An introduction to geostatistics with R/gstat

D G Rossiter

Cornell University, Section of Soil & Crop Sciences
南京师范大学地理学学 院

November 21, 2017

Contents

学而不思则罔，思而不学则殆
– 《论语·为政》

"Learning without thinking leads to confusion; thinking without learning ends in danger."
– The Analects: Politics

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** [13, 21] and the `gstat` [20] and `sp` [19] R packages. Sections 1–9 (not including the optional subsections in §7) were designed to be used in a two-day interactive tutorial course[1]; the optional subsections of §7 and §10–§13 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. 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. It is most convenient to run R within an **integrated development environment** (IDE); in this exercise we will use RStudio[2] as explained in §1.

The exercise is organized as a set of **discussions**, **tasks**, **R code** to complete the tasks, self-study **questions** (with answers) to test your understanding, and a few **challenges**. §16 is a small test of how well the material has been mastered.

After completing the exercise, you will have seen just a very small part of the R environment, and the simplest concepts of (geo)statistics. There are many resources for going futher, see §14, which you should consult before undertaking your own geostatistical analyses.

1 Installing R and RStudio

If you do not have R and RStudio on your computer, proceed as follows:

1. Download base R for your operating system from `https://cran.r-project.org`.
2. Install it on your system.
3. Download RStudio desktop version for your operating system from `https://www.rstudio.com/products/RStudio/`.
4. Install it on your system.
5. Start RStudio; you should see a screen like Figure 1.

1.1 Installing contributed packages

In this exercise we will use (at least) two of the thousands of contributed R packages, in addition to the standard packages supplied with a fresh installation of R. To get these, use the "Install" toolbar button of the "Packages" tab in RStudio; see Figure 2.

The first time you do this you will be asked to specify a "mirror", i.e., one of the many servers around the world from which R and packages can be

[1] Originally given at Institut Teknologi Bandung, Indonesia in August 2010
[2] `http://www.rstudio.org`

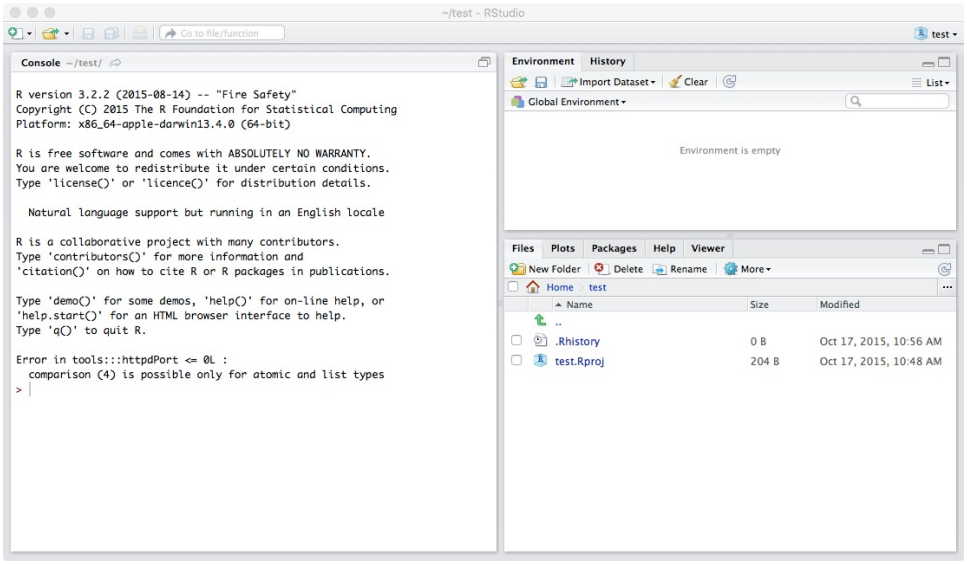


Figure 1: RStudio screen after installation

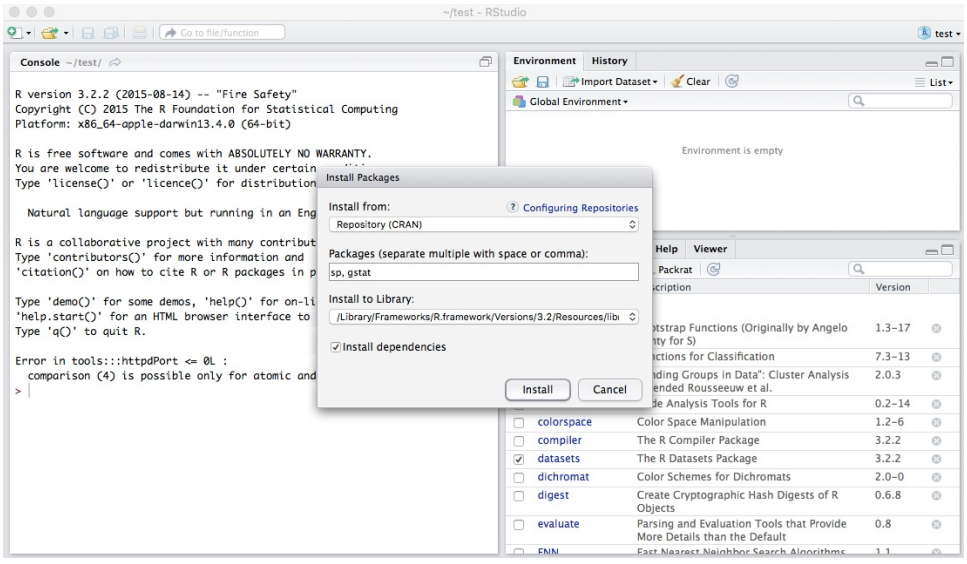


Figure 2: Installing Packages with RStudio

downloaded. They should all have the same packages; so pick one close to you for faster access.

> **Note:** You can also use the `install.packages` function at the console prompt (see below). To install `sp` and `gstat`, use this command, with the packages you want to load as a vector of character strings, using the `c` "catenate" (Latin for "make a chain") function:
>
> ```
> > install.packages(c("sp", "gstat"))
> ```

2 First interaction with R

The simplest way to interact with the R environment is by typing **commands** at the "R console" **command line**; this is the "Console" window in the RStudio IDE. R shows that it is waiting for your command by showing a `>` character as a **prompt** for you to enter a command for R to execute.

The simplest use of R is as a calculator. You type an **expression** and R computes the result.

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:

```
>
```

1. You type an expression, in this case `2*pi/360`, and press the "Enter" key.
2. R then shows the result in the console. 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.

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 [18] and the `sp` package for representing spatial data in R [1].

Task 2 : Load the `sp` and `gstat` packages into the workspace. •

You can load these in RStudio by checking the small "check box" next to the package name in the "Packages" tab; see Figure 3.

You can also load these this from the R console with the `library` function. This loads a package into the **R workspace**:

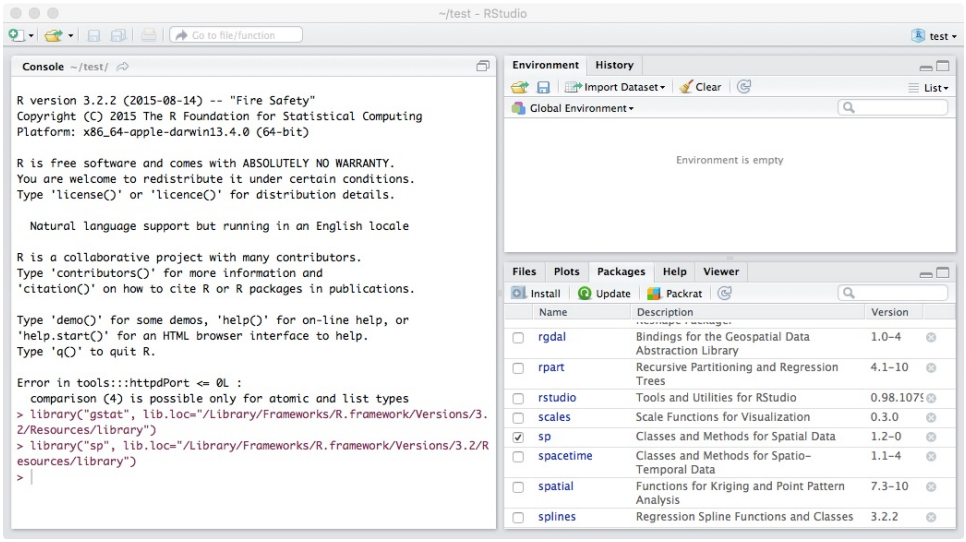


Figure 3: Loading installed Packages with RStudio

```
> library(sp)
> library(gstat)
```

3 Loading and examining the Meuse dataset

Most R packages include **sample datasets** that illustrate the functionality of the package. 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**.

