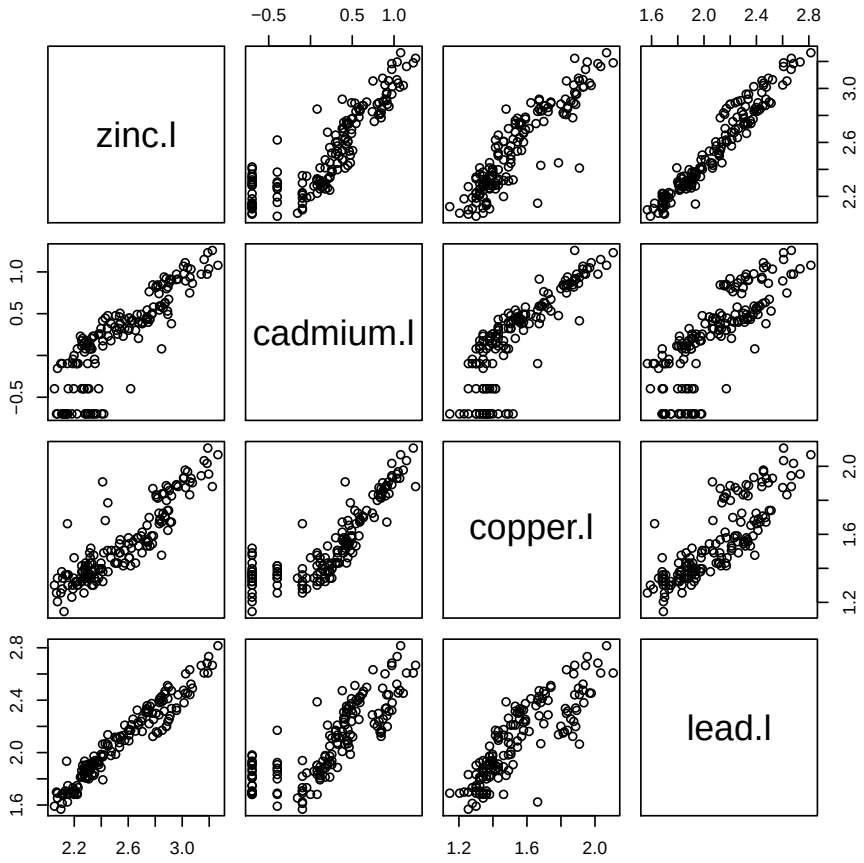
Q9 : *Do the two variables appear to be related in feature space? Describe the relation. Are there any observations that do not fit the general pattern?* *Jump to A9* •

Challenge: Find the observations that do not fit the general pattern of the copper vs. zinc relation (hint: try the `which` function).

Task 11 : Plot all four metals against each other. •

The `pairs` graphics function plots any number of vectors against each other:

```
> pairs(meuse[, 15:18])
```



This example illustrates how R can access a dataframe as a matrix. The notation `meuse[,15:18]` means: object `meuse`, all rows (the blank before the `,`), columns 15 through 18 (the `:` meaning from . . . to). This is standard matrix notation. Note how the **rows** of the matrix are the observations, and the **columns** are the fields.

Q10 : *Are all the metals related in feature space? Describe the relation.* *Jump to A10* •

5 Feature-space modelling

A common **non-spatial** approach to prediction is to **model** one variables' distribution (the **dependent** or **response** variable) by another, either continuous or categorical. This is sometimes called "regression modelling".

In this case we suspect that the flooding frequency affects the metal concentration; this would be evidence that the metals are brought from upstream industry.

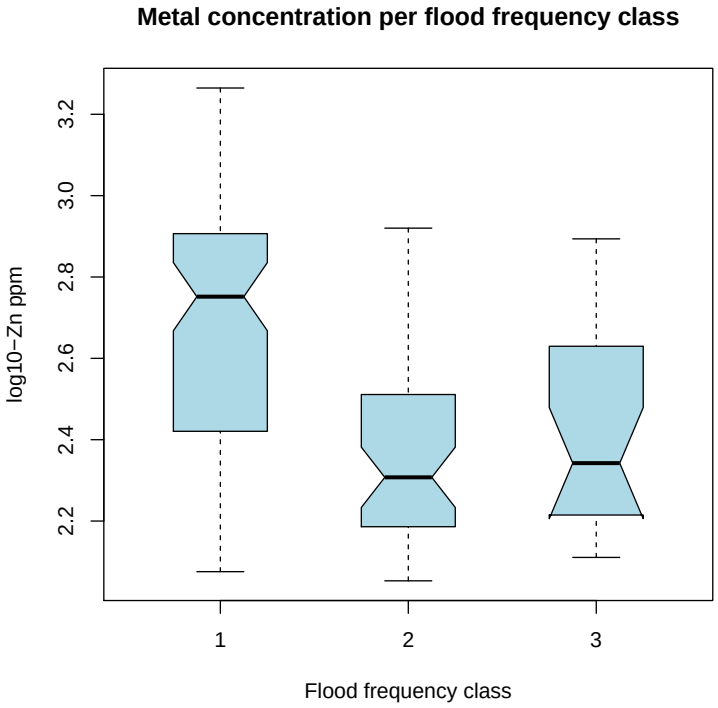
Task 12 : Model the metal concentration of log-Zn from the flooding frequency. •

First, find out how many observations are in each class, using the `table` function:

```
> table(meuse$ffreq)

 1  2  3
84 48 23
```

Second, display a **grouped boxplot** of log-Zn in each class using the `boxplot` function:

```
> boxplot(meuse$zinc.l ~ meuse$ffreq, xlab = "Flood frequency class",
+     ylab = "log10-Zn ppm", main = "Metal concentration per flood frequency clas
+     boxwex = 0.5, col = "lightblue", notch = T)
```



This example shows how many **optional arguments** can be used to enhance a plot.

Q11 : *Describe the relation between the different flood frequencies and the metal concentration.* *Jump to A11* •

Third, build a **linear model**, using the `lm` function; note the use of the `~` formula operator to indicate **functional dependence**:

```
> m.lzn.ff <- lm(zinc.l ~ ffreq, data = meuse)
```

Note that here we can write the functional dependence as `zinc.l ~ ffreq`, i.e., just the field names, without the data frame name, by using the optional `data` argument to name the dataframe where the `lm` function should look for those names.

Note also that the `<-` assignment operator saved the results of the modelling in a workspace object.

Task 13 : List the workspace to see this model object; compare the types of the two object. •

The `class` function shows the type of object:

```
> ls()

[1] "m.lzn.ff" "meuse"

> class(meuse)

[1] "data.frame"

> class(m.lzn.ff)

[1] "lm"
```

This shows that R can store different kinds of objects in the workspace. Each object has a **class**, so that methods can be used appropriately. Here is an example:

Task 14 : View the model summary. •

The `summary` method can be applied to model objects (above we applied it to a vector); because of the class system it produces an appropriate summary.

```
> summary(m.lzn.ff)

Call:
lm(formula = zinc.l ~ ffreq, data = meuse)

Residuals:
    Min      1Q  Median      3Q     Max
-0.6242 -0.2233 -0.0176  0.2017  0.5648

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.6997     0.0298   90.73  < 2e-16 ***
ffreq2       -0.3319     0.0493   -6.73  3.3e-10 ***
ffreq3       -0.2750     0.0642   -4.28  3.2e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.273 on 152 degrees of freedom
Multiple R-squared: 0.253,        Adjusted R-squared: 0.243
F-statistic: 25.8 on 2 and 152 DF,  p-value: 2.34e-10
```

Q12 : *How much of the total variation in metal concentration is explained by the flooding frequency?* *Jump to A12* •

Q13 : *What is the modelled mean log concentration of the metal in each class?* *Jump to A13* •

Clearly, this prediction is not so good. So, we turn to **spatial** analysis.

6 Local spatial structure

We now consider the **coordinates** of each point; this allows us to either look for a regional **trend** or a **local structure**. In this study area we don't expect a trend, as we might with e.g.. aquifer depth in a tilted bed. So we concentrate on local structure.

Task 15 : Make a **spatially-explicit** object. •

The coordinates are now just fields in the dataframe; we should give them special status – they are *not* attributes in the same sense as the soil properties or covariables. The `sp` "spatial objects" package provides classes for spatially-explicit data; we just need to tell it which fields represent the coordinates. Again, the `class` function shows the class name of an object.

```
> class(meuse)

[1] "data.frame"

> coordinates(meuse) <- c("x", "y")
> class(meuse)

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

Note the use of the `c` "catenate" (meaning "make a chain") function to create a **list** of two names, here `"x"` and `"y"`; the `coordinates` method expects such a list, to know which fields represent the coordinates.

The class has changed, from `data.frame` to `SpatialPointsDataFrame`.

Task 16 : Display the structure of the spatially-explicit object. •

```
> str(meuse)

Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
  ..@ data       :'data.frame':        155 obs. of  16 variables:
  .. ..$ cadmium  : num [1:155] 11.7 8.6 6.5 2.6 2.8 3 3.2 2.8 2.4 1.6 ...
  .. ..$ copper   : num [1:155] 85 81 68 81 48 61 31 29 37 24 ...
  .. ..$ lead     : num [1:155] 299 277 199 116 117 137 132 150 133 80 ...
  .. ..$ zinc     : num [1:155] 1022 1141 640 257 269 ...
```

```
.. ..$ elev     : num [1:155] 7.91 6.98 7.8 7.66 7.48 ...
.. ..$ dist     : num [1:155] 0.00136 0.01222 0.10303 0.19009 0.27709 ...
.. ..$ om       : num [1:155] 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 ...
.. ..$ soil     : Factor w/ 3 levels "1","2","3": 1 1 1 2 2 2 2 1 1 2 ...
.. ..$ lime     : Factor w/ 2 levels "0","1": 2 2 2 1 1 1 1 1 1 1 ...
.. ..$ landuse  : Factor w/ 15 levels "Aa","Ab","Ag",..: 4 4 4 11 4 11 4 2 2 15
.. ..$ dist.m   : num [1:155] 50 30 150 270 380 470 240 120 240 420 ...
.. ..$ zinc.l   : num [1:155] 3.01 3.06 2.81 2.41 2.43 ...
.. ..$ cadmium.l: num [1:155] 1.068 0.934 0.813 0.415 0.447 ...
.. ..$ copper.l : num [1:155] 1.93 1.91 1.83 1.91 1.68 ...
.. ..$ lead.l   : num [1:155] 2.48 2.44 2.3 2.06 2.07 ...
..@ 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
```

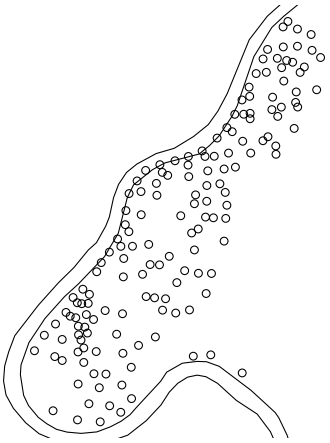
The notation `@` in the structure refers to sets of object properties, called **slots**.

Q14 : *Which slots in the object refer to the spatial structure and which to the attributes?* *Jump to A14* •

> **Note:** The `@proj4string` slot has no projection information, so we don't know what the coordinates represent. That information can be added, but for now we just consider them as meters on an arbitrary grid.

Task 17 : Display a map of the sample locations, along with the line of the Meuse river. •

```
> plot(meuse, asp = 1, pch = 1)
> data(meuse.riv)
> lines(meuse.riv)
```



Note the use of the optional `pch` argument to specify the plotting character.

Q15 : *How are the points distributed over the study area? For example, are they evenly-spaced (gridded)? Random? Denser in some areas?* *Jump to A15* •

Task 18 : Display a **postplot** of the untransformed Zn values, that is, plot the sample locations (as above) and represent the data value by the size of the symbol. •

We use the `cex` optional argument to specify the symbol size; this is computed as a proportion of the maximum.

```
> plot(meuse, asp = 1, cex = 4 * meuse$zinc/max(meuse$zinc),
+     pch = 1)
> lines(meuse.riv)
```

Q16 : *Do nearby observations seem to more similar to each other than would be expected at random?* *Jump to A16* •

Now we investigate the idea of **local spatial dependence**: "closer in geograhic space implies closer in attribute space". This may be true or not; and if true, the **range** of dependence will vary, depending on the physical process that produced the attribute being investigated.

The fundamental concept is **spatial auto-correlation**: an attribute value can be correlated *to itself*, with the strength of the correlation depending on **separation distance** (and possibly direction).

This should be evident as a relation between separation distance and correlation; the latter is often expressed as the **semi-variance**.

Each pair of observation points has a **semi-variance**, usually represented as the Greek letter γ ('gamma'), and defined as:

$$\gamma(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{2}[z(\mathbf{x}_i) - z(\mathbf{x}_j)]^2$$

where $\mathbf{x}$ is a geographic point and $z(\mathbf{x})$ is its **attribute value**.

Task 19 : Compute the number of point-pairs. •

The `length` function returns the **length** of a vector, i.e. the number of observations of an attribute. There are $(n \times (n-1))/2$ **pairs** of these:

```
> n <- length(meuse$zinc.l)
> n * (n - 1)/2

[1] 11935
```

Q17 : *How many unique* pairs *of points are there in this dataset?* *Jump to A17* •

Task 20 : Compute the distance and semivariance between the first two points in the data set. •

We follow the formula for the semivariance, and use Euclidean distances. Individual coordinates (E or N) in the two-element vector (consisting of both the E and N coordinates) are selected by the `[]` matrix selection operator.

```
> (gamma <- 0.5 * (meuse$zinc.l[1] - meuse$zinc.l[2])^2)

[1] 0.0011441

> dim(coordinates(meuse))

[1] 155   2

> head(coordinates(meuse)[, 1])

[1] 181072 181025 181165 181298 181307 181390

> head(coordinates(meuse)[, 2])

[1] 333611 333558 333537 333484 333330 333260

> coordinates(meuse)[1, ]

     x      y
181072 333611

> coordinates(meuse)[2, ]

     x      y
181025 333558

> coordinates(meuse)[1, 1]

     x
181072

> coordinates(meuse)[1, 2]

     y
333611

> (sep <- sqrt((coordinates(meuse)[1, 1] - coordinates(meuse)[2,
+     1])^2 + (coordinates(meuse)[1, 2] - coordinates(meuse)[2,
+     2])^2))

     x
70.838
```

Q18 : *What is the separation and semivariance for this point-pair?* *Jump to A18* •

! → Now, the question is, how are the semivariances related to the separations, *on average*? The **theory of stationary random fields** is based on the assumption that **absolute location** is not important; only **relative location**, i.e., the **same local spatial structure** is found anywhere in the study area.

The tool for investigating this is the **empirical variogram**, defined as the average semivariance within some separation range:

$$\overline{\gamma}(\mathbf{h}) = \frac{1}{2m(\mathbf{h})} \sum_{(i,j)|\mathbf{h}_{ij}=\mathbf{h}} (z(\mathbf{x}_i) - z(\mathbf{x}_j))^2$$

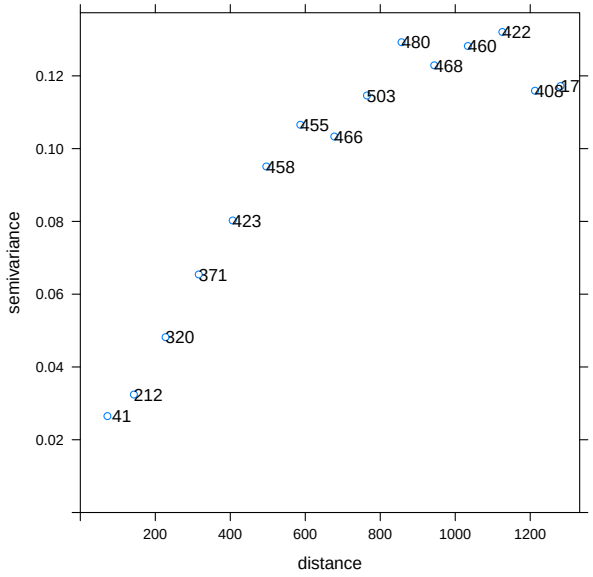
Here we consider **points** numbered $1, 2, \ldots, i, \ldots, j, \ldots n$, i.e. the list of points, out of which we make **point-pairs**.

- $\mathbf{h}_{ij}$ is the distance between points i and j
- the notation $(i, j)|$ reads "pairs of points indexed as i and j, *such that* ...
- $= \mathbf{h}$ means that this point-pair has the separation vector.
- In practice $\mathbf{h}$ is some small range, the **bin**.

Task 21 : Plot the experimental variogram of the log-Zn concentrations, i.e. the average semi-variances of the point-pairs versus average distance (lag), with a bin width of 90 m, to a maximum lag distance of 1300 m. •

The `variogram` function computes the experimental variogram:

```
> (v <- variogram(zinc.l ~ 1, meuse, cutoff = 1300,
+     width = 90))

    np     dist    gamma dir.hor dir.ver   id
1   41   72.248 0.026500       0       0 var1
2  212  142.880 0.032424       0       0 var1
3  320  227.322 0.048189       0       0 var1
4  371  315.855 0.065431       0       0 var1
5  423  406.448 0.080259       0       0 var1
6  458  496.094 0.095098       0       0 var1
7  455  586.786 0.106566       0       0 var1
8  466  677.396 0.103335       0       0 var1
9  503  764.557 0.114613       0       0 var1
10 480  856.694 0.129244       0       0 var1
11 468  944.029 0.122901       0       0 var1
12 460 1033.623 0.128203       0       0 var1
13 422 1125.632 0.132065       0       0 var1
14 408 1212.623 0.115913       0       0 var1
15 173 1280.654 0.117200       0       0 var1

> print(plot(v, plot.numbers = T))
```



The `plot.numbers` optional argument is used to display the number of point-pairs in each bin; this aids interpretation of the variogram, because bins with more point-pairs are more reliable (based on a larger proportion of the sample).

The formula `zinc.l ~ 1` specifies the dependence of the left-hand side "dependent" variable, here `zinc.l`, on the right-hand side "independent" variable(s), here just `1`. As usual, the `~` formula operator is used to separate the dependent and independent variables. The `1` here represents the spatial mean of the dependent variable – that is, the variable `zinc.l` is only dependent on itself! This is why we use the term spatial **auto-** ("self") correlation.

Q19 : *What is the **average separation** and **average semivariance** in the first bin? How many **point-pairs** are averaged for this?* *Jump to A19* •

Q20 : *What evidence does this plot give that closer points are more similar, i.e., what is the evidence for **local spatial dependence**?* *Jump to A20* •

Q21 : *At what separation between point-pairs is there no more spatial dependence? (This is called the **range**)* *Jump to A21* •

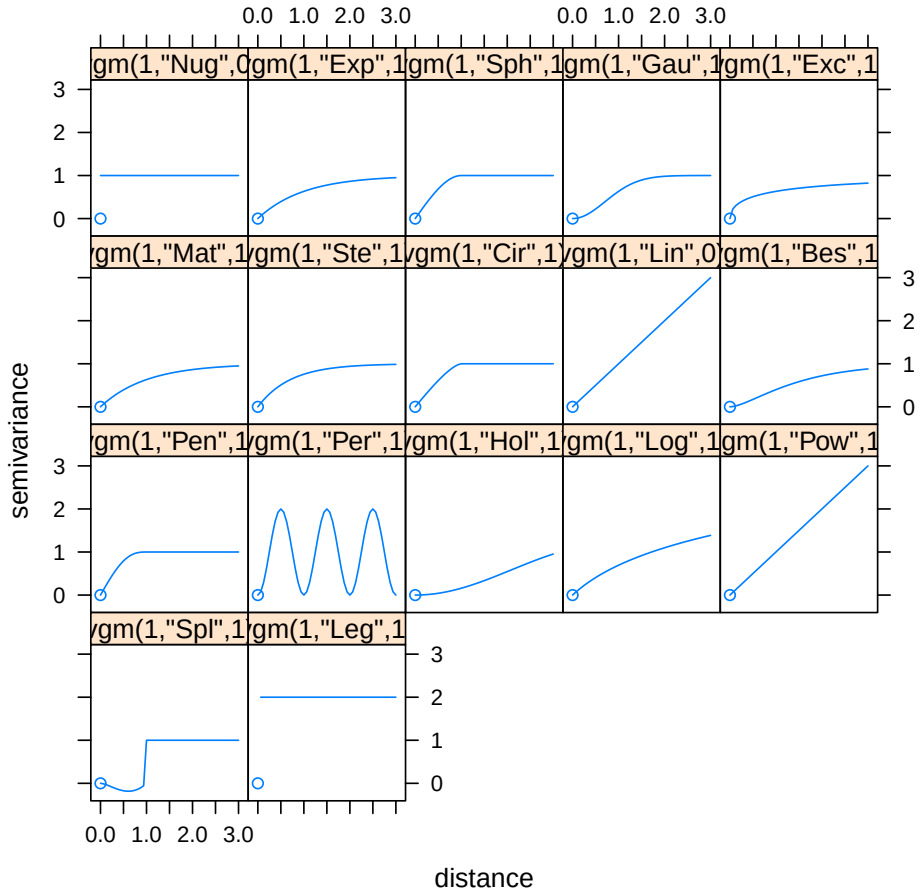
Q22 : *What is the semivariance at zero separation? (This is called the **nugget**). Why is it not zero?* *Jump to A22* •

Q23 : *What is the semivariance at the range and beyond? (This is called*

*the **total sill**)* *Jump to A23* •

We **summarize** the spatial dependence with a **variogram model**. This is a **continuous function** of semi-variance on separation. There are many model forms:

```
> print(show.vgms())
```



Selecting a model form is an art; the best way is from knowledge of the assumed spatial process. From many studies we know that a common model for soil attributes is the **spherical** model:

$$\gamma(h) = \begin{cases} c \cdot \left[\frac{3}{2}\frac{h}{a} - \frac{1}{2}\left(\frac{h}{a}\right)^3\right] & : \quad h < a \\ c & : \quad h \geq a \end{cases}$$

This has two **parameters** which must be fit to the empirical variogram:

- a: the range; i.e., separation at which there is no more spatial dependence.
- c: the sill; i.e., maximum semivariance.

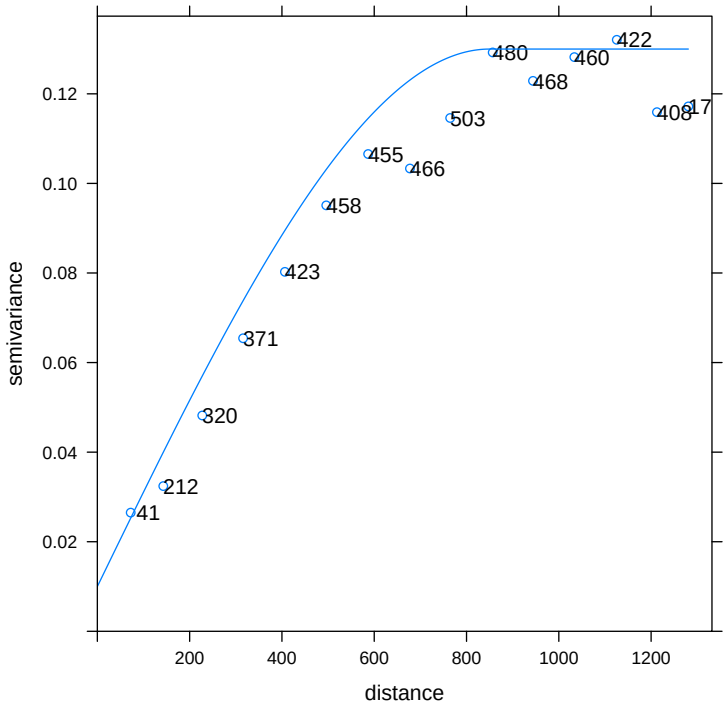
In addition, the whole variogram is raised by a **nugget** variance.

> **Note:** A second method to select a variogram model form is a visual assessment of the empirical variogram's shape, comparing to the various authorized models; a third method is to compare goodness-of-fit of a fitted model to the empirical variogram. Visual assessment is subjective and requires considerable experience; goodness-of-fit can be used when the process must be automated and as a data-mining technique.

Task 22 : Fit a spherical variogram model by eye to the experimental semivariogram and plot it; then adjust it with `gstat` automatic fit (`fit.variogram` function). •

The `vgm` function specifies a variogram model. In the previous set of questions we estimated the these parameters from looking at the empirical variogram, so we supply them as the model parameters. Note that the `psill` "partial sill" model parameter is the total sill (which we estimated as 0.13), less the nugget variance (which we estimated as 0.01), i.e., 0.12:

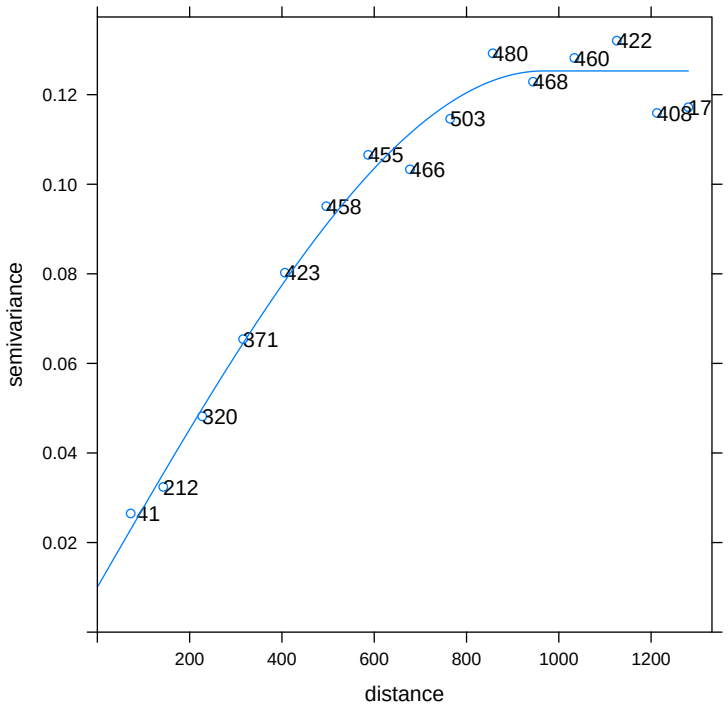
```
> vm <- vgm(psill = 0.12, model = "Sph", range = 850,
+     nugget = 0.01)
> print(plot(v, pl = T, model = vm))
```



We can see our original estimate does not fit the empirical variogram very well; we could adjust this by eye but when a variogram has a regular form (as this one does), the `fit.variogram` function will adjust it nicely:

```
> (vmf <- fit.variogram(v, vm))
```

```
  model    psill  range
1   Nug 0.010041   0.00
2   Sph 0.115257 967.26

> print(plot(v, pl = T, model = vmf))
```



Q24 : *What are the nugget, total sill, and range of this model, as adjusted by the automatic fit? How do these compare to our initial estimates?* *Jump to A24* •

This is an excellent model of spatial dependence, and gives insight into the physical process.

7 Mapping by interpolation

We now use the spatial structure to "optimally" interpolate to un-sampled points. There are many ways to interpolate; we will first investigate **Ordinary Kriging**.

7.1 Theory of Ordinary Kriging

The theory of regionalised variables leads to an **"optimal"** prediction method, in the sense that the **kriging variance** is **minimized**.

Of course, there are many other local interpolators, but they all have problems:

- Problems with **Thiessen polygons**:

1. Abrupt changes at boundaries are an artifact of the sample spatial distribution;
2. Only uses one sample point for each prediction; inefficient use of information.

- Problems with **average-in-circle** methods:
  1. There is no objective way to select radius of circle or number of points;
  2. Obviously false underlying assumption.
- Problems with **inverse-distance** methods:
  1. There is no objective way to choose the power (inverse, inverse squared ...);
  2. There is no objective way to choose the limiting radius.
- In all cases:
  1. **Uneven distribution of samples**: over– or under–emphasize some areas.
  2. The **kriging variance** must be estimated from a separate validation dataset.

These deficiencies in existing local interpolations were well-known. The aim was to develop a **linear predictor** as a **weighted average** of the observations, with an objectively **optimal** method of assigning the weights.

The theory for this developed several times but current practise dates back to Matheron (1963), formalizing the practical work of the mining engineer D G **Krige** (RSA). In his honour these methods are called **kriging** (now with a small "k");

What is so special about kriging?

- Predicts at any point as the **weighted average** of the values at sampled points
  - as for inverse distance (to a power)
- Weights given to each sample point are **optimal**, given the **spatial covariance structure** as revealed by the **variogram model** (in this sense it is "best")
  - Spatial structure **between known points**, as well as **between known points and each prediction point**, is accounted for.
  - So, the prediction is only as good as the model of spatial structure.
- The **kriging variance** at each point is automatically generated as part of the process of computing the weights.

In case you are interested, here is the Ordinary Kriging system, in matrix form:

$$\mathbf{A}\lambda = \mathbf{b} \qquad \mathbf{A} = \begin{bmatrix} \gamma(\mathbf{x}_1,\mathbf{x}_1) & \gamma(\mathbf{x}_1,\mathbf{x}_2) & \cdots & \gamma(\mathbf{x}_1,\mathbf{x}_N) & 1 \\ \gamma(\mathbf{x}_2,\mathbf{x}_1) & \gamma(\mathbf{x}_2,\mathbf{x}_2) & \cdots & \gamma(\mathbf{x}_2,\mathbf{x}_N) & 1 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \gamma(\mathbf{x}_N,\mathbf{x}_1) & \gamma(\mathbf{x}_N,\mathbf{x}_2) & \cdots & \gamma(\mathbf{x}_N,\mathbf{x}_N) & 1 \\ 1 & 1 & \cdots & 1 & 0 \end{bmatrix}$$

$$\lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_N \\ \psi \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} \gamma(\mathbf{x}_1,\mathbf{x}_0) \\ \gamma(\mathbf{x}_2,\mathbf{x}_0) \\ \vdots \\ \gamma(\mathbf{x}_N,\mathbf{x}_0) \\ 1 \end{bmatrix}$$

This system shows that the semivariance γ must be known for the prediction point and all observation points (this is the **b** vector) and also between all pairs of known points (this is the **A** matrix); this is why a **variogram function** is needed, to know the semivariance at any separation.

This is a system of $N + 1$ equations in $N + 1$ unknowns, so can be solved uniquely, as long as **A** is positive definite; this is guaranteed by using authorized models. This has the solution (in matrix notation):

$$\lambda = \mathbf{A}^{-1}\mathbf{b}$$

Now we can **predict** at the point, using the weights:

$$\hat{z}(\mathbf{x}_0) = \sum_{i=1}^{N} \lambda_i z(\mathbf{x}_i)$$

The **kriging variance** at a point is given by the scalar product of the weights (and multiplier) vector λ with the right-hand side of the kriging system: Note that λ includes as its last element ψ, which depends on covariance structure of the sample points:

$$\hat{\sigma}^2(\mathbf{x}_0) = \mathbf{b}^T\lambda$$

! → Before going further, we must emphasize: **the "optimality" of the kriging prediction depends on a correct model of spatial variability**, i.e., the variogram model should reflect the spatial process that gave rise to the attribute being mapped by interpolation. Thus, the variogram modelling of the previous section must be carried out correctly. There is no objective way to do this! If there are not enough points (at least 100 to 150) with a good distribution of separations, it is not possible to model a variogram, and kriging should *not* be used.

7.2 Ordinary kriging on a regular grid

The most common use of kriging is to predict at the nodes of a **regular grid** which covers an area of interest.

For this sample dataset, a regular grid has been prepared, named `meuse.grid`.

Task 23 : Load the 40 m x 40 m interpolation grid covering the sample area and convert it to a spatial object. •

As before, the `data` function loads a built-in dataset, and the `coordinates` method assigns coordinates. The `gridded` method specifies that the data is on a regular grid; this allows more efficient storage and computation than for isolated points.