> **Note:** R also provides functions for loading data from external sources, e.g., `read.table` to read delimited text. In this exercise we will not show to how to load your own datasets; see the R Data Import/Export Manual [22].

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)

'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**. The `$` symbol separates the dataframe name from the field name.

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**.

In-program help

All R functions and built-in datasets have **help text** in the R environment.

Task 5 : View the in-program help information for the Meuse dataset. •

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

```
> help(meuse)
```

On some systems this will display in a browser window; in RStudio it will display in the "Help" tab of the bottom-right window pane.

Q3 : *Which fields show that this is **spatial** data, i.e., data where each observation has a known **georeference**?* *Jump to A3* •

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

Figure 4 is a Google Earth view of the Meuse observation points, with their Zn concentrations. The village of Stein is on a high terrace; the study area is the flood plain and active terrace. The Meuse river borders the study area on the S and W; the water body to the E is a canal.

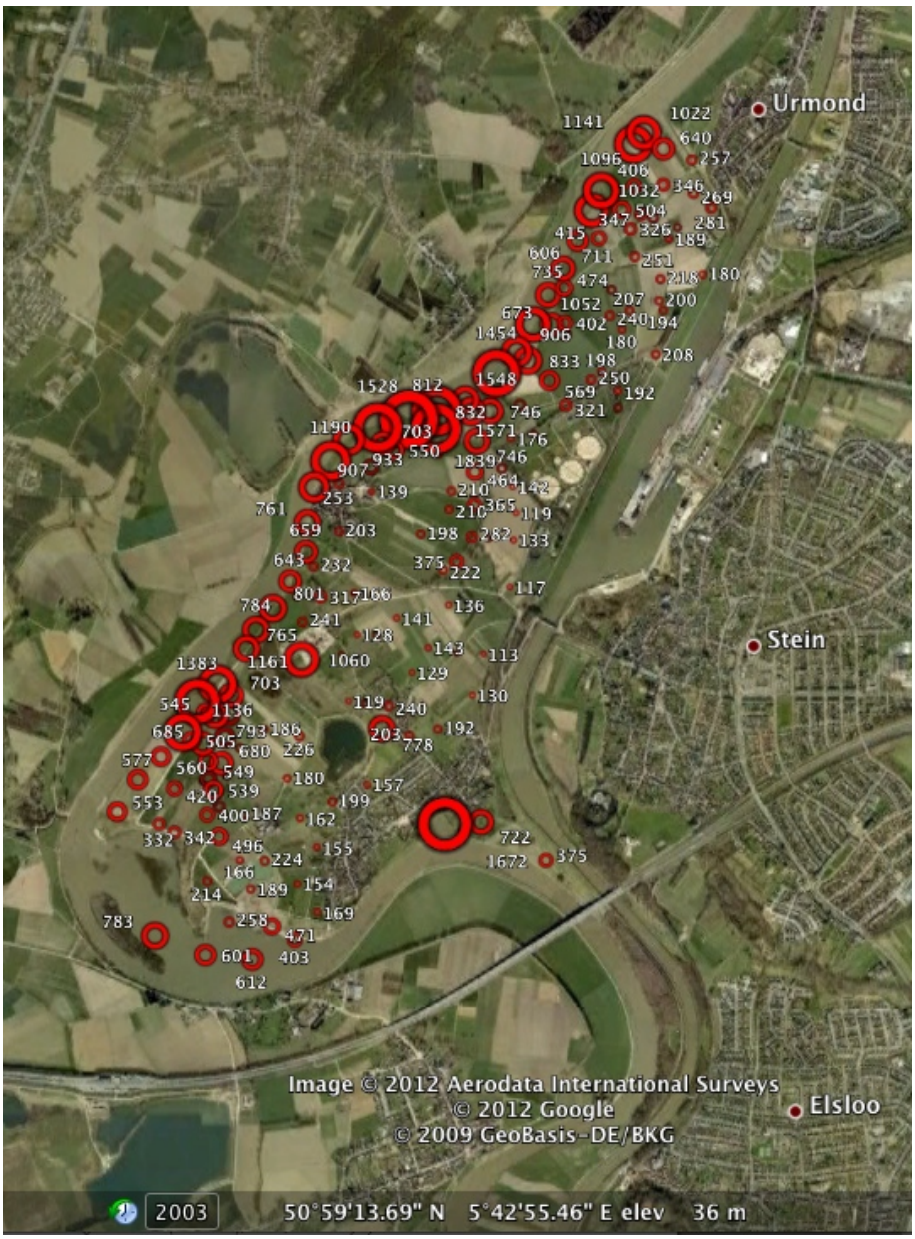


Figure 4: Meuse sample points, with zinc concentrations, shown in Google Earth

4 Taking a break and re-starting

At any point during the exercise you can to take a break, close R and re-start another time. This section explains how to take a break while saving your

work, and then restart where you left off.

Before stopping:

Task 6 : 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")
```

This will save all your workspace objects in this file in the current directory.

Now if you wish you can exit R with the `q` "quit" function, or you can use the normal way to leave a program, e.g., RStudio `File | Close Projet` or `File | Quit RStudio`.

```
> 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).

When you are ready to continue:

Task 7 : 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, also if you re-start RStudio in the same directory.

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] "meuse"
```

However, R does *not* automatically reload add-in packages, so you have to again load `sp` and `gstat`:

```
> library(sp)
> library(gstat)
```

5 Non-spatial 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 8 : Display the actual data values for zinc (Zn) content, both in sample and sort order. •

the $ field separator

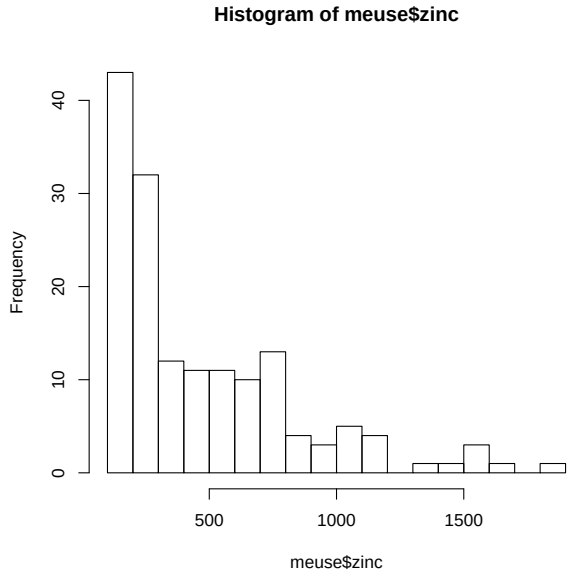
To reference a field in a dataframe, use the `$` field separator to name the field within the dataframe, using the notation `dataframe$fieldname`. To sort a vector (here, the set of Zn concentrations) use the `sort` function:

```
> 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 9 : Display a a histogram and a five-number summary of the zinc content. •

These are obtained with the `hist` function and the `summary` method, respectively:

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

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    113     198     326     470     674    1839
```



Note in the `hist` "histogram" graphics function the use of the optional `breaks` **argument** to (approximately) 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 Zn 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* •

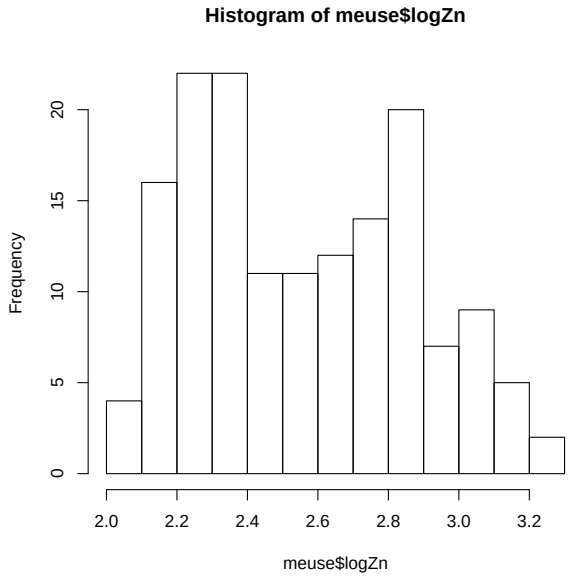
5.1 Transformation

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 10 : 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 $\log_{10}\text{Zn} = 3$, then $\text{Zn} = 10^3 = 1000$). Repeat the summary and histogram. •

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

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



assignment

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

vectorized operations

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$logZn)

[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. In this exercise we will also work with the copper concentration.

Task 11 : Log-transform the copper concentration and attach it as a new field to the data frame. •

```
> meuse$logCu <- log10(meuse$copper)
> str(meuse)

'data.frame':        155 obs. of  16 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 ...
 $ logZn  : num  3.01 3.06 2.81 2.41 2.43 ...
 $ logCu  : num  1.93 1.91 1.83 1.91 1.68 ...
```

Notice the new field `logCu` in the data frame.

6 Non-spatial bivariate EDA

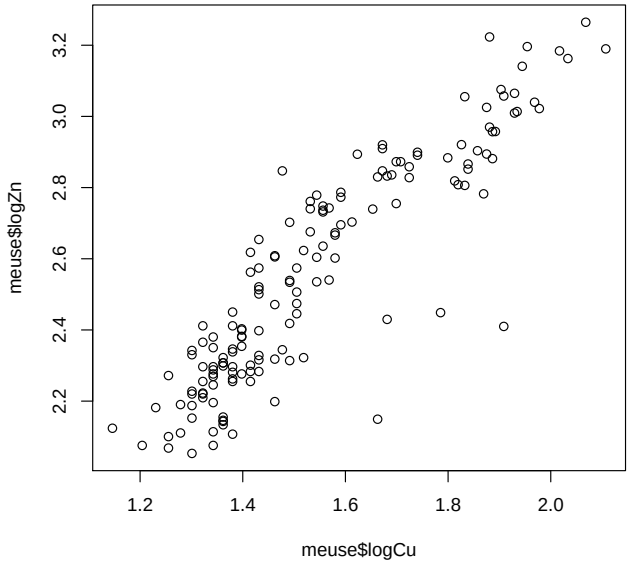
We continue the analysis of **feature** (attribute) space by considering the relation between **two** variables.

Task 12 : Show a **scatterplot** of the relation between log-transformed Zn and Cu. •

the ~ formula operator

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$logZn ~ meuse$logCu)
```



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? What could be the reason(s)?* *Jump to A9* •

Task 13 : Find the observations that do not fit the general pattern of the Cu vs. Zn relation. •

We can clearly see in the scatterplot the four observations that do not fit the pattern. But, which are these? Where are they located? What are their other attributes that might help explain the unusal Cu vs. Zn relation?

Recall, the dataframe is a matrix, and if we can find the rows (observations) in the matrix of these unusual observations, we can use matrix notation to display their records. To find the rows, we need to build up a **vector** of their row numbers, also called **array indices**. R makes these easy with **logical operations**.

logical operators

The `which` function uses a **logical condition** and evaluates which of these is `TRUE` or `FALSE`. It returns the **indices** within the vector of the `TRUE` items. We can then use these indices to examine the corresponding rows in the dataframe.

We can see from the graph that the unusual points have Zn less than about 2.6 $\log_{10}$(mg) kg^{-1} but Cu greater than 1.6 $\log_{10}$(mg) kg^{-1}. First look at two simple conditions:

```
> which(meuse$logZn < 2.6)

 [1]   4   5   6   7   9  10  11  12  15  23  24  25  26  27  28
[16]  29  30  31  32  33  34  35  36  41  42  43  44  48  49  50
[31]  51  68  84  85  86  90  94  95  96  97  98  99 100 101 102
[46] 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117
[61] 119 120 121 122 125 126 127 128 131 132 133 134 135 136 137
[76] 138 140 141 142 143 149 150 151 153 154 155

> which(meuse$logCu > 1.6)

 [1]   1   2   3   4   5   6  13  16  17  18  19  20  21  37  38
[16]  39  40  45  46  52  53  54  55  56  59  60  61  62  63  64
[31]  65  66  67  69  70  71  72  73  75  76  79  80  81  82  83
[46]  88 118 124 135 148
```

These are indices in the dataframe of records (observations) that satisfy the two conditions independently.

Now we combine them with the `&` "and" logical operator. Notice the parentheses around each simple logical condition.

```
> which((meuse$logZn < 2.6) & (meuse$logCu > 1.6))

[1]   4   5   6 135
```

With these indices we can display the dataframe records for these points: their coördinates and the values of their other attributes. But first we can save the indices into the workspace, rather than just display them in the console output as in the previous command. We choose to name the workspace variable `ix`, short for "index"; of course you could use any new name.

```
> ix <- which((meuse$logZn < 2.6) & (meuse$logCu >
      1.6))
```

Now we can use this vector as the **row index** into the dataframe, considered as a **matrix**:

```
> meuse[ix, ]

          x      y cadmium copper lead zinc  elev    dist  om
4    181298 333484     2.6     81  116  257 7.655 0.19009 8.0
5    181307 333330     2.8     48  117  269 7.480 0.27709 8.7
6    181390 333260     3.0     61  137  281 7.791 0.36407 7.8
140  179917 331325     0.8     46   42  141 9.970 0.44558 4.5
    ffreq soil lime landuse dist.m  logZn  logCu
4       1    2    0      Ga    270 2.4099 1.9085
5       1    2    0      Ah    380 2.4298 1.6812
6       1    2    0      Ga    470 2.4487 1.7853
140     3    2    0      Am    540 2.1492 1.6628
```

This example illustrates how R can access a **dataframe** as a **matrix**. The notation `meuse[ix, ]` means: object `meuse`, the rows named in the `ix` workspace variable, and all columns (the blank after the `,`). This is standard matrix notation. Note that the **rows** of the matrix are the observations, and the **columns** are the fields.

7 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". We consider two kinds of models: **linear** (§7.2 and §7.3) and **trees** (§7.5 and §7.7). For the linear models, we have to introduce some theory.

7.1 Theory of linear models

A **linear** model is one in which a **response** variable y, also called the **predictand** or **independent** variable, is modelled as being **linearly dependent** on one or more **predictors** X, also called **independent** variables, with some **residual** ε due to non-exact fit. The **coefficients** of the relation, i.e., the slopes of the linear relation, are a vector β, with one element per predictor, including one for the overall mean.

$$y = \beta X + \varepsilon \tag{1}$$

This can be written in expanded form, showing all p predictors, as:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \ldots + \beta_p x_p + \varepsilon \tag{2}$$

A linear relation means that one unit of change in a predictor x_j, no matter what its value, brings the same amount of change β_j in the predictand y. It is the values of these coefficients in the vector β that we need to find.