```
> data(meuse.grid)
> coordinates(meuse.grid) <- c("x", "y")
> gridded(meuse.grid) <- T
```

Task 24 : Predict the attribute value at all grid points using Ordinary Kriging. •

The `krige` method performs many forms of kriging; by specifying that a variable is only dependent on itself (i.e., the right-hand side of the formula only specifies the intercept, symbolically `~1`) the spatial mean is calculated, and there is no trend: this is Ordinary Kriging

> **Note:** The "ordinary" means (1) the variable is only modelled from itself, with no other information such as a geographic trend or other variable; (2) the spatial mean is not known *a priori* and must be estimated from the data.

```
> k40 <- krige(zinc.l ~ 1, locations = meuse, newdata = meuse.grid,
+     model = vmf)

[using ordinary kriging]
```

As with the empirical variogram plot (§6) and feature-space modelling (S5), the `~` formula operator is used to separate the dependent and independent variables.

Note the use of the **named arguments** `locations`, `newdata`, and `model`; these are explained in the help for this command; if you wish you can view it with `?krige`.

Task 25 : Display the structure of the kriging prediction object. •

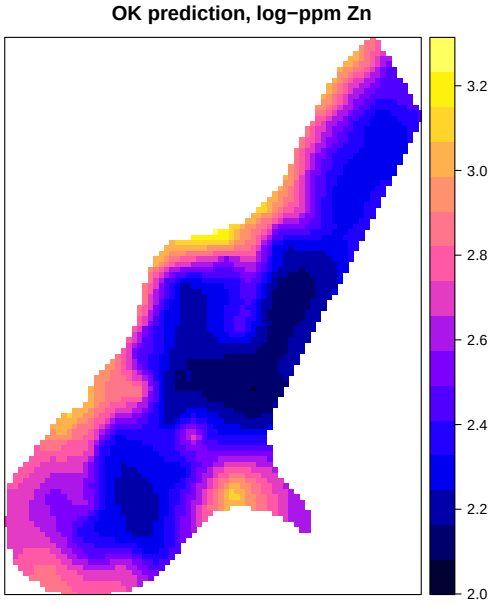
```
> str(k40)

Formal class 'SpatialPixelsDataFrame' [package "sp"] with 7 slots
  ..@ data       :'data.frame':        3103 obs. of  2 variables:
  .. ..$ var1.pred: num [1:3103] 2.83 2.88 2.83 2.78 2.94 ...
  .. ..$ var1.var : num [1:3103] 0.0596 0.0471 0.0509 0.055 0.0335 ...
  ..@ coords.nrs : int [1:2] 1 2
  ..@ grid       :Formal class 'GridTopology' [package "sp"] with 3 slots
  .. .. ..@ cellcentre.offset: Named num [1:2] 178460 329620
  .. .. .. ..- attr(*, "names")= chr [1:2] "x" "y"
```

```
.. .. ..@ cellsize         : Named num [1:2] 40 40
.. .. .. ..- attr(*, "names")= chr [1:2] "x" "y"
.. .. ..@ cells.dim        : Named int [1:2] 78 104
.. .. .. ..- attr(*, "names")= chr [1:2] "x" "y"
..@ grid.index : int [1:3103] 69 146 147 148 223 224 225 226 300 301 ...
..@ 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
```

This is also a spatial object, of class `SpatialPixelsDataFrame`. Note that the `@data` slot has two fields: the **prediction** (field `var1.pred`) and the **prediction variance** (field `var1.var`).

Task 26 : Display the map of predicted values. •

```
> print(spplot(k40, "var1.pred", asp=1, col.regions=bpy.colors(64),
+              main="OK prediction, log-ppm Zn"))
```



The `spplot` "spatial plot" method displays spatial objects:

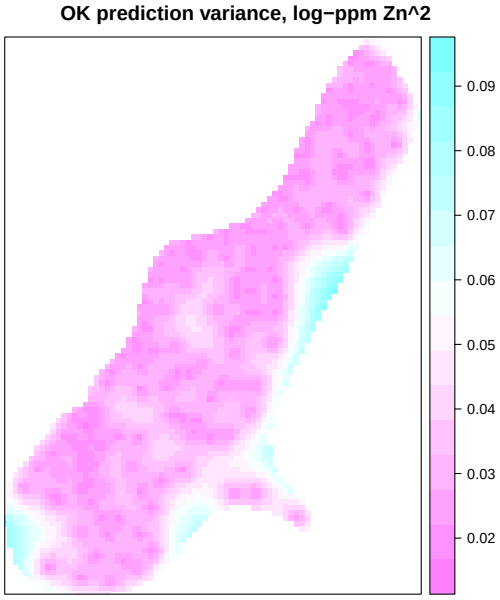
Note the use of the `bpy.colors` function to use a colour ramp; the result of this function is a vector of colours, used as the values of the `col.regions` optional argument of `spplot`.

Q25 : *Describe the kriged map with respect to its (1) smoothness, (2)*

values at observed data points. *Jump to A25* •

Task 27 : Display the map of kriging prediction variances. •

```
> print(spplot(k40, "var1.var", asp=1,
+              main="OK prediction variance, log-ppm Zn^2"))
```



Q26 : *Describe the variances map. Where is the prediction variance lowest? Does this depend on the data value?* *Jump to A26* •

8 Taking a break

This concludes the **minimal** introduction to the R environment and point geostatistics. The material that follows shows some interesting extensions. Most likely you need to take a break (maybe a long one!). Before stopping:

Task 28 : Save the workspace. •

The `save.image` function saves all objects in your workspace to a file in the current working directory. It is conventional to use the `.RData` file extension to R data files.

```
> save.image(file = "minimal_geostat.RData")
```

Now if you wish you can exit R with the `q` "quit" function, or you can use the normal way to leave a program.

```
> q()
```

You will be asked if you wish to save the workspace; if you do so the `save.image` function is called with filename `.Rdata` (i.e., only an extension, no file name).

9 Loading a workspace

If you want still more geostatistics, here we go!

If you **did not** take a break, skip the rest of this section, go directly to §10.

If you **did** take a break:

Task 29 : Start R, and load your saved workspace. •

If you answered "yes" to the query "Save workspace?" when you took a break, and you start R in the same **working directory**, the workspace in `.RData` will be restored.

Otherwise you can load the image you saved:

```
> load(file = "minimal_geostat.RData")
```

You should now see all the objects from your last session:

```
> ls()

 [1] "gamma"      "k40"        "m.lzn.ff"   "meuse"
 [5] "meuse.grid" "meuse.riv"  "n"          "sep"
 [9] "v"          "vm"         "vmf"
```

However, R does *not* automatically reload add-in packages, so you have to again load `gstat` (which then loads `sp`):

```
> require(gstat)
```

10 Non-parameteric methods: Indicator kriging

In some situations we are not interested in the actual value of some attribute, but rather the **probability** that it **exceeds some threshold**. Soil pollution is an excellent example: we want to find the probability that a site exceeds a regulatory threshold. Mining is another example: we want to find the probability that a site exceeds some economic threshold.

One way to approach this is by converting continuous variables to **indicators**: a True/False (or, 0/1) value that "indicates" whether an observation is below (1, True) or above (0, False) some threshold.

According to the Berlin Digital Environmental Atlas[3], the *critical level* for Zn is 150 mg kg^{-1}; crops to be eaten by humans or animals should not be grown in these conditions.

Task 30 : Convert the observations for Zn to an indicator, and add to the data frame; summarize them. •

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

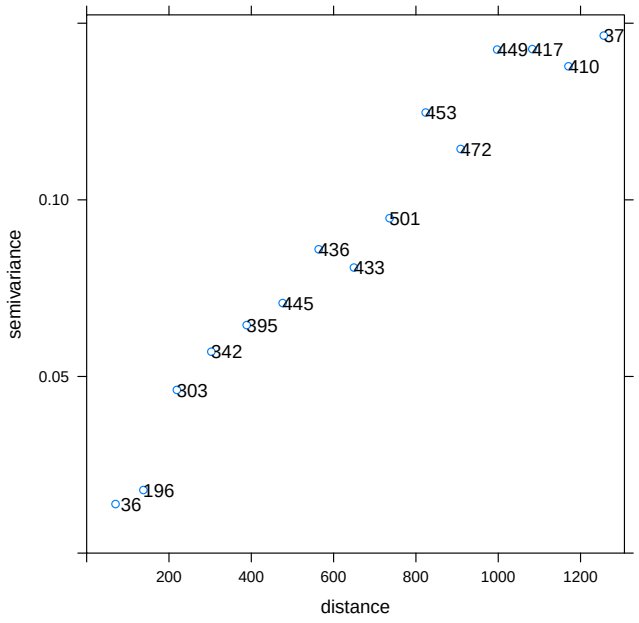
We use a **logical expression** which is either True or False for each element of the data vector, and assign the result of the expression, i.e., a **logical vector**, to a new field in the dataframe:

```
> meuse$zn.i <- (meuse$zinc < 150)
> summary(meuse$zn.i)

   Mode    FALSE     TRUE     NA's
logical      140       15        0
```

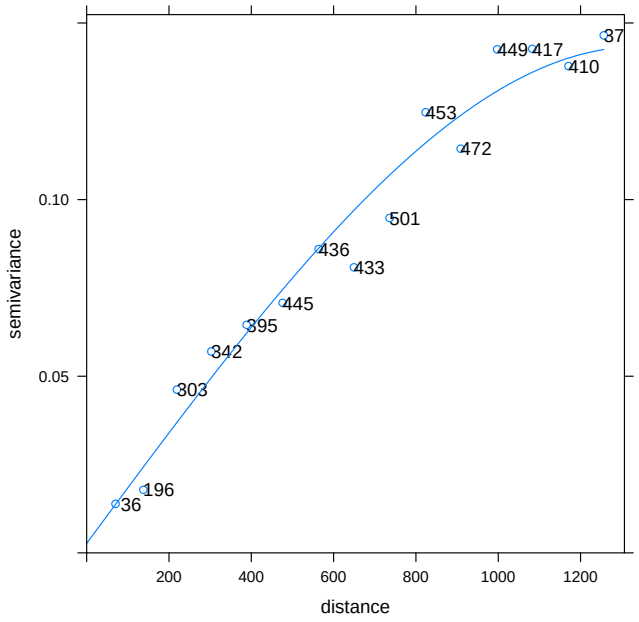
Q27 : *How many observations are above the threshold?* *Jump to A27* •

Task 31 : Make an empirical **indicator variogram** and model it. •

```
> vi <- variogram(zn.i ~ 1, location = meuse, cutoff = 1300)
> print(plot(vi, pl = T))
```




```
> (vimf <- fit.variogram(vi, vgm(0.12, "Sph", 1300,
+     0)))

  model    psill  range
1   Nug 0.002684    0.0
2   Sph 0.140557 1338.3

> print(plot(vi, pl = T, model = vimf))
```



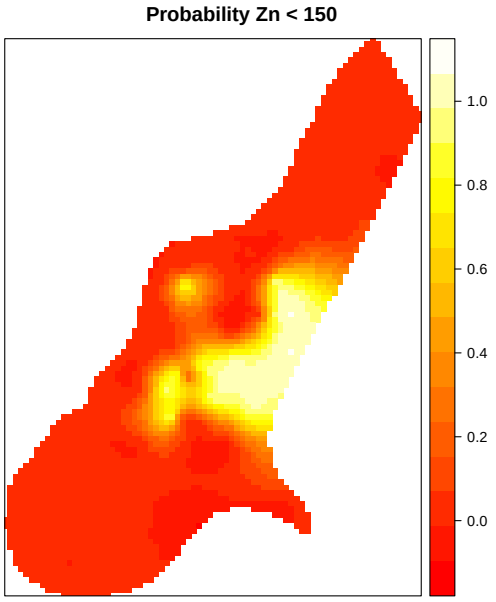
Q28 : *What is the range of the indicator? Does this match that for the original variable?* *Jump to A28* •

Q29 : *What is the sill? What are its units?* *Jump to A29* •

Task 32 : Intepolate over the prediction grid, using indicator kriging; display the prediction, •

Again we krige with `krige` and plot with `spplot`:

```
> k40.i <- krige(zn.i ~ 1, loc = meuse, newdata = meuse.grid,
+     model = vimf)

[using ordinary kriging]

> print(spplot(k40.i, zcol = "var1.pred", col.regions = heat.colors(64),
+     asp = 1, main = "Probability Zn < 150"))
```



Note the use of the `heat.colors` function to use a colour ramp; the result of this function is a vector of colours, used as the values of the `col.regions` optional argument of `spplot`.

Q30 : *What parts of the study area are safe for agricultural land use?* *Jump to A30* •

11 Mixed predictors

In §5 we saw that the feature-space attribute "flooding frequency" explained about 25% of the variation in log-Zn concentration. Yet, we ignored this information when predicting by Ordinary Kriging (§7.2). In this section we examine a method to combine the two.

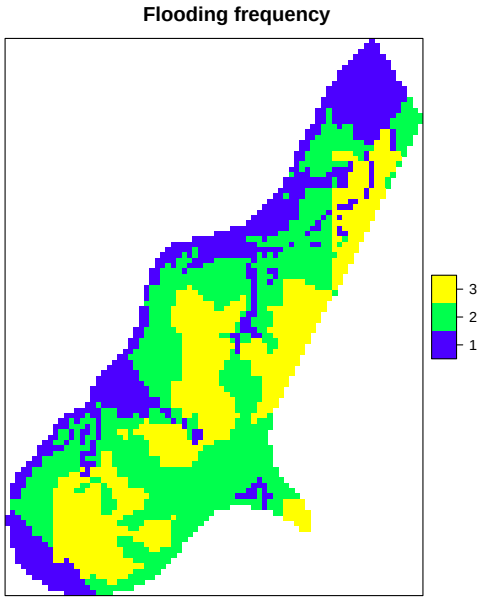
11.1 Feature-space prediction

In §5 we modelled metal concentration by a categorical variable, flood-frequency class. We can use this model to make a map by **reclassification**, i.e., each pixel is in one of the three flood-frequency classes, and we predict at that pixel by the mean value of metal concentration from the linear model.

To do this, we must know the value of the co-variable (here, flooding frequency class) at each prediction point; fortunately this is provided in the prediction grid (although its reliability is not known; still it can be used for illustration).

```
> summary(meuse.grid$ffreq)

   1    2    3
 779 1335  989
```

```
> print(spplot(meuse.grid, zcol = "ffreq", col.regions = topo.colors(3),
+     main = "Flooding frequency"))
```

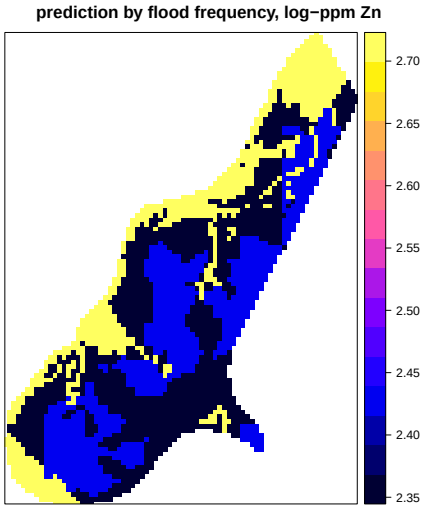


Here we use yet another colour ramp, using three colours from the `topo.colors` function.

Task 33 : Predict the metal concentration by flood-frequency class. •

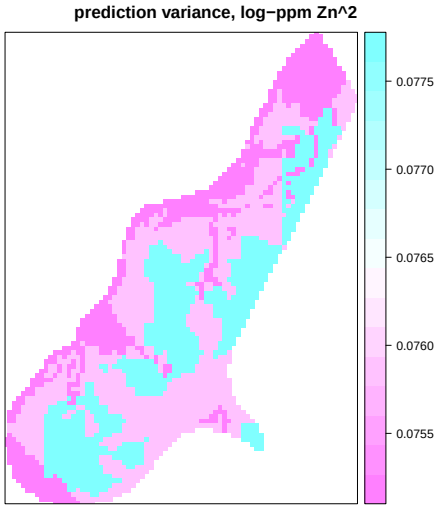
The `krige` method with the `model` argument set to `NULL` predicts without a model of spatial dependence, i.e., just from the feature-space model (here, metal predicted by flood-frequency class). The `krige` method computes the OLS regression exactly as does the `lm` function, and then uses that regression to fill the interpolation grid.

```
> k.ffreq <- krige(zinc.l ~ ffreq, locations=meuse,
+                  newdata=meuse.grid, model=NULL)

[ordinary or weighted least squares prediction]

> print(spplot(k.ffreq, zcol="var1.pred", col.regions=bpy.colors(64),
+              main="prediction by flood frequency, log-ppm Zn"))
```



And like any linear model, this also has a prediction variance:

```
> print(spplot(k.ffreq, zcol="var1.var",
+              main="prediction variance, log-ppm Zn^2"))
```



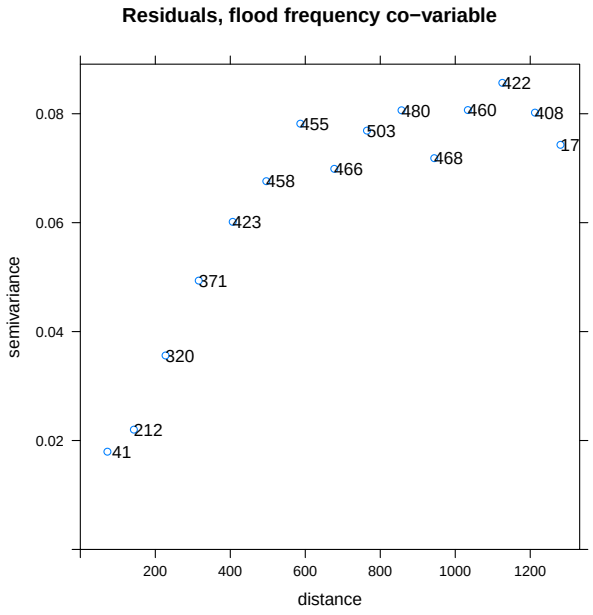
Q31 : *Explain the spatial pattern of this prediction and its variance.* *Jump to A31* •

11.2 The residual variogram

If some of the variation is explained by an attribute, it makes sense to remove that variation before looking for local spatial structure. A variogram where this has been done is called a **residual variogram**, and is specified by the functional dependence as in a linear model.

Task 34 : Compute and display the empirical residual variogram of log10-Zn, after accounting for flooding frequency. •

```
> (vr <- variogram(zinc.l ~ ffreq, location=meuse,
+                  cutoff=1300, width=90))

    np     dist    gamma dir.hor dir.ver   id
1   41   72.248 0.017949       0       0 var1
2  212  142.880 0.022000       0       0 var1
3  320  227.322 0.035616       0       0 var1
4  371  315.855 0.049363       0       0 var1
5  423  406.448 0.060152       0       0 var1
6  458  496.094 0.067618       0       0 var1
7  455  586.786 0.078177       0       0 var1
8  466  677.396 0.069887       0       0 var1
9  503  764.557 0.076906       0       0 var1
10 480  856.694 0.080635       0       0 var1
11 468  944.029 0.071843       0       0 var1
12 460 1033.623 0.080678       0       0 var1
13 422 1125.632 0.085676       0       0 var1
14 408 1212.623 0.080207       0       0 var1
15 173 1280.654 0.074295       0       0 var1

> print(plot(vr, plot.numbers=T,
+            main="Residuals, flood frequency co-variable"))
```



Q32 : *How does this empirical variogram compare to the original (non-residual) empirical variogram?* *Jump to A32*

•

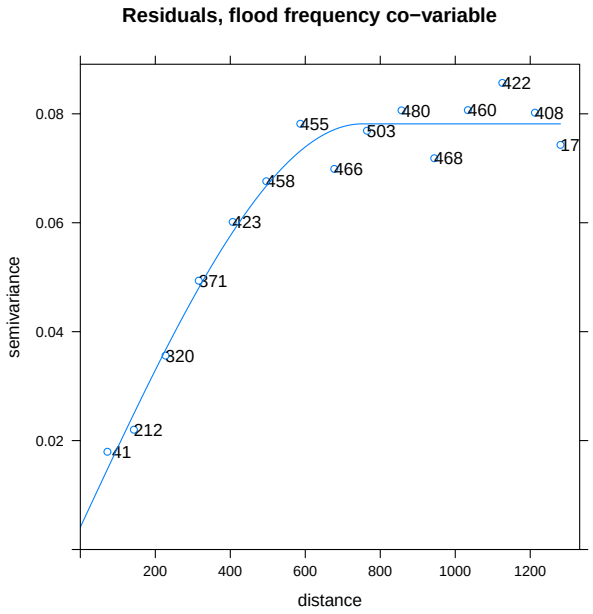
Clearly, accounting for the flood frequency has lowered the total variability (as expected from the results of the linear modelling), but it has also reduced the range of spatial dependence. Some of the apparent range in the original variogram was due to the spatial extent of the flooding classes; this has now been removed.

Task 35 : Model this variogram, first by eye and then with an automatic fit. Compare the model (partial sill, nugget, range) to the original variogram. •

```
> (vrmf <- fit.variogram(vr, vgm(psill = 0.08, model = "Sph",
+     range = 700, nugget = 0.01)))

  model     psill  range
1   Nug 0.0040945   0.00
2   Sph 0.0740612 752.11

> print(vmf)

  model    psill  range
1   Nug 0.010041   0.00
2   Sph 0.115257 967.26
```

There is no need to save the result of the `vgm` function in the workspace; the model object is immediately used as the second argument to the `fit.variogram` function.

```
> print(plot(vr, plot.numbers=T, main="Residuals, flood frequency co-variable",
+            model=vrmf))
```



Q33 : *How does this variogram model compare to the original (non-residual) variogram model?* *Jump to A33* •

The residual variogram clearly has a substantially lowered sill and reduced range. Also, the nugget variance has been halved; this is because several near-neighbour point pairs with different metal concentrations are in different flood frequency classes.

11.3 Prediction by Kriging with External Drift (KED)

The mixed predictor where some of the variation is from one or more attributes and some from local structure is often called **Kriging with External Drift** (KED), the "drift" being the value of the covariable. It is also sometimes called **Universal Kriging** (UK), although that term is reserved by many authors for prediction of a geographic trend plus local structure. They are mathematically equivalent.

Task 36 : Predict the attribute value at all grid points using KED on flood frequency. •

Now the prediction. We use the same `feature-space dependence` formula `zinc.l ~ffreq` as we used for the residual variogram. That is, the formula which was used to examine the spatial dependence must also be used for spatial prediction.

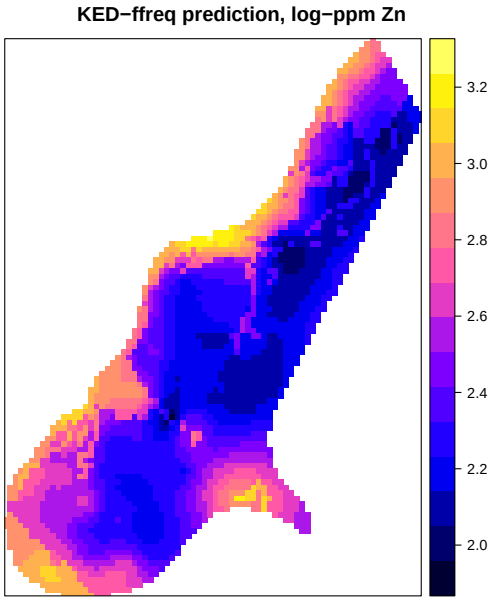
! → In KED, the formula for kriging *must* match that for the residual variogram.

```
> kr40 <- krige(zinc.l ~ ffreq, locations = meuse,
+     newdata = meuse.grid, model = vrmf)

[using universal kriging]
```

The `krige` method now reports `[using universal kriging]`; in the OK example it reports `[using ordinary kriging]`. The term **universal** kriging is used here; we prefer to call it KED.

Task 37 : Display the map of predicted values. •

```
> print(spplot(kr40, "var1.pred", asp=1, col.regions=bpy.colors(64),
+              main="KED-ffreq prediction, log-ppm Zn"))
```



Q34 : *How does this KED map compare to the OK map? Where is the effect of flood frequency class reflected in the prediction?* *Jump to A34* •

11.3.1 Displaying several maps on the same scale

To get a better idea of the differences in the predictions, we'd like to show the two maps side-by-side.

The `spplot` method returns a plotting object, which can be saved in the workspace rather than displayed immediately. The `plot` method of Lattice graphics can display several of these saved objects, using the `split` arguments.

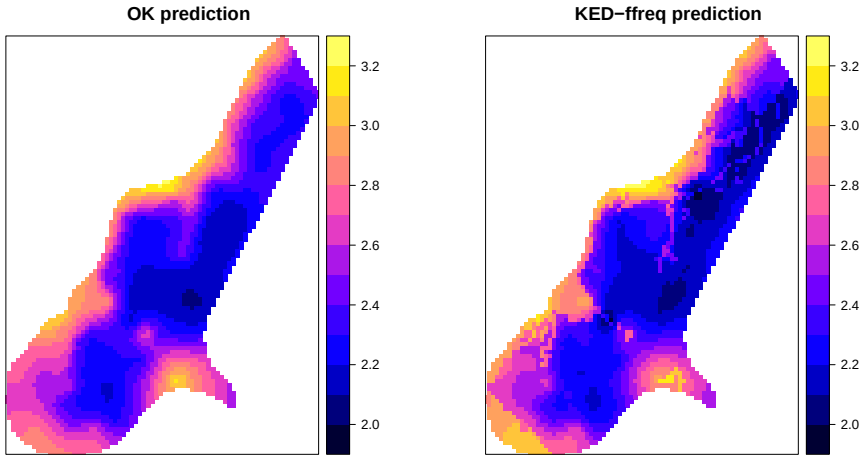
However, there is one important detail before we can do this – the scales of the two plots must be the same, for correct visual comparison. So, we detemine the overall maximum range and then use this for both the plots. The `max` and `min` functions find the extremes; we round these up and down to the next decimal. The `seq` function builds a list of breakpoints for the colour ramp.