Considering just one observation:

$$y_i = \beta X_i + \varepsilon_i \tag{3}$$

where each observation i of n total observations is a pair (X_i, y_i), i.e., the value of the independent and dependent variables at observation i. Note that the same β applies to all observations.

The **residuals** ε_i are defined as $(y_i - \widehat{y_i})$, i.e., actual observed valued vs. the value predicted by the linear model; this is also called the **fitted** value. These residuals ε_i are assumed to be **identically and independently distributed** (IID):

- no relation between the magnitude of the residual and that of the predictor (**homoscedascity**);
- no systematic relation between fitted values and residuals;
- no **serial correlation** between residuals (e.g., small residuals systematically followed by other small residuals) in the sequence of predictors.
- no **dependence** between pairs of residuals; in particular this means **spatial independence**: pairs of residuals at *close* spatial separation are no more likely to be similar to each other than pairs of residuals at *far* spatial separation. If this is not true **mixed predictors**, combining

feature and geographic space, must be used; see later in this exercise §11.

These assumptions can be examined after the regression parameters are estimated, using **regression diagnostics**, and in the case of the last one, the **residual variogram**.

In the simplest case of **univariate** linear regression X_i is a two-element vector $(1, x_i)$, which results in a line with intercept:

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \tag{4}$$

7.1.1 * Least-squares solution of the linear model

In this **optional** section we explain how to find optimal values of the linear model coefficients. This is implemented in the `lm` function of R, which we will use in the following sections.

In the general linear model, with any number of predictors, there is a $n \times p$ **design matrix** of predictor values usually written as $\mathbf{X}$, with one row per observation (data point), i.e., n rows, and one column per predictor, i.e., p columns. In the single-predictor with intercept case, it is a $n \times 2$ matrix with two columns: (1) a column of 1 representing the intercept, and (2) a column of predictor values x_i. The predictand (response variable) is a $n \times 1$ column vector $\mathbf{y}$, one row per observation. The coefficient vector β is a $p \times 1$ column vector, i.e., one row per predictor (here, 2). This multiplies the design matrix to produce the response:[3]

$$\mathbf{y} = \mathbf{X}\beta + \varepsilon \tag{5}$$

where ε is a $n \times 1$ column vector of **residuals**, also called **errors**, i.e., the lack of fit. We know the values in the predictor matrix $\mathbf{X}$ and the response vector $\mathbf{y}$ from our observations, so the task is to find the optimum values of the coefficients vector β.

To solve this we need an optimization criterion. The obvious criterion is to minimize the total error (lack of fit) as some function of $\varepsilon = \mathbf{y} - \mathbf{X}\beta$; the goodness-of-fit is then measured by the size of this error. A common way to measure the total error is by the sum of vector norms; in the simplest case the Euclidean distance from the expected value, which we take to be 0 in order to have an unbiased estimate. If we decide that both positive and negative residuals are equally important, and that larger errors are more serious than smaller, the vector norm is expressed as the sum of squared errors, which in matrix algebra can be written as:

$$S = (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta) \tag{6}$$

which expands to:

$$\begin{aligned} S &= \mathbf{y}^T\mathbf{y} - \beta^T\mathbf{X}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\beta + \beta^T\mathbf{X}^T\mathbf{X}\beta \\ S &= \mathbf{y}^T\mathbf{y} - 2\beta^T\mathbf{X}^T\mathbf{y} + \beta^T\mathbf{X}^T\mathbf{X}\beta \end{aligned} \tag{7}$$

[3] The dimensions of the matrix multiplication are $n \times 1 = (n \times p)(p \times 1)$

Note: $\mathbf{y}^T\mathbf{X}\boldsymbol{\beta}$ is a 1×1 matrix, i.e., a scalar[4], so it is equivalent to its transpose: $\mathbf{y}^T\mathbf{X}\boldsymbol{\beta} = [\mathbf{y}^T\mathbf{X}\boldsymbol{\beta}]^T = \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{y}$. So we can collected the two identical 1×1 matrices (scalars) into one term.

This is minimized by finding the partial derivative with respect the the unknown coefficients $\boldsymbol{\beta}$, setting this equal to $\mathbf{0}$, and solving:

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\beta}^T} S &= -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} \\ \mathbf{0} &= -\mathbf{X}^T\mathbf{y} + \mathbf{X}^T\mathbf{X}\boldsymbol{\beta} \\ (\mathbf{X}^T\mathbf{X})\boldsymbol{\beta} &= \mathbf{X}^T\mathbf{y} \\ (\mathbf{X}^T\mathbf{X})^{-1}(\mathbf{X}^T\mathbf{X})\boldsymbol{\beta} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \\ \hat{\boldsymbol{\beta}}_{\text{OLS}} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \end{aligned} \tag{8}$$

which is the usual OLS solution.

7.2 Continuous response, continuous predictor

Looking at the scatterplots of §6, a natural question is whether one metal concentration can be predicted from another. This could be useful if one metal is measured and another must be estimated.

Task 14 : Model the $\log_{10}$Zn concentration as a linear function of the $\log_{10}$Cu concentration. •

This is also a **linear model**, using the `lm` function, specifying $\log_{10}$Zn as the dependent variable (left-hand side of the model formula) and $\log_{10}$Cu as the independent variable (right-hand side), again using the `~` ("tilde") formula operator symbolizing the **dependence** of the left-hand side on the right-hand side.

```
> m.lzn.lcu <- lm(logZn ~ logCu, data = meuse)
```

The `<-` assignment operator saved the results of the modelling by the `lm` function as workspace object, which we name `m.lzn.lcu`. Note that we wrote the functional dependence as `logZn ~ logCu`, 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.

Task 15 : 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] "ix"        "m.lzn.lcu" "meuse"

> class(meuse)
```

[4] The dimensions of the matrix multiplication are $(1 \times n)(n \times p)(p \times 1)$

```
[1] "data.frame"

> class(m.lzn.lcu)

[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.

7.2.1 Model summary

Task 16 : Display the model summary. •

The `summary` method applied to a linear model object displays a useful summary of the model results:

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

Call:
lm(formula = logZn ~ logCu, data = meuse)

Residuals:
    Min      1Q  Median      3Q     Max
-0.6097 -0.0790 -0.0003  0.0874  0.3769

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.5882     0.0791    7.43    7e-12 ***
logCu         1.2740     0.0507   25.12   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.139 on 153 degrees of freedom
Multiple R-squared:  0.805,        Adjusted R-squared:  0.804
F-statistic:  631 on 1 and 153 DF,  p-value: <2e-16
```

The summary shows:

1. the model **formula** that we specified;
2. a summary of the **residuals**, the values after subtracting the model fit, i.e., actual - fitted by the model;
3. The two coefficients of the linear model:
   - `(Intercept)`: the predicted value of the response ($\log_{10}$Zn) if the predictor ($\log_{10}$Cu) were zero; this is $\widehat{\beta_0}$.
   - `logCu`: the **slope** of the regresssion line: the amount the response changes, on average, for each unit change in the predictor, here $\log_{10}$Cu; this is $\widehat{\beta_1}$
4. the **standard errors** of the coefficients and the probability that rejecting the null hypothesis of 0 would be an error;

5. the residual standard error, an estimate of σ, the standard deviation of the normally-distributed residuals, i.e., how closely does the model fit the known values;
6. the **adjusted R-squared**; this gives the proportion of the variance of the response variable explained by the model.

Q10 : *How much of the variability in the log_{10}Zn content of these topsoils can be explained if we know the log_{10}Cu contents at the same locations?* *Jump to A10* •

There are two model coefficients: (1) the intercept, which is the value of the response variable (here, $\log_{10}$Zn) at the zero value of the predictor (here, $\log_{10}$Cu, i.e., when Cu = 1 mg kg^{-1}), and (2) the slope, which is the change in the response per unit change in the predictor. These each have a standard error, i.e., one standard deviation of uncertainty.

Q11 : *If log_{10}Cu increases by one unit (i.e., increases ten-fold; recall this is a log_{10} transformation), how much does log_{10}Zn increase? What is the standard error of that coefficient?* *Jump to A11* •

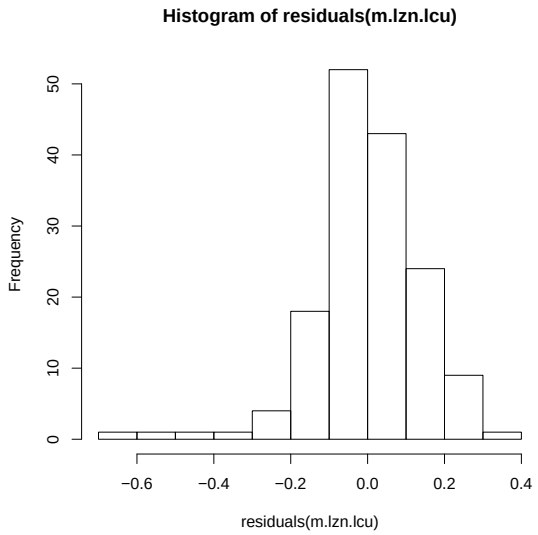
7.2.2 Model diagnostics

Linear modelling is a complicated topic, covered in many texts, e.g., [8]. A fundamental requirement for the ordinary least squares (OLS) fit such as computed by the `lm` function is that the **residuals** must be **independent and identically normally-distributed**; if this assumption is not met various adjustments must be made or other methods used; consult a text.

Task 17 : Examine a histogram of the model residuals. •

The `residuals` function extracts the residuals from a model object, and of course the `hist` function display the histogram of a vector in the base graphics system:

```
> hist(residuals(m.lzn.lcu))
```



Q12 : *Do the residuals appear to be normally-distributed?* *Jump to A12* •

A linear model must satisfy several assumptions [8, 5], 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.

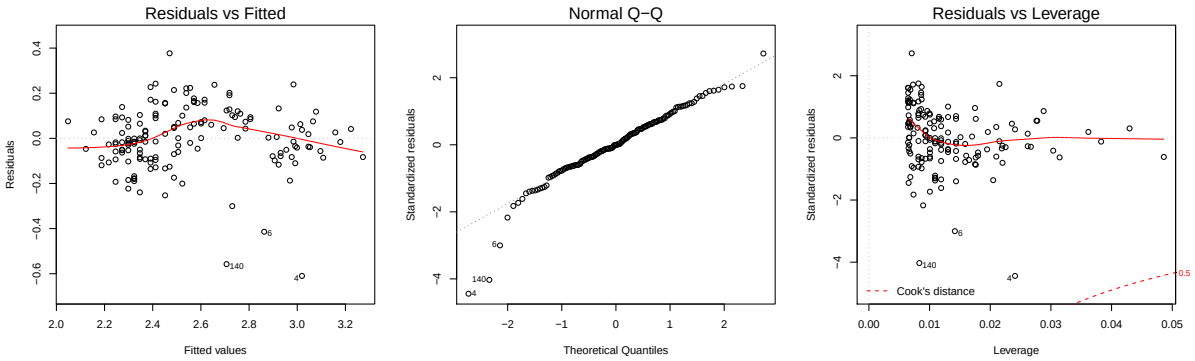
We can view these graphically, with the `plot` method, which specializes to the `plot.lm` function if it is called on objects of class `lm`. This function produces six different plots; the most useful are `1` "Residuals vs. fitted values", `2` "Normal Q-Q", and `5` "Residuals vs. leverage"; see `?plot.lm` for more options.

Task 18 : Display a plot of (1) residuals-vs-fitted values, (2) quantile-quantile plot of the residuals compared to a normal distribution, (3) residuals vs. leverage. •

Note that the `which` argument of the `plot.lm` function selects which of the six possible diagnostic plots to display. These are `1` for residuals-vs-fitted values, `2` for the quantile-quantile plot, and `5` for residuals vs. leverage[5]. The `mfrow` argument to the `par` "graphics parameters" function specifies the layout of multiple graphs; here we want one row and three columns.

[5] see `?plot.lm`

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



1. The "Residuals vs. fitted values" plot shows the residual (actual - fitted value) for all the known points. That is, using the regression equation just computed, and using the known values of the predictor variables at each point, we can estimate the response variable at that point. We then can compare it with the known value at that point. The mean residual is by definition 0 for ordinary least squares regression (OLS), as used here. There should be no pattern of residuals vs. fitted values, i.e., no systematic over- or under-prediction at a given range.

2. The "Normal Q-Q" plot shows the quantiles of the standardized residuals (i.e., mean = 0, then ± a number of standard deviations) on the y-axis, vs. the quantiles of a normal distribution with the same mean and standard deviation. If the residuals are normally-distributed (an assumption of the OLS fit) the points, representing observations, should all fall on the 1:1 line and become sparser (thinner) at the extremes.

3. The "Residuals vs. leverage" plot shows the "leverage" h_i of each observation[6], against its standardized residual r_{S_i}[7]. The leverage of an observation measures how much the influence the observation has on the fits, i.e., how much the fits would change should that observation be removed from the dataset. There should not be any high-leverage points with large standardized residuals. This would indicate that the point has high influence on the fits, but itself is not well-fit; this could be because the point is from a different population or is otherwise unusual, and is distorting the overall relation. This situation is revealed by the "Cook's distance".

[6] $h_i = \mathbf{H}_{ii}$, i.e., the diagonal of the "hat" matrix $\mathbf{H} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$

[7] The residual $r_i = y_i - \hat{y}_i$ is standardized by dividing by an estimate of its standard deviation, $\hat{\sigma}\sqrt{1-h_i}$, where $\hat{\sigma}$ is the estimate of the standard deviation of the residuals. That is, the greater the leverage h_i, the smaller the variance of the corresponding r_i, so the greater the adjustment by standardization

This plot also shows contours for the Cook's distance, which is a measure of the difference between the vector β of regression coefficients computed with all observations, and the vector $\beta_{(-1)}$ of regression coefficients computed *without* a single observation i.[8] A large value of Cook's distance shows that observation i has a large influence on the regression coefficients, i.e., if it were omitted, the coefficients would be substantially different. A rule of thumb is that observations with a Cook's distance $D_i > 0.5$ is cause for concern and $D_i > 1$ indicates that observation i has an undue effect on the regression coefficients.

Q13 : *Is there any pattern with respect to the fitted values (see the first diagnostic plot)?* *Jump to A13* •

Task 19 : Identify the observations that do not fit the overall pattern and display their data and model residuals. •

The `which` function identifies observations in a data frame which meet some **logical condition**, here that the absolute residual (determined with functions `abs` and `residuals`) is larger than some threshold:

```
> which.ix <- which(abs(residuals(m.lzn.lcu)) > 0.3)
> meuse[which.ix, ]

          x      y cadmium copper lead zinc  elev     dist  om
4    181298 333484     2.6     81  116  257 7.655 0.190094 8.0
5    181307 333330     2.8     48  117  269 7.480 0.277090 8.7
6    181390 333260     3.0     61  137  281 7.791 0.364067 7.8
129 179849 332142     1.2     30  244  703 8.540 0.092135 8.3
140 179917 331325     0.8     46   42  141 9.970 0.445580 4.5
    ffreq soil lime landuse dist.m  logZn  logCu
4       1    2    0      Ga    270 2.4099 1.9085
5       1    2    0      Ah    380 2.4298 1.6812
6       1    2    0      Ga    470 2.4487 1.7853
129     2    1    0      Fw     70 2.8470 1.4771
140     3    2    0      Am    540 2.1492 1.6628