```
> (zmax <- max(k40$var1.pred,kr40$var1.pred))

[1] 3.2373

> (zmin <- min(k40$var1.pred,kr40$var1.pred))

[1] 1.9566

> (zmax <- round(zmax, 1) + 0.1)

[1] 3.3

> (zmin <- round(zmin, 1) - 0.1)
```

```
[1] 1.9

> (ramp <- seq(from=zmin, to=zmax, by=.1))

 [1] 1.9 2.0 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 3.0 3.1 3.2 3.3

> p1 <- spplot(k40, "var1.pred", asp=1, main="OK prediction",
+              at=ramp, col.regions=bpy.colors(64))
> p2 <- spplot(kr40, "var1.pred", asp=1, main="KED-ffreq prediction",
+              at=ramp, col.regions=bpy.colors(64))
```

Now the two plots can be created, saved, and displayed in a grid:

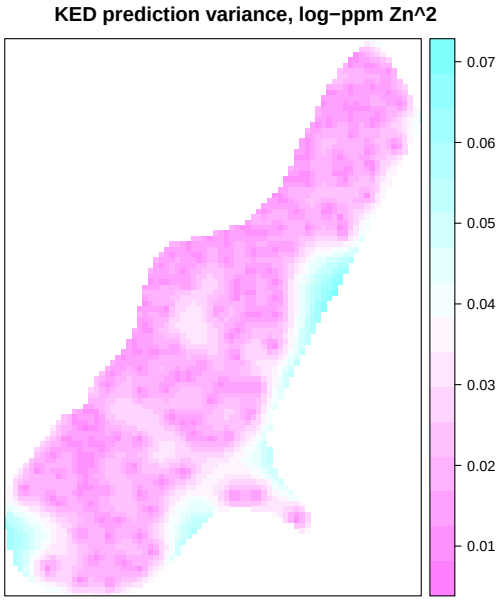
```
> plot(p1, split = c(1, 1, 2, 1), more = T)
> plot(p2, split = c(2, 1, 2, 1), more = F)
```



11.3.2 KED Prediction variances

Task 38 : Display the map of the KED prediction variances. •

```
> print(spplot(kr40, "var1.var", asp=1,
+              main="KED prediction variance, log-ppm Zn^2"))
```

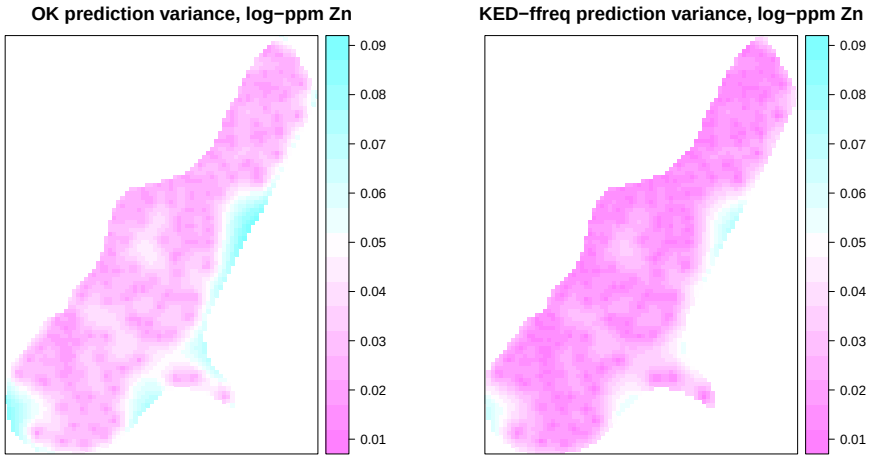


Task 39 : Compare these prediction variances to those for OK, both numerically and graphically. •

```
> summary(kr40$var1.var)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00807 0.01630 0.02050 0.02360 0.02810 0.06860

> summary(k40$var1.var)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0166  0.0260  0.0305  0.0347  0.0394  0.0923
```

We repeat the technique of standardizing the two scales; but here at the third decimal place:

```
> zmax <- round(max(k40$var1.var,kr40$var1.var), 3) + 0.001
> zmin <- round(min(k40$var1.var,kr40$var1.var), 3) - 0.001
> (ramp <- seq(from=zmin, to=zmax, by=.005))

 [1] 0.007 0.012 0.017 0.022 0.027 0.032 0.037 0.042 0.047 0.052
[11] 0.057 0.062 0.067 0.072 0.077 0.082 0.087 0.092

> p1 <- spplot(k40, "var1.var", asp=1, at=ramp,
+              main="OK prediction variance, log-ppm Zn")
> p2 <- spplot(kr40, "var1.var", asp=1, at=ramp,
+              main="KED-ffreq prediction variance, log-ppm Zn")

> plot(p1, split = c(1, 1, 2, 1), more = T)
> plot(p2, split = c(2, 1, 2, 1), more = F)
```



Clearly, KED has smoothed out the prediction variance, because within one flood-frequency class the prediction variance is the same everywhere. This helps especially at locations far from observation points.

11.4 A mutivariate mixed predictor

The feature-space model used for KED can include multiple predictors. However, recall that all covariables must be known across the grid, and of course also known at the observation points.

Task 40 : Determine which covariables are available for prediction. •

The `names` function lists the variable names in a data frame; the `intersect` function shows the intersection of two sets. Here, the two sets are the list of names of the observation dataframe and the grid dataframe.

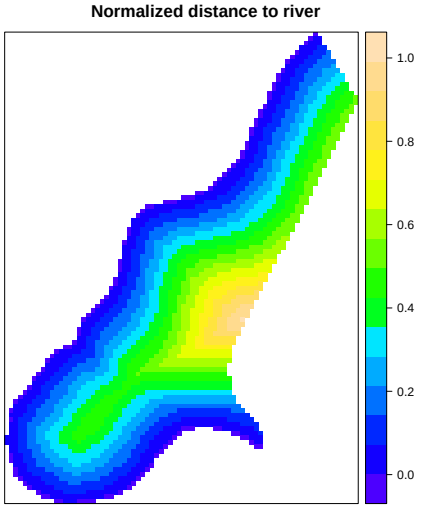
```
> names(meuse.grid)

[1] "part.a" "part.b" "dist"   "soil"   "ffreq"

> intersect(names(meuse), names(meuse.grid))

[1] "dist"  "ffreq" "soil"
```

Q35 : *What covariables are available?* *Jump to A35* •

We've already used `ffreq`. Looking at the documentation for the Meuse dataframe (§A or `help(meuse)`), we see that field `dist` is the normalized distance to the main channel. This seems promising for further refining the flood frequency: it may be that closer to the channel receives a heavier sediment load.

Task 41 : Display the normalized distance to river. •

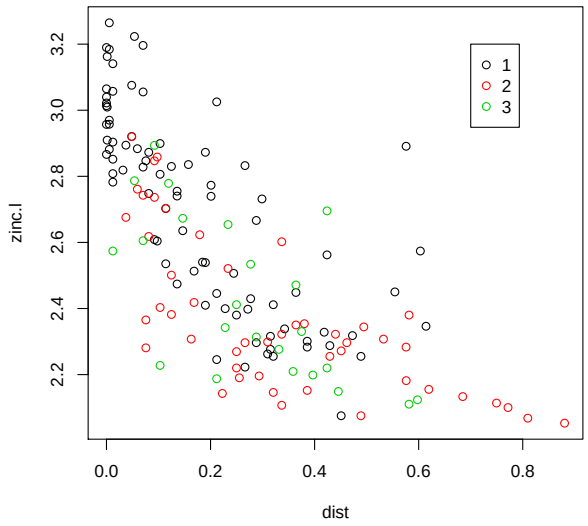
```
> print(spplot(meuse.grid, zcol="dist",
+              main="Normalized distance to river",
+              col.regions=topo.colors(64)))
```



In §5 we saw how to model the dependence of metal concentration on flood frequency, now we extend to a more complicated model. But first we start with a single continuous predictor.

Task 42 : Display a feature-space scatterplot of metal concentration vs. distance from river. •

```
> plot(zinc.l ~ dist, data = meuse@data, col = meuse$ffreq)
> legend(x = 0.7, y = 3.2, legend = c("1", "2", "3"),
+     pch = 1, col = 1:3)
```



Q36 : *Does there appear to be a linear relation between distance and metal concentration? How strong does it appear to be?* *Jump to A36* •

Task 43 : Model the dependence of metal concentration on distance to river •

Distance to river is a continuous variable; however the linear modelling and prediction follows the same procedure as in §11.1 for the classified predictor (flood frequency class).

```
> m.lzn.dist <- lm(zinc.l ~ dist, data = meuse)
> summary(m.lzn.dist)

Call:
lm(formula = zinc.l ~ dist, data = meuse)

Residuals:
    Min      1Q  Median      3Q     Max
-0.4889 -0.1583 -0.0007  0.1387  0.7286

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.8376     0.0268   105.9   <2e-16 ***
dist         -1.1726     0.0863   -13.6   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.212 on 153 degrees of freedom
Multiple R-squared: 0.547,        Adjusted R-squared: 0.544
F-statistic:  185 on 1 and 153 DF,  p-value: <2e-16
```

Q37 : *Which of the single-predictors models (flood-frequency class, distance to river) has the lowest residual sum-of-squares and highest adjusted R^2 (i.e., explains more of the variance)?* *Jump to A37* •

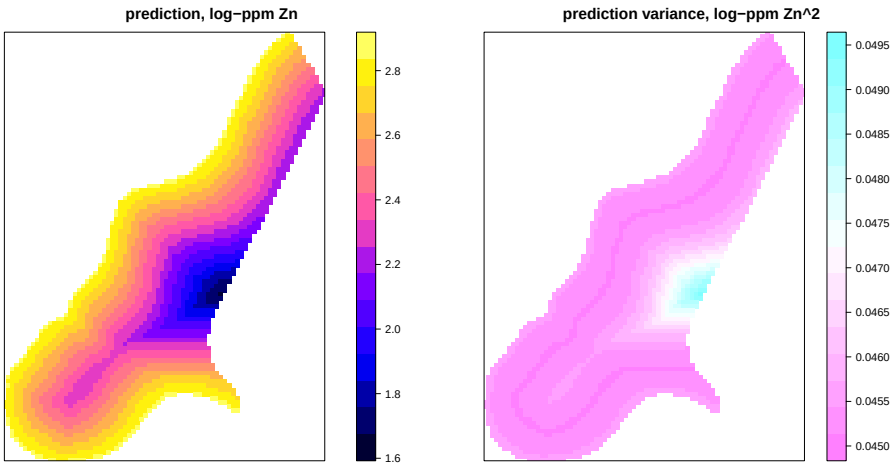
Task 44 : Predict the metal concentration over the study area, from the distance to river. •

As in §11.1 we use the `krige` method with the `model` argument set to `NULL` to predict from the linear model fit by ordinary least squares:

```
> k.dist <- krige(zinc.l ~ dist, locations=meuse,
+                 newdata=meuse.grid, model=NULL)

[ordinary or weighted least squares prediction]

> p1 <- spplot(k.dist, zcol="var1.pred", col.regions=bpy.colors(64),
+              main="prediction, log-ppm Zn")
> p2 <- spplot(k.dist, zcol="var1.var",
+              main="prediction variance, log-ppm Zn^2")
```

Note: The following code uses a feature of the `lattice` graphics package to ensure that the legends of the two maps are the same width. We do this by setting the `layout.widths` lattice graphics option with the `lattice.options` function.

```
> require(lattice)
> tmp <- lattice.options()
> lattice.options(layout.widths = list(key.right = list(x = 3,
+     units = "cm", data = NULL)))
> print(p1, split = c(1, 1, 2, 1), more = T)
> print(p2, split = c(2, 1, 2, 1), more = F)
> lattice.options(tmp)
```



Q38 : *Explain the spatial pattern of this prediction and its variance.* *Jump to A38* •

Task 45 : Model the dependence of metal concentration on distance to river combined with flood frequency, both as an additive effect and as an interaction. Compare the models, also to the previously-fit models of metal conconcentration based on flood frequency alone and distance to river alone. •

In the model formula for the `lm` function, two (or more) predictors are specified as **additive** effects with the `+` statistical formula operator; **interactive** effects with the `*` operator. When a set of linear models share some factors in a hierarchy, they can be compared by analysis of variance, using the `anova` function.

Recall, we computed the dependence of metal concentration on flood frequency as model `m.lzn.ff`, in §5; that model should still be in the workspace.

```
> m.lzn.ff.dist <- lm(zinc.l ~ ffreq + dist, data=meuse)
> m.lzn.ff.dist.i <- lm(zinc.l ~ ffreq * dist, data=meuse)
> anova(m.lzn.ff.dist.i, m.lzn.ff.dist, m.lzn.dist, m.lzn.ff)
```

```
Analysis of Variance Table

Model 1: zinc.l ~ ffreq * dist
Model 2: zinc.l ~ ffreq + dist
Model 3: zinc.l ~ dist
Model 4: zinc.l ~ ffreq
  Res.Df   RSS Df Sum of Sq     F  Pr(>F)
1    149  5.35
2    151  5.79 -2     -0.45  6.23  0.0025 **
3    153  6.86 -2     -1.07 14.89 1.3e-06 ***
4    152 11.31  1     -4.45
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The ANOVA table shows the degrees of freedom (lower as more predictors are added to the model), the residual sum-of-squares (how much of the variance is not explained by the model), and the probability that the reduction in sum-of-squares from a more complex model is due to chance.

Q39 : *Do the two-predictor models give significantly lower residual sum-of-squares?* *Jump to A39*

•

Q40 : *Does the interaction model give significantly lower residual sum-of-squares than the additive model?* *Jump to A40*

•

Another way to compare models is with an **information criterion** such as the AIC (Akaike's Information Criterion). The lower AIC indicates the lower entropy, i.e., a better model. The `AIC` function (surprise!) computes this:

Task 46 : Compare the AIC of the four models. •

```
> AIC(m.lzn.dist, m.lzn.ff, m.lzn.ff.dist, m.lzn.ff.dist.i)

                df     AIC
m.lzn.dist       3 -37.364
m.lzn.ff         4  42.062
m.lzn.ff.dist    5 -59.610
m.lzn.ff.dist.i  7 -68.050
```

Q41 : *Which model has the lowest AIC? Based on this and the ANOVA, which model gives the best feature-space prediction of metal concentration? What does this imply about the process?* *Jump to A41* •

Task 47 : Display the model summary for the best model. •

```
> summary(m.lzn.ff.dist.i)
```

```
Call:
lm(formula = zinc.l ~ ffreq * dist, data = meuse)

Residuals:
    Min      1Q  Median      3Q     Max
-0.4099 -0.1349 -0.0025  0.1080  0.7242

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.9398     0.0305   96.44  < 2e-16 ***
ffreq2       -0.3328     0.0573   -5.81  3.6e-08 ***
ffreq3       -0.2423     0.0849   -2.86  0.00491 **
dist         -1.3424     0.1253  -10.71  < 2e-16 ***
ffreq2:dist   0.6158     0.1747    3.52  0.00056 ***
ffreq3:dist   0.3596     0.2772    1.30  0.19657
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.189 on 149 degrees of freedom
Multiple R-squared: 0.647,        Adjusted R-squared: 0.635
F-statistic: 54.6 on 5 and 149 DF,  p-value: <2e-16
```

Q42 : *How much of the variability in metal concentration is explained by this model?* *Jump to A42* •

11.4.1 Linear model diagnostics

Recall that a linear model assumes the form:

$$z_i = \beta_0 + \sum_{j=1}^{k} \beta_i x_{ij} + \varepsilon_i$$

with $k + 1$ linear coefficients, where x_{ij} is the data value of variable j at observation i. A major assumption is that the residuals ε_i are **independently and identically distributed**, i.e., pure noise. If this assumption is violated, the linear model is not justified.

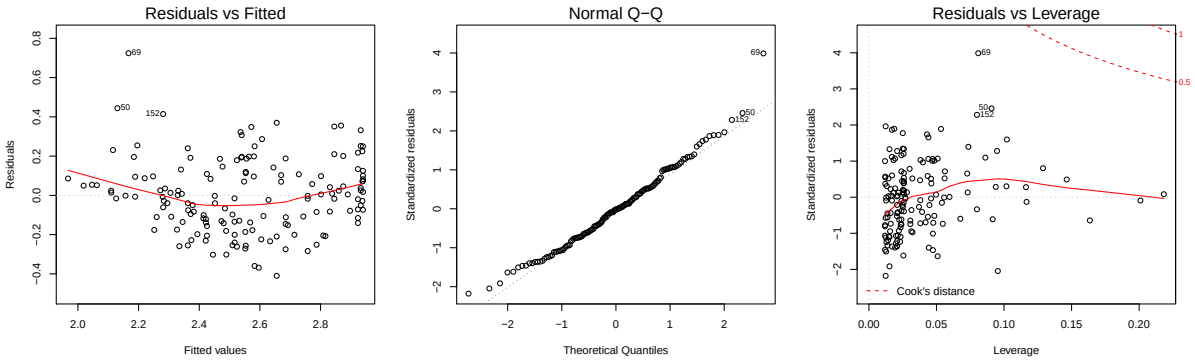
A linear model must satisfy several assumptions [4, 2], among which are:

1. no relation between predicted values and residuals (i.e., errors are independent);
2. normal distribution of residuals;
3. homoscedascity, i.e,. variance of residuals does not depend on the fitted value.

In addition, any high-influence observations ("high leverage") should not unduly influence the fit. The `influence.measures` function computes these; but it is easier to view these graphically, with the `plot` method, which specializes to the `plot.lm` function if it is called on objects of class `lm`. This

function produces six different plots; the most useful are `1` "Residuals vs. fitted values", `2` "Normal Q-Q", and `5` "Residuals vs. leverage"; see `?plot.lm` for more options.

```
> par(mfrow = c(1, 3))
> plot(m.lzn.ff.dist.i, which = c(1, 2, 5))
> par(mfrow = c(1, 1))
```



Q43 : *Looking at the "Residuals vs. fitted values" graph, do the residuals appear to be independent of the fitted values? Does the variance of the residuals appear to be the same throughout the range of fitted values?* *Jump to A43* •

Q44 : *Looking at the "Normal Q-Q" graph, do the residuals appear to be normally distributed?* *Jump to A44* •

Q45 : *Looking at the "Residuals vs. leverage" graph, do the high-leverage residuals have a high Cook's distance (a measure of how much the observation influences the model)?* *Jump to A45*
•

There are three poorly-modelled points (observations 50, 152, and especially 69) that are highlighted on all three graphs; these should be investigated to see if they are part of the population or the result of some unusual process.

The observation names, given by the `row.names` function, are in this case not the same as the matrix row numbers, given by the `row` function; this is because some field observations were not included in this dataset. In the case of row `69` the row name is 76:

```
> meuse@data[69, ]

   cadmium copper lead zinc elev    dist  om ffreq soil lime
76     3.4     55  325  778 6.32 0.57588 6.9     1    1    0
```

```
   landuse dist.m zinc.l cadmium.l copper.l lead.l  zn.i
76      Bw    750  2.891   0.53148   1.7404 2.5119 FALSE
```

```
> row.names(meuse@data)[69]
```

```
[1] "76"
```

Task 48 : Plot the suspicious regression residuals in geographic space. •

We use the `row` function to extract the rows of the data frame, and the `%in%` set operator to determine whether each row matches the list of suspicious points. We then plot the points by their coordinates, using the `coordinates` method to extract these from the spatial object, and the `ifelse` function to distinguish these from the remaining points. We also define a colour ramp to represent the flooding classes, from frequent (red = "danger") to rarely (green = "safe") and use this to select the colour of each point.

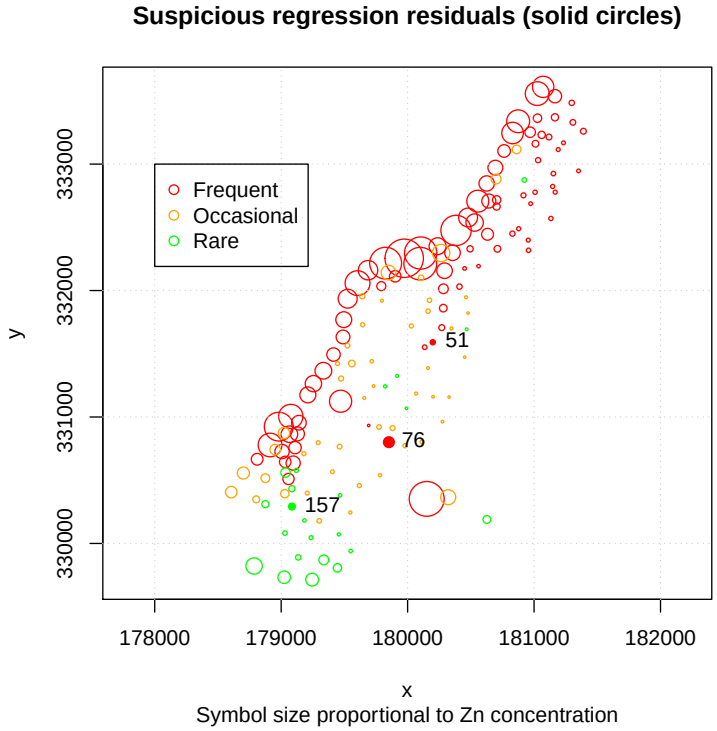
```
> bad.pt <- c(50, 69, 152)
> bad.row <- (row(meuse@data)[, 1] %in% bad.pt)
> (bad.row.names <- row.names(meuse@data)[bad.row])
```

```
[1] "51"  "76"  "157"
```

```
> table(bad.row)
```

```
bad.row
FALSE  TRUE
  152     3
```

```
> colours.ffreq = c("red", "orange", "green")
> plot(coordinates(meuse), asp = 1, col = colours.ffreq[meuse$ffreq],
+     pch = ifelse(bad.row, 20, 1), cex = 4 * meuse$zinc/max(meuse$zinc),
+     main = "Suspicious regression residuals (solid circles)",
+     sub = "Symbol size proportional to Zn concentration")
> grid()
> legend(178000, 333000, pch = 1, col = colours.ffreq,
+     legend = c("Frequent", "Occasional", "Rare"))
> text(coordinates(meuse)[bad.row, ], bad.row.names,
+     pos = 4)
```



It's unclear from the figure why observations 51 and 157 are poorly-modelled; they seem to match nearby points both in their flood frequency class, distance from river, and Zn concentration. However, observation 76 (the highest residual) is clearly anomalous: listed as frequently-flooded although far from the river, and with a much higher Zn concentration than any point in its neighbourhood, even within the same flooding class. This point should be checked for (1) recording error; (2) a different process.

Overall the model is satisfactory, so we continue with the mixed feature space – geographic space model, after removing the temporary variables from the workspace:

```
> rm(bad.pt, bad.row, bad.row.names)
```

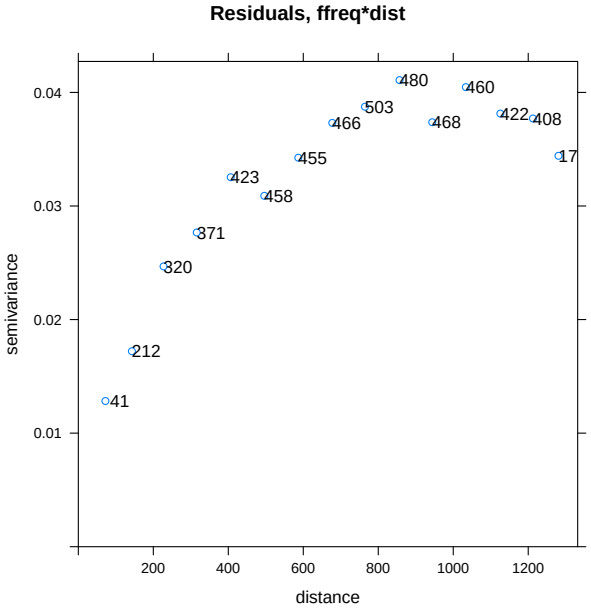
11.4.2 Spatial structure of the the residuals

Now that we've identified a good model (substantially better than the single-predictor model with just flooding frequency), we continue with the local structure of the residuals.

Task 49 : Compute and model the residual variogram from this feature-space model. Compare the models with the single-predictor residual variogram model. and the no-predictor variogram model. •

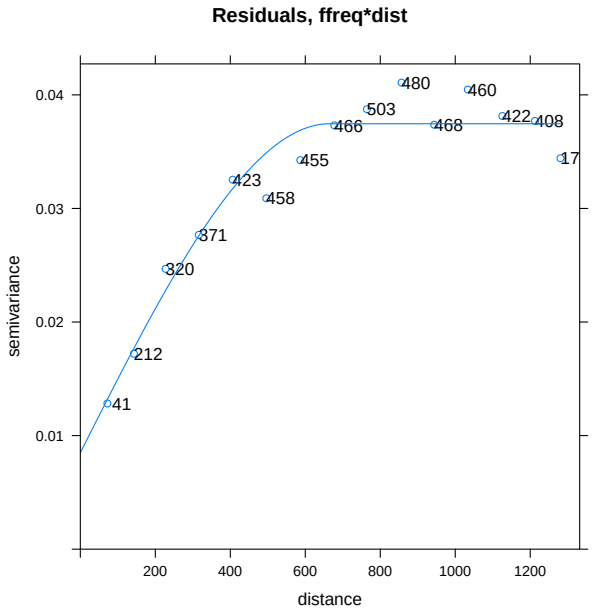
```
> (vr2 <- variogram(zinc.l ~ ffreq * dist, location = meuse,
+     cutoff = 1300, width = 90))
```

```
    np     dist    gamma dir.hor dir.ver   id
1   41   72.248 0.012830       0       0 var1
2  212  142.880 0.017211       0       0 var1
3  320  227.322 0.024683       0       0 var1
4  371  315.855 0.027669       0       0 var1
5  423  406.448 0.032533       0       0 var1
6  458  496.094 0.030899       0       0 var1
7  455  586.786 0.034251       0       0 var1
8  466  677.396 0.037318       0       0 var1
9  503  764.557 0.038742       0       0 var1
10 480  856.694 0.041085       0       0 var1
11 468  944.029 0.037375       0       0 var1
12 460 1033.623 0.040467       0       0 var1
13 422 1125.632 0.038139       0       0 var1
14 408 1212.623 0.037706       0       0 var1
15 173 1280.654 0.034421       0       0 var1

> print(plot(vr2, plot.numbers = T, main = "Residuals, ffreq*dist"))
```




```
> (vrm2f <- fit.variogram(vr2, vgm(psill = 0.04, model = "Sph",
+     range = 700, nugget = 0.01)))

  model     psill  range
1   Nug 0.0084928   0.00
2   Sph 0.0289650 664.78

> vrmf

  model     psill  range
1   Nug 0.0040945   0.00
2   Sph 0.0740612 752.11

> vmf

  model    psill  range
1   Nug 0.010041   0.00
```

```
2   Sph 0.115257 967.26

> print(plot(vr2, plot.numbers = T, model = vrm2f,
+     main = "Residuals, ffreq*dist"))
```



Q46 : *What is the trend of the ranges and partial (structural) sills as the model includes more predictors?* *Jump to A46* •

11.4.3 KED prediction

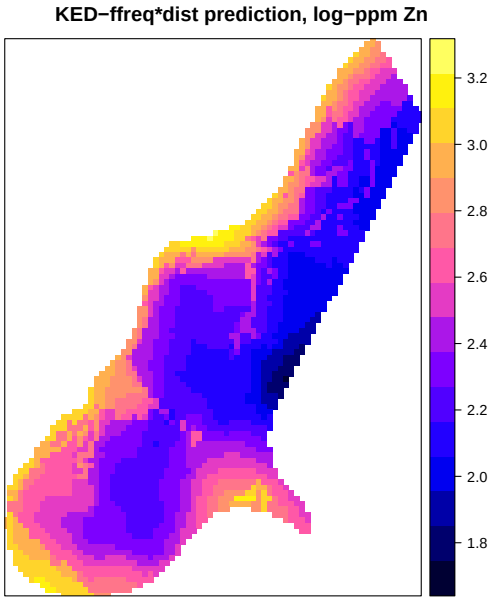
Task 50 : Predict by KED using the best feature-space model as covariables. •

```
> kr240 <- krige(zinc.l ~ ffreq * dist, locations = meuse,
+     newdata = meuse.grid, model = vrm2f)

[using universal kriging]
```

! → Again, the `krige` method **must** use the same model formula as the empirical variogram computed by the `variogram` function.

Task 51 : Display the map of predicted values. •

```
> print(spplot(kr240, "var1.pred", asp = 1, col.regions = bpy.colors(64),
+     main = "KED-ffreq*dist prediction, log-ppm Zn"))
```



Q47 : *How does this KED map compare to the OK map, and the single-predictor (flood frequency) KED map?*

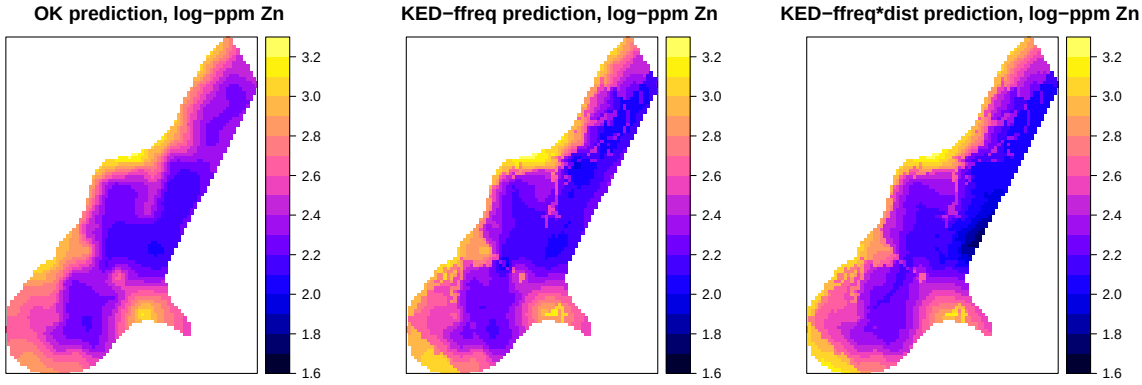
Where is the effect of flood frequency class and distance to river reflected in the prediction? *Jump to A47* •

Task 52 : Display the three predictions side-by-side. •

We repeat the technique of standardizing the ranges of several plots and displaying them in a grid, but now with three plots.

```
> zmax <- round(max(k40$var1.pred,kr40$var1.pred,kr240$var1.pred), 1) + 0.1
> zmin <- round(min(k40$var1.pred,kr40$var1.pred,kr240$var1.pred), 1) - 0.1
> ramp <- seq(from=zmin, to=zmax, by=.1)
> p1 <- spplot(k40, "var1.pred", asp=1, col.regions=bpy.colors(64),
+              main="OK prediction, log-ppm Zn", at=ramp)
> p2 <- spplot(kr40, "var1.pred", asp=1, col.regions=bpy.colors(64),
+              main="KED-ffreq prediction, log-ppm Zn", at=ramp)
> p3 <- spplot(kr240, "var1.pred", asp=1, col.regions=bpy.colors(64),
+              main="KED-ffreq*dist prediction, log-ppm Zn", at=ramp)
```

```
> plot(p1, split = c(1, 1, 3, 1), more = T)
> plot(p2, split = c(2, 1, 3, 1), more = T)
> plot(p3, split = c(3, 1, 3, 1), more = F)
```