> residuals(m.lzn.lcu)[which.ix]

       4        5        6      129      140
-0.60973 -0.30039 -0.41405  0.37686 -0.55738

> which.max(residuals(m.lzn.lcu)[which.ix])

129
  4
```

[8] $D_i = \frac{r_i}{\text{trace}(\mathbf{H})} \frac{h_i}{1-h_i}$, where r_i is the ith residual, h_i is the ith diagonal element of the "hat" matrix $\mathbf{H} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$, and its trace is just the number of predictors (including the intercept) p. Thus observations with poor fits (large residuals) *and* large influence ("hat" value) have the largest Cook's distances.

Note: The last expression results in a `4`; this is the position in the five-element vector `residuals(m.lzn.lcu)[which.ix]` of the maximum positive value. The `129` above it is its label; this is the observation ID from the data frame; the linear model recorded this and associated it with the appropriate residual.

Q14 : *At which observation point is log_{10}Zn most seriously* ***under****-predicted by the model? Does this point have a high value of Zn, i.e., could it be polluted?* *Jump to A14* •

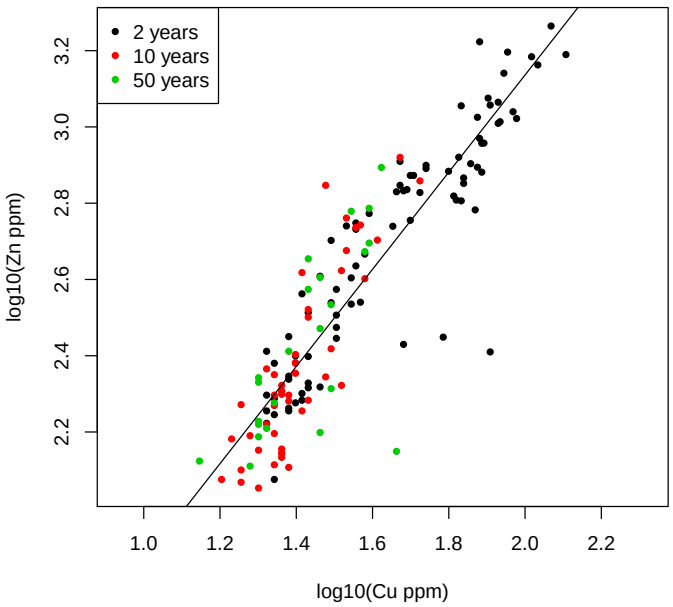
Q15 : *Looking at the "Normal Q-Q" plot, do the residuals appear to be normally distributed?* *Jump to A15* •

Q16 : *Looking at the "Residuals vs. leverage" plot, do the high-leverage residuals have a high Cook's distance?* *Jump to A16* •

Task 20 : Repeat the scatterplot of the relation between log-transformed Zn and Cu, adding the OLS regression line. •

The `abline` function adds a straight line to a scatterplot; if its argument is a linear model object, the line is the best-fit line from the model. To show the true relation between the two, we use the optional `asp` argument. To enhance understanding, we use the optional `col` argument to colour the points according to their flood frequency class; we also specify a printing character with the optional `pch` argument. We also use the `legend` function to add a legend showing the flood frequency classes (see `?meuse` for explanation of the classes).

```
> plot(meuse$logZn ~ meuse$logCu, asp = 1, col = meuse$ffreq,
      pch = 20, xlab = "log10(Cu ppm)", ylab = "log10(Zn ppm)")
> abline(m.lzn.lcu)
> legend("topleft", legend = c("2 years", "10 years",
      "50 years"), pch = 20, col = 1:3)
```



Q17 : *What do you conclude about the use of a simple linear regression to predict log_{10}Zn content of these topsoils from their log_{10}Cu content?* *Jump to A17* •

7.3 Continuous response, categorical predictor

The linear model can also be applied to **categorical**, also called **classified**, predictors. The model formulation is the same, but the design matrix now contains information on the class of each observation, rather than on a continuous value. This is done with so-called "dummy" variables or more sophisticated ways to show the contrasts between classes.

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

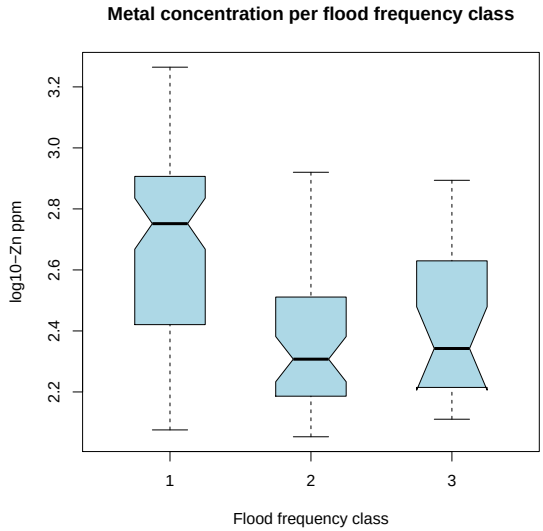
Task 21 : Model the concentration of log_{10}Zn from the flooding frequency. •

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_{10}Zn in each class using the `boxplot` function:

```
> boxplot(meuse$logZn ~ meuse$ffreq, xlab="Flood frequency class",
          ylab="log10-Zn ppm",
          main="Metal concentration per flood frequency class",
          boxwex=0.5, col="lightblue", notch=TRUE)
```



This example shows how **optional function arguments** (here, to the `boxplot` function) can be used to enhance a plot. The `notch` argument can be specified to show the approximate confidence interval of the median (the heavy horizontal line in the boxplot). For flood frequency class 3 the lower limit of this interval is below the first quartile (25%), i.e., the bottom of the box, so that the `boxplot` function shows a warning message on the console: `...some notches went outside hinges ('box')`.

Q18 : *Describe the relation between the different flood frequencies and the metal concentration.* *Jump to A18* •

Third, build a **linear model**, using the `lm` function; note the use of the `~` formula operator to indicate **functional dependence** of the left-hand side (here, `logZn`) on the right-hand side (here, `ffreq`):

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

Task 22 : View the model summary. •

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

Call:
lm(formula = logZn ~ 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
```

Here we see the same summary as for the continuous predictor model (§7.2) except that the coefficients are handled somewhat differently. In a model with categorical predictors the `(Intercept)` coefficient is the predicted mean value for the **first** class in the list, here Flood Frequency Class 1, and then the others which are the **differences** in predicted mean value for the other classes compared to the first;

Q19 : *How much of the total variation in metal concentration is explained by the flooding frequency?* *Jump to A19* •

Q20 : *What is the modelled mean log concentration of the metal in each class?* *Jump to A20* •

Clearly, this prediction is not so good. So, we turn to **spatial** analysis (below, §8), and later to **mixed predictors** (§11).

7.4 * Multivariate linear models

In this **optional** section we show how to model a continuous variable from several predictors, i.e., a *multivariate* model.

We saw in the previous sections that flooding frequency (§7.3) and Cu concentration (§7.2) both can help in predicting Zn concentration; can a combination do better?

7.4.1 Additive linear model

additive model

The simplest way to consider two or more predictive variables is the **additive** model, which assumes that the variables act **linearly** and **independently**.