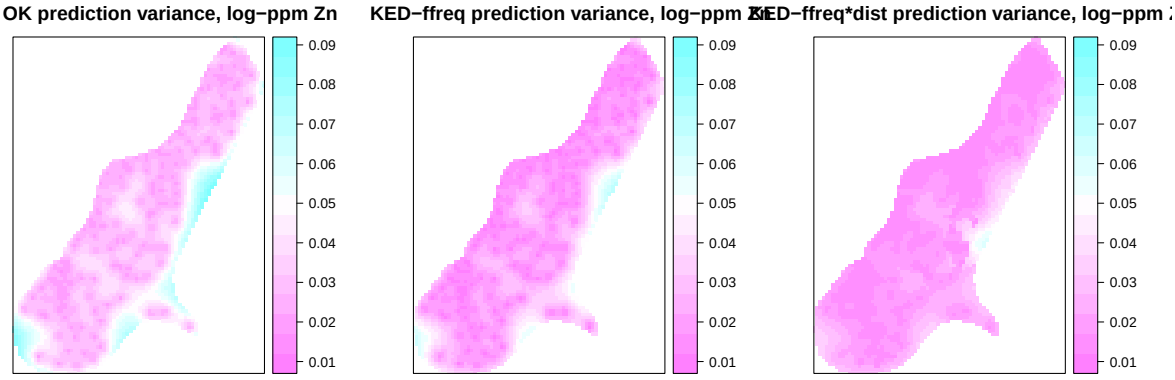


11.4.4 KED prediction variances

Task 53 : Compare these prediction variances to those for OK, both numerically and graphically. •

```
> summary(kr240$var1.var)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0121  0.0160  0.0182  0.0205  0.0231  0.0598

> summary(kr40$var1.var)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00807 0.01630 0.02050 0.02360 0.02810 0.06860

> summary(k40$var1.var)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0166  0.0260  0.0305  0.0347  0.0394  0.0923

> zmax <- round(max(k40$var1.var,kr40$var1.var, kr240$var1.var), 3) + 0.001
> zmin <- round(min(k40$var1.var,kr40$var1.var, kr240$var1.var), 3) - 0.001
> (ramp <- seq(from=zmin, to=zmax, by=.005))

 [1] 0.007 0.012 0.017 0.022 0.027 0.032 0.037 0.042 0.047 0.052
[11] 0.057 0.062 0.067 0.072 0.077 0.082 0.087 0.092

> p1 <- spplot(k40, "var1.var", asp=1, at=ramp,
+              main="OK prediction variance, log-ppm Zn")
> p2 <- spplot(kr40, "var1.var", asp=1, at=ramp,
+              main="KED-ffreq prediction variance, log-ppm Zn")
> p3 <- spplot(kr240, "var1.var", asp=1, at=ramp,
+              main="KED-ffreq*dist prediction variance, log-ppm Zn")
```

```
> plot(p1, split = c(1, 1, 3, 1), more = T)
> plot(p2, split = c(2, 1, 3, 1), more = T)
> plot(p3, split = c(3, 1, 3, 1), more = F)
```



Q48 : *What is the effect of adding the distance to river as a predictor on the spatial pattern of the prediction variances?* *Jump to A48* •

12 Cross-validation

We've produced some nice-looking maps; but the question remains, how good are they? This is the issue of model **evaluation**, often called model **validation**[4]

One of the outstanding features of kriging is that the *same* dataset can be used to model and predict, and to evaluate the predictions. This is called **cross-validation**. The idea is to predict at **known** points, using all the other data and the variogram model, and compare the predictions to reality:

1. Build a variogram model from the known points;
2. *For each* known point:
   (a) Remove it from the dataset;
   (b) Use the model to predict at this point from the others;
   (c) Compute the *residual*, i.e. difference between known and predicted.
3. Summarize the residuals.

This is called **leave-one-out cross-validation** (LOOCV).

> **Note:** LOOCV can be used for any local interpolator, such as nearest-neighbour or inverse distance. This provides an objective measure by which to compare interpolators.

[4] The author prefers the term "evaluation" because it is never possible call a model "valid", only "acceptable" according to certain criteria.

Task 54 : Perform LOOCV for the OK and KED predictions. •

The `krige.cv` method performs this.

```
> kcv.ok <- krige.cv(zinc.l ~ 1, locations = meuse,
+     model = vmf)
> kcv.rk <- krige.cv(zinc.l ~ ffreq, locations = meuse,
+     model = vrmf)
> kcv.rk2 <- krige.cv(zinc.l ~ ffreq * dist, locations = meuse,
+     model = vrm2f)
```

Note the use of the appropriate model formula and variogram model for the two types of kriging: the original variogram model (`vmf`) for OK and the residual variogram model (`vrmf` and `vrm2f`) for KED.

Task 55 : Summarize the results of the OK cross-validation. •

```
> summary(kcv.ok)

Object of class SpatialPointsDataFrame
Coordinates:
     min    max
x 178605 181390
y 329714 333611
Is projected: NA
proj4string : [NA]
Number of points: 155
Data attributes:
   var1.pred       var1.var         observed
 Min.   :2.10   Min.   :0.0221   Min.   :2.05
 1st Qu.:2.33   1st Qu.:0.0295   1st Qu.:2.30
 Median :2.55   Median :0.0332   Median :2.51
 Mean   :2.56   Mean   :0.0351   Mean   :2.56
 3rd Qu.:2.75   3rd Qu.:0.0379   3rd Qu.:2.83
 Max.   :3.15   Max.   :0.1023   Max.   :3.26
    residual            zscore             fold
 Min.   :-0.42847   Min.   :-2.3133   Min.   :  1.0
 1st Qu.:-0.09643   1st Qu.:-0.5011   1st Qu.: 39.5
 Median :-0.00452   Median :-0.0242   Median : 78.0
 Mean   :-0.00015   Mean   :-0.0001   Mean   : 78.0
 3rd Qu.: 0.08528   3rd Qu.: 0.4550   3rd Qu.:116.5
 Max.   : 0.64139   Max.   : 3.2204   Max.   :155.0
```

The spatial points dataframe returned by `krige.cv` has fields for the prediction (field `var1.pred`), the observation (field `observed`), and their difference (field `residual`). A **positive** residual is an **under-prediction** (predicted less than observed).

Task 56 : Compare the results of the OK and KED cross-validations. •

The appropriate measure is the residuals, i.e., how close to the truth?

```
> summary(kcv.ok$residual)
```

```
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.42800 -0.09640 -0.00452 -0.00015  0.08530  0.64100

> summary(kcv.rk$residual)

    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.56100 -0.08340 -0.01360  0.00094  0.08990  0.47700

> summary(kcv.rk2$residual)

    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.56100 -0.08460  0.00142  0.00141  0.07840  0.59100
```

Q49 : *Is any prediction **biased**? (Compare the mean to zero). Which has the narrower overall and inter-quartile range?* *Jump to A49* •

An overall measure is the **root of the mean squared error**, RMSE:

```
> sqrt(sum(kcv.ok$residual^2)/length(kcv.ok$residual))

[1] 0.17256

> sqrt(sum(kcv.rk$residual^2)/length(kcv.rk$residual))

[1] 0.14102

> sqrt(sum(kcv.rk2$residual^2)/length(kcv.rk$residual))

[1] 0.14486
```

Q50 : *Which prediction is, on average, more precise?* *Jump to A50* •

Adding the co-variable (flooding frequency) improved the precision somewhat; the more complex model with flooding frequency and distance in fact decreased the precision a bit.

Another evaluation criterion in a **spatial** prediction is the spatial distribution of the cross-validation residuals. The `sp` package provides a nice `bubble` function which produces a so-called **bubble plot**: the symbol size is proportional to the absolute value, and the sign is shown by a colour.

Task 57 : Display bubble plots of the OK and KED (flood frequency and distance interaction) cross-validations. •

Again we harmonize the legend scales:

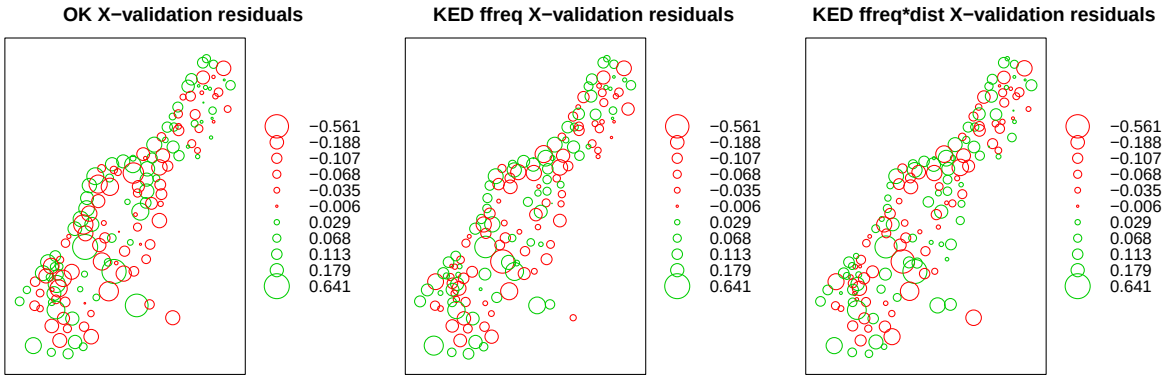
```
> (zmax <- round(max(kcv.ok$residual, kcv.rk$residual,
+     kcv.rk2$residual), 2) + 0.01)

[1] 0.65

> (zmin <- round(min(kcv.ok$residual, kcv.rk$residual,
+     kcv.rk2$residual), 2) - 0.01)
```

```
[1] -0.57
```

```
> ramp <- quantile(c(kcv.ok$residual, kcv.rk$residual,
+     kcv.rk2$residual), probs = seq(0, 1, by = 0.1))
> p1 <- bubble(kcv.ok, zcol = "residual", main = "OK X-validation residuals",
+     key.entries = ramp, pch = 1)
> p2 <- bubble(kcv.rk, zcol = "residual", main = "KED ffreq X-validation residual
+     key.entries = ramp, pch = 1)
> p3 <- bubble(kcv.rk2, zcol = "residual", main = "KED ffreq*dist X-validation re
+     key.entries = ramp, pch = 1)
```

And again print three maps side-by-side:

```
> plot(p1, split = c(1, 1, 3, 1), more = T)
> plot(p2, split = c(2, 1, 3, 1), more = T)
> plot(p3, split = c(3, 1, 3, 1), more = F)
```



Q51 : *Is there any pattern to the positive and negative residuals? Is there a difference between the OK and KED patterns?* *Jump to A51* •

It's always good practice to remove temporary variables:

```
> rm(zmax, zmin, ramp, p1, p2, p3)
```

If you wish, you can save the results of the advanced modelling:

```
> save.image(file = "minimal_geostat_day2.RData")
```

Now if you wish you can exit R with the `q` "quit" function, or you can use the normal way to leave a program.

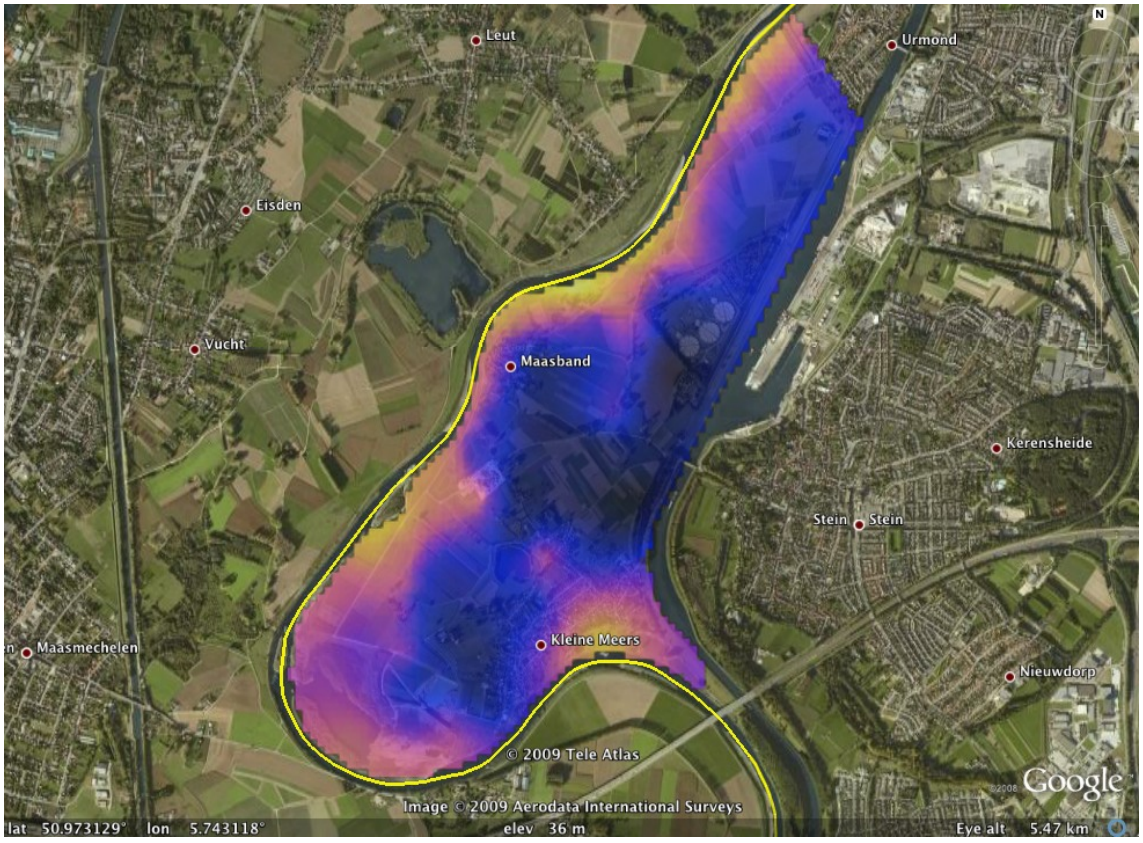
```
> q()
```

13 Final words

There is much, much more to geostatistics than this simple introduction. Some good reference texts are by Goovaerts [6], Webster & Oliver [13] and Isaaks & Srivastava [8]. For spatial analysis in R, the essential reference is Bivand *et al.* [1]. For general modelling in R, a good text is Fox [5].

The approach to mixed modelling taken here, kriging with external drift (KED) (§11.3), has proven to perform well in many studies; however its theoretical basis is weak. This is because the feature-space model does not take into account the spatial correlation of the residuals, which is clearly present as shown in the residual variogram. This violates an assumption of ordinary least squares. There are three ways to consider both the feature space and geographic space models at the same time: (1) using generalized least-squares (GLS) instead of OLS to fit the linear model, considering the covariance structure of the residuals to assign weights, (2) REML fitting of mixed models [9] and (3) model-based geostatistics [3].

The spatial analysis in R can be linked to other GIS; here is an example: a Google Earth image of the Meuse study area, with the kriging interpolation overlaid:



14 Answers

A1 : *Before: none (reported as* `character(0)`*, meaning a zero-length vector of characters); After: one object,* `meuse`*.* *Return to Q1* •

A2 : *155 observations (cases) and 14 fields (variables).* *Return to Q2* •

A3 : *Fields* `x` *and* `y` *are coordinates, so this is spatial data. But, this is not explicit in the dataframe, we have to read the documentation to know what each field represents.* *Return to Q3* •

A4 : *mg* kg^{-1} *(popularly called "parts per million" on a weight basis).* *Return to Q4* •

A5 : *The distribution is not symmetric, it is strongly right-skewed (decreasing number of observations at increasing values of the variable). There may be two populations: The skewed distribution from 0 to about 1200, and then six very high values. But with this small sample size, it may just be one population.* *Return to Q5* •

A6 : *Minimum: 113, first quartile: 198, median: 326, third quartile: 674, maximum: 1840, mean: 470.* *Return to Q6* •

A7 : *The mean is well above the median, this implies a right-skew.* *Return to Q7* •

A8 : *The distribution is now symmetric with no outliers. But the clear dip in frequency around 2.4 to 2.8 log(mg* kg^{-1}*) shows clearly two partially-overlapping populations: low and high pollution.* *Return to Q8* •

A9 : *The two variables are strongly related in feature space. The relation appears to be bivariate normal, so that a linear correlation is appropriate. There are four clear observations that do not fit the pattern: relatively high Cu and low Zn.* *Return to Q9* •

A10 : *All the metals are related in feature space, fairly strongly and linearly. The exception is low Cd levels, where there is a "striping" pattern in the scatterplot due to the precision of the laboratory procedure at low concentrations of Cd.* *Return to Q10* •

A11 : *The most frequently flooded soils have all the high metal concentrations, as*

well as a higher median. The other two classes are almost identical. But, all three classes contain observations with low concentrations. *Return to Q11* •

A12 : *Variance explained: 24.3%* *Return to Q12* •

A13 : *Mean concentration Flood Frequency class 1: 2.7 log(mg kg^{-1}); class 2: 2.368 log(mg kg^{-1}) (i.e., 2.7 + -0.332); class 3: 2.425 log(mg kg^{-1}).* *Return to Q13* •

A14 : *Slots* `@coords`, `@coords.nrs`, `bbox`, `proj4string` *refer to the spatial structure; slot* `@data` *to the feature-space (attribute) data.* *Return to Q14* •

A15 : *The points are unevenly distributed. Many are along the river; the density away from the river is lower, and there are some unsampled areas.* *Return to Q15* •

A16 : *Yes, big circles tend to be near other big ones, same for small circles.* *Return to Q16* •

A17 : *There are (155 * (155-1))/2 = 11935 point-pairs.* *Return to Q17* •

A18 : *Separation is 70.84 m, semivariance is 0.001144 log(mg kg^{-1})^2.* *Return to Q18* •

A19 : *Average separation is 72.25 m, average semivariance is 0.0265 log(mg kg^{-1})^2; this is an estimate from 41 point-pairs.* *Return to Q19* •

A20 : *The evidence is that at closer separations the semivariance is, on average, lower. This increases steadily until the range.* *Return to Q20* •

A21 : *At about 850 m separation there is not any reduction in average semivariance; this is the range.* *Return to Q21* •

A22 : *The trend in decreasing semivariance with decreasing separation seems to intersect the y-axis (i.e., at 0 separation) at about 0.01 log(mg kg^{-1})^2; this is the nugget.* *Return to Q22* •

A23 : *At the range and beyond the average semivariance is about 0.13 log(mg kg^{-1})^2;*

this is the total sill. *Return to Q23* •

A24 : *The fitted model is: nugget 0.01 log(mg kg^{-1})^2, partial sill 0.1153 log(mg kg^{-1})^2, range 967 m. So the total sill is 0.1253 log(mg kg^{-1})^2. Compared to the estimate, the range is longer, the nugget almost the same, and the structural sill a bit lower.* *Return to Q24* •

A25 : *The kriged map is very smooth. "Hot" and "cold" spots are clearly controlled by observations with especially high and low values.* *Return to Q25* •

A26 : *The kriging prediction variances are lowest at observation points, still low near many points, and highest away from any points. There is no relation with the data value or kriged prediction.* *Return to Q26* •

A27 : *Only 15 of the 155 observations above the threshold.* *Return to Q27* •

A28 : *The range is 1338 m; the range of the variogram of the value was 967 m. Thus there is stronger spatial dependence if we just want to know if the value is above or below this threshold.* *Return to Q28* •

A29 : *The total sill is 0.1432; units are dimensionless.* *Return to Q29* •

A30 : *Only the right-centre is "definitely" safe. One could pick any probability threshold (depending on one's risk tolerance) and slice the map at that value.* *Return to Q30* •

A31 : *The prediction is simply a reclassification of each pixel according to the mean of its flood-frequency class, so there are only three predicted values, corresponding to the three classes. The variance is from the linear model summary and is also uniform for each class; it depends on the variance of that class' observations. The spatial pattern is exactly the same as the map of flood-frequency classes.* *Return to Q31* •

A32 : *This empirical variogram has a shorter range (about 500 instead of about 950 m) and a lower total sill (about 0.08 instead of about 0.125 log(mg kg^{-1})^2; nugget estimate 0.01 log(mg kg^{-1})^2 is about the same (indeed, theory requires that the nugget be exactly the same).* *Return to Q32* •

A33 : *Confirming the eyeball estimate: the range has reduced by 215 m (about 30%), the total sill by about 0.047 log(mg kg^{-1})^2 (about 25%). The modelled nugget is lower, although by theory it should not be any different.* *Return to Q33* •

A34 : *The KED map clearly shows the boundary of flood frequency class 1*

(frequently flooded). "Hot" and "cold" spots are somewhat smaller because of the shorter variogram model range. *Return to Q34* •

A35 : *Distance from river* `dist`*, flooding frequency class* `ffreq`*, and soil type* `soil`*.* *Return to Q35* •

A36 : *There is a clear inverse relation: further from the river there is, in general, lower metal concentration. All the high concentrations are very close to the river. The relation is diffuse (scattered). It seems (by eye) not quite linear, maybe an inverse power, but not too far from linear.* *Return to Q36* •

A37 : *The single-predictor model with distance has a much lower residual sum-of-squares (RSS) than the single-predictor model with flood frequency.* *Return to Q37* •

A38 : *The prediction is simply a re-assignment to each pixel by a linear function of distance, so the spatial pattern looks exactly like the map of distance to river (i.e., contours of distance) with different units of measure, here the metal concentration. The prediction variance is lowest at the centroid of the metal-vs.-distance plot; this will be* *Return to Q38* •

A39 : *Yes, the two-predictor models give significantly lower residual sum-of-squares.* *Return to Q39* •

A40 : *Yes, the interaction model has lower RSS than the additive model. The probability this is due to chance is only 0.0025.* *Return to Q40* •

A41 : *Again, the interaction model gives the lowest (most negative) AIC. Thus the interaction is significant. The process of pollution depends on how frequently an area is flooded, but within that, the closer distance to river tends to be more polluted. Together these are strong evidence that the pollution comes from river flooding.* *Return to Q41* •

A42 : *Variance explained: 63.5%.* *Return to Q42* •

A43 : *No, there is a clear trend (shown by the red curve): at intermediate fitted values the residuals tend to be negative; at both lower and higher fitted values the residuals tend to be positive. The variance (spread of residuals) is also not the same throughout the range of the fitted values: it is greater at the middle of the range. And of course there are three very poorly-modelled points, indentified in the graph.* *Return to Q43* •

A44 : *Yes, normally-distributed except for the three most positive residuals.*

Return to Q44 •

A45 : *The high-leverage residuals do not have high influence as shown by Cook's distance, this is good.* Return to Q45 •

A46 : *As the model includes more predictors, the total sills decrease and the ranges are shorter. More of the variability is taken out in feature space, leaving less for spatial structure.* Return to Q46 •

A47 : *In this KED map we can see some "banding" due to distance from the river, as well as the boundaries between the flood frequency classes. The effect of distance from river is especially noticeable at the central E edge of the study area, where the predictions are lowest (darkest colours); this is the furthest from the river and so the prediction is lowered.* Return to Q47 •

A48 : *Adding distance to river reduces the prediction variances and makes them almost uniform across the area.* Return to Q48 •

A49 : *The means of the cross-validations are -0.000147 (OK), 0.000937 (KED, flood frequency), and 0.001413 (KED, flood frequency * distance to river). All are quite close to zero, thus almost unbiased.*

KED with flood frequency has the narrowest overall range: 1.0381. The two-factor KED has the narrowest IQR: 0.163. Return to Q49 •

A50 : *The one-factor KED prediction, with flood frequency as co-variable, is most precise; RMSE is 0.141 log(mg kg^{-1}).* Return to Q50 •

A51 : *In all cases the positive residuals are concentrated along the river; these are under-predicted due to the influence of nearby less-polluted sites. The reverse is true for points just behind the river dikes. In the middle of the area the positive and negative residuals are mixed in no apparent pattern. There are differences in detail among the three cross-validations but the pattern is quite similar.* Return to Q51 •

15 Self-test

This section is a small test of how well you mastered this material. You should be able to complete the tasks and answer the questions with the knowledge you have gained from the exercise. After completing it, you can compare with an answer sheet.

For this test we will work with another dataset.

Task 1 : Load the `jura` dataset, provided with the `gstat` package. Examine the structure of the `jura.pred` dataframe; this is the "calibration" dataset. •

Q1 : *How many observations are in the prediction dataframe? How many fields?* •

Task 2 : Display histograms of the copper (Cu) concentration and its log transform. •

Q2 : *Describe the two distributions.* •

Task 3 : Model the log(Cu) concentration as a linear function of rock type and land use separately, and then additively. •

Q3 : *Why might these be reasonable predictors of Cu concentration?* •

Q4 : *(1) Which of the single-predictor models is better? (2) Is the additive model significantly better than the best single-predictor model? (3) How much of the log(Cu) concentration is explained by the additive model?* •

Task 4 : Display the graphs of linear model diagnostics: (1) residuals vs. fitted; (2) normal Q-Q plot of residuals; (3) residuals vs. leverage. •

Q5 : *Comment on the linear model diagnostics: (1) do the residuals have the same spread at all fitted values? (2) are the residuals normally-distributed? (3) do any high-leverage points have high residuals?* •

Task 5 : Convert the prediction dataframe into a spatial object, add the linear model residuals as a field, and display a bubble plot of the residuals. •

Q6 : *Does there appear to be spatial dependence in the residuals?* •

Task 6 : Compute and display the empirical semivariogram of the linear model residuals. Use a cutoff of 0.8 km (i.e., 800 m) and a bin width of 0.08 km (i.e., 80 m). •

Q7 : *Describe the empirical variogram qualitatively. What are the approximate partial (structural) sill, range, and nugget variance?* •

Task 7 : Fit a spherical variogram model to the empirical variogram. •

Q8 : *What are the fitted partial (structural) sill, range, and nugget variance?* •

Task 8 : Convert the `jura.grid` dataframe into a gridded spatial object, and display maps of the two covariates (land use and rock type). •

Task 9 : Predict by Kriging with External Drift (KED) from the prediction points to this grid, using the fitted variogram model of the linear model residuals. Plot the kriging predictions and their variances (or standard deviations). •

Q9 : *Where are the highest and lowest predicted concentrations? How do these relate to the covariables (if at all)?* •

Q10 : *Where are the highest and lowest prediction standard deviations (or variances)? How do these relate to the covariables (if at all) and sample point configuration?* •

Task 10 : Extra: Cross-validate the KED predictions at the calibration points set `jura.cal`, summarize the cross-validation statistics, and plot the cross-validation residuals. •

Q11 : *Is there any apparent spatial pattern to the cross-validation residuals?* •

References

[1] Bivand, R. S.; Pebesma, E. J.; & Gómez-Rubio, V. 2008. *Applied Spatial Data Analysis with R*. UseR! Springer. http://www.asdar-book.org/ 4, 59

[2] Cook, R. & Weisberg, S. 1982. *Residuals and influence in regression*. New York: Chapman and Hall 47

[3] Diggle, P. J. & Ribeiro Jr., P. J. 2007. *Model-based geostatistics*. Springer. ISBN 987-0-387-32907-9 59

[4] Fox, J. 1997. *Applied regression, linear models, and related methods*. Newbury Park: Sage 47

[5] Fox, J. 2002. *An R and S-PLUS Companion to Applied Regression*. Newbury Park: Sage 59

[6] Goovaerts, P. 1997. *Geostatistics for natural resources evaluation*. Applied Geostatistics. New York; Oxford: Oxford University Press 59

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

[8] Isaaks, E. H. & Srivastava, R. M. 1990. *An introduction to applied geostatistics*. New York: Oxford University Press 59

[9] Lark, R. M. & Cullis, B. R. 2004. *Model based analysis using REML for inference from systematically sampled data on soil*. *European Journal of Soil Science* **55**(4):799–813 59

[10] Pebesma, E. J. 2004. *Multivariable geostatistics in S: the gstat package*. *Computers & Geosciences* **30**(7):683–691 4

[11] Pebesma, E. J. & Wesseling, C. G. 1998. *Gstat: a program for geostatistical modelling, prediction and simulation*. *Computers & Geosciences* **24**(1):17–31
URL http://www.gstat.org/ 3

[12] Rikken, M. G. J. & Van Rijn, R. P. G. 1993. *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 69

[13] Webster, R. & Oliver, M. A. 2008. *Geostatistics for environmental scientists*. John Wiley & Sons Ltd., 2nd edition 59

Index of R Concepts

A The Meuse dataset

The project that produced this data set is described in the fieldwork report of Rikken & Van Rijn [12].

R data set The `sp` package includes this as a sample data set named `meuse`, which can be loaded with the `data` function:

```
> data(meuse)
```

This package also includes a 40x40 m interpolation grid of the study area, `meuse.grid`, and a point file which outlines the banks of the Meuse river, `meuse.riv`.

Structure The "Meuse" dataset consists of 155 observations taken on a support of 15x15 m from the top 0-20 cm of alluvial soils in a 5x2 km part of the right bank of the floodplain of the River Maas (in French and English, "Meuse") near Stein in Limburg Province (NL). The left bank of this reach of the Meuse is in Belgium and was not sampled.

The dataset records the following data items (fields) for each observation:

`x, y`	E and N coordinates on the Dutch national grid (RD), meters
`cadmium`	concentration in the soil, in weight mg kg^{-1}; zero cadmium values have been shifted to 0.2 (half the lowest non-zero value, supposed to be the detection limit)
`copper`	concentration in the soil, in mg kg^{-1}
`lead`	concentration in the soil, in mg kg^{-1}
`zinc`	concentration in the soil, in mg kg^{-1}
`elev`	elevation above local reference level, in meters
`dist`	distance from the main Maas channel; obtained from the nearest cell in `meuse.grid`; this in turn was derived by a 'spread' (spatial distance) GIS operation, therefore it is accurate up to 20 metres; normalized to [0, 1] across the study area
`om`	organic matter loss on ignition, as percentage of dry weight
`ffreq`	flood frequency class, 1: annual, 2: 2-5 years, 3: every 5 years
`soil`	soil class, arbitrary code
`lime`	has the land here been limed? 0 or 1 = F or T
`landuse`	land use, coded
`dist.m`	distance from main Maas channel, in meters, from field survey

Soil pollution thresholds According to the Berlin Digital Environmental Atlas[5], the *critical level* for the four metals in soils are 2 mg kg^{-1}(Cd), 25 mg kg^{-1}(Cu), 600 mg kg^{-1}(Pb), and 150 mg kg^{-1}(Zn) for agricultural fields: crops to be eaten by humans or animals should not be grown in these conditions. At half these levels crops must be tested.

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