Task 23 : Build an additive model of the concentration of $\log_{10}$Zn from the flooding frequency and $\log_{10}$Cu. •

We specify the two predictors on the right-hand side of the model formula, separated by the `+` formula operator, which is a symbolic way to specify an additive model.

```
> m.lzn.ff.lcu <- lm(logZn ~ ffreq + logCu, data = meuse)
> summary(m.lzn.ff.lcu)

Call:
lm(formula = logZn ~ ffreq + logCu, data = meuse)

Residuals:
    Min      1Q  Median      3Q     Max
-0.6093 -0.0789  0.0004  0.0895  0.3877

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.5961     0.1026    5.81  3.6e-08 ***
ffreq2       -0.0122     0.0296   -0.41     0.68
ffreq3        0.0181     0.0357    0.51     0.61
logCu         1.2697     0.0612   20.73  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.14 on 151 degrees of freedom
Multiple R-squared:  0.806,        Adjusted R-squared:  0.802
F-statistic:  209 on 3 and 151 DF,  p-value: <2e-16
```

Q21 : *How much of the variance is explained? How does this compare with the two single-factor models?* *Jump to A21* •

We prefer a simpler ("parsimonious") model if possible, because it's easier to interpret and requires less input data. So we want to know if the increase in variance explained with the mixed model is significantly better than that explained by the best single model. To answer this, we compare the two models with a hierarchical analysis of variance (ANOVA) using the `anova` function:

```
> anova(m.lzn.ff.lcu, m.lzn.lcu)

Analysis of Variance Table

Model 1: logZn ~ ffreq + logCu
Model 2: logZn ~ logCu
  Res.Df  RSS Df Sum of Sq    F Pr(>F)
1    151 2.94
2    153 2.95 -2   -0.0144 0.37   0.69
```

Q22 : *How many degrees of freedom are lost by adding flooding frequency to the model using only Cu concentration? What is the decrease is residual sum of squares? If we reject the null hypothesis of no improvement, what is the probability that we are making a Type I error, i.e., falsely rejecting the null hypothesis?* *Jump to A22* •

Clearly the additive model is no improvement; i.e., knowing flooding frequency does not add to our ability to model Zn, if we have the Cu concen-

tration.

7.4.2 Interaction linear model

interaction model

The additive model implies that the slope of the $\log_{10}$Zn vs. $\log_{10}$Cu is the same in all three flooding frequency zones; this may not be the case. To investigate this, an **interaction** model is needed. This is still linear but also allows a cross-term, in this case different slopes of $\log_{10}$Zn vs. $\log_{10}$Cu in each of the the three flooding frequency zones.

Task 24 : Build an interaction model of the concentration of $\log_{10}$Zn from the flooding frequency and $\log_{10}$Cu. Summarize the model and compare it to the single-factor model with ANOVA. •

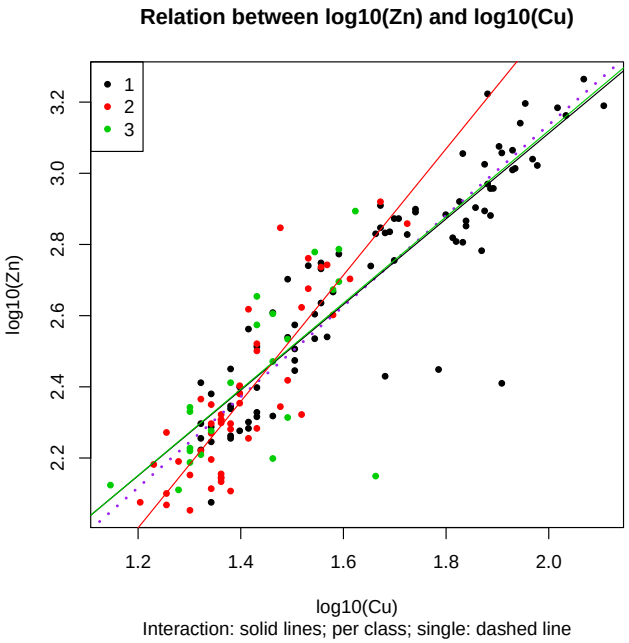
We specify the two predictors on the right-hand side of the model formula, separated by the * formula operator, which is a symbolic way to specify an interaction model.

```
> m.lzn.ff.lcu.i <- lm(logZn ~ ffreq * logCu, data = meuse)
> summary(m.lzn.ff.lcu.i)

Call:
lm(formula = logZn ~ ffreq * logCu, data = meuse)

Residuals:
    Min      1Q  Median      3Q     Max
-0.5924 -0.0701  0.0099  0.0842  0.3513

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.70679    0.11124    6.35  2.4e-09 ***
ffreq2       -0.83351    0.27166   -3.07   0.0026 **
ffreq3       -0.00874    0.32882   -0.03   0.9788
logCu         1.20283    0.06654   18.08  < 2e-16 ***
ffreq2:logCu  0.57249    0.18799    3.05   0.0027 **
ffreq3:logCu  0.00797    0.22608    0.04   0.9719
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.136 on 149 degrees of freedom
Multiple R-squared:  0.817,        Adjusted R-squared:  0.811
F-statistic:  133 on 5 and 149 DF,  p-value: <2e-16

> anova(m.lzn.ff.lcu.i, m.lzn.lcu)

Analysis of Variance Table

Model 1: logZn ~ ffreq * logCu
Model 2: logZn ~ logCu
  Res.Df  RSS Df Sum of Sq    F Pr(>F)
1    149 2.77
2    153 2.95 -4    -0.188 2.53  0.043 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Q23 : *How much of the variance is explained by this model? Would we have a significant risk of committing a Type I error if we use this model rather than one of the simpler ones?* *Jump to A23* •

We can visualize the difference between the interaction and single-factor model by constructing a scatterplot of the two metals, along with the single slope and slopes for each flooding frequency class, separately.

```
> with(meuse, plot(logZn ~ logCu, col=ffreq, pch = 20,
                   xlab = "log10(Cu)", ylab = "log10(Zn)"))
> legend("topleft", legend=levels(meuse$ffreq), pch=20,
         col=1:3)
> title(main = "Relation between log10(Zn) and log10(Cu)")
> title(sub =
            "Interaction: solid lines; per class; single: dashed line")
> abline(lm(logZn ~ logCu, data = meuse),col = "purple",
         lty=3, lwd=2.5)
> abline(lm(logZn ~ logCu, data = meuse,
            subset=(meuse$ffreq==1)),col = 1)
> abline(lm(logZn ~ logCu, data = meuse,
            subset=(meuse$ffreq==2)),col = 2)
> abline(lm(logZn ~ logCu, data = meuse,
            subset=(meuse$ffreq==3)),col = 3)
```



Q24 : *In which flooding frequency class is the relation between $log_{10}Zn$ and $log_{10}Cu$ different from the overall relation?* *Jump to A24* •

We return to this theme in §11.4, when we use a multivariate model as part of kriging with external drift.

7.5 * Regression trees

In this **optional** section we investigate regression methods for continuous predictands that make no assumptions about linearity in their relation with predictors. These methods partition the feature space of predictors into a set of "boxes" in multidimensional feature space, defined by threshold values of each predictor. These "boxes" then each have a simple prediction model, in the simplest case just a constant, i.e., a predicted value of the response variable for all combinations of predictor variables in that feature-space "box".

The advantages of this approach are: (1) no assumption that the functional form is the same throughout the range of the predictors, and (2) over-fitting can be avoided by specifying large enough boxes; their optimum size can be calculated by cost-complexity pruning. This is a high-variance, low-bias method.

Hastie *et al.* [11, §9.2] give a thorough explanation of a tree-based regression method known as CART ("Classification and Regression Trees") [2], which we illustrate here with functions from the `rpart` "Recursive Partitioning" package. A simplified explanation of the same material is given in James *et al.* [15, §8.1].

We are here replacing a regression, i.e., predicting a continuous variable, such as $\log_{10}$Zn, from categorical and/or continuous predictors, such as flooding frequency and distance from the river.[9] The procedure is as follows:

1. We first specify a statistical model, as for the linear model, but with no interactions. That is, we just specify the response variable and the possible predictors.
2. We specify a calibration dataset, as for the linear model.
3. The `rpart` function of the `rpart` package then looks for one predictor variable that "best" splits the data into two groups. Intuitively, "best" refers to the maximum reduction in sum of within-group sums of squares in the response variable, compared to its overall sum of squares with no split; this is the same measure as used in Analysis of Variance (ANOVA); symbolically the reduction is $SS_T - (SS_L + SS_R)$, where L, R represent the "left" and "right" branches of the tree.
4. Following the split, this process is applied separately to both subgroups; this continues recursively until the subgroups either reach a minimum size (specified by us) or until no improvement can be made; that is the sum of the within-groups sum of squares can not be further reduced.
5. This model is then pruned, i.e., some branches are combined, by cross-validation, to avoid over-fitting.

[9] The "classification" trees follow a similar logic, but are used to predict a categorical (classified) outcome.

Task 25 : Build a regression tree to model $\log_{10}$Zn from the flooding frequency, distance to river, and relative elevation. •

Q25 : *What theory of the origin of the Zn is this model testing?* *Jump to A25* •

We first load the library, and a supplementary library for nice plots:

```
> library(rpart)
> library(rpart.plot)
```

We now build the model. The `rpart` function has several control options; in particular we specify the minimum number of observations which can be considered for a split (using the `minsplit` argument), and the minimum value of a complexity parameter (using the `cp` argument); this corresponds to the improvement in R^2 with each split. A small complexity parameter (close to 0) grows a larger tree, which may be over-fitting. For illustration, we set these to allow maximum splitting: split even if only two cases, using the `minsplit` optional argument. Also specify a small complexity parameter with the `cp` optional argument: keep splitting until there is less than 0.3% improvement in (unadjusted) R^2.

The model formulation is the same as for linear modelling: specify the predictand (dependent variable) on the left side of the ~ formula operator and the predictors on the right side, separated by the + formula operator. Note there is no interaction possible in tree models; the predictors are considered separately when determining which to use for a split.

```
> m.lzn.rp <- rpart(logZn ~ ffreq + dist.m + elev,
      data = meuse, minsplit = 2, cp = 0.003)
```

Task 26 : Examine the resulting tree and how it was built. •

We first print the model result:

```
> print(m.lzn.rp)

n= 155

node), split, n, deviance, yval
      * denotes terminal node

   1) root 155 1.5136e+01 2.5562
     2) dist.m>=145 101 4.6521e+00 2.3886
       4) elev>=6.943 93 2.8355e+00 2.3500
         8) dist.m>=230 78 1.7358e+00 2.3117
          16) elev>=9.028 29 3.6572e-01 2.2185
            32) dist.m>=670 8 7.8427e-03 2.1074 *
            33) dist.m< 670 21 2.2142e-01 2.2608
              66) elev>=9.5415 9 7.2874e-02 2.1872 *
              67) elev< 9.5415 12 6.3106e-02 2.3161 *
          17) elev< 9.028 49 9.6898e-01 2.3669
```

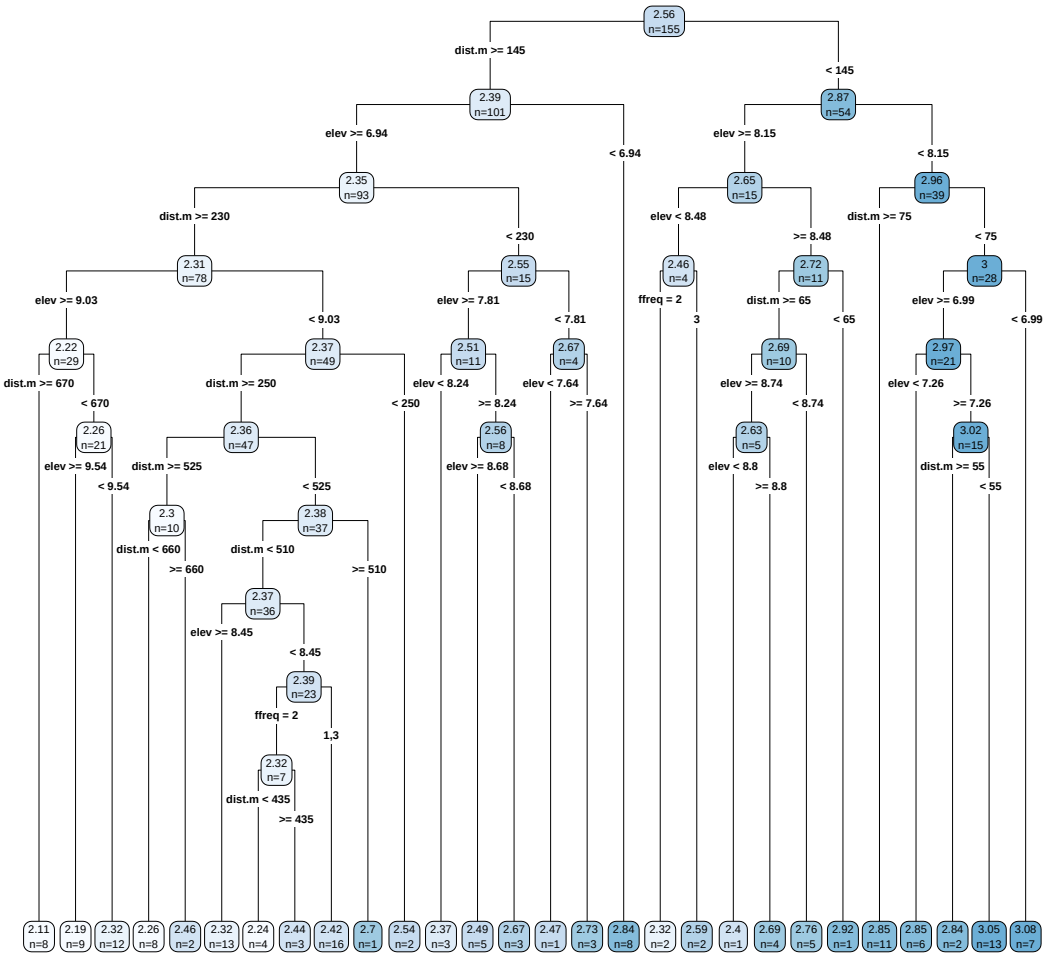
```
        34) dist.m>=250 47 9.0671e-01 2.3595
          68) dist.m>=525 10 1.4000e-01 2.3022
           136) dist.m< 660 8 5.1662e-02 2.2627 *
           137) dist.m>=660 2 2.5919e-02 2.4602 *
          69) dist.m< 525 37 7.2492e-01 2.3750
           138) dist.m< 510 36 6.1938e-01 2.3661
             276) elev>=8.4485 13 1.1079e-01 2.3194 *
             277) elev< 8.4485 23 4.6416e-01 2.3925
               554) ffreq=2 7 1.1678e-01 2.3230
                1108) dist.m< 435 4 8.7392e-03 2.2386 *
                1109) dist.m>=435 3 4.1635e-02 2.4355 *
               555) ffreq=1,3 16 2.9872e-01 2.4230 *
           139) dist.m>=510 1 0.0000e+00 2.6955 *
        35) dist.m< 250 2 7.8548e-07 2.5397 *
      9) dist.m< 230 15 3.9264e-01 2.5488
       18) elev>=7.81 11 2.5014e-01 2.5055
         36) elev< 8.236 3 3.7517e-02 2.3703 *
         37) elev>=8.236 8 1.3723e-01 2.5562
           74) elev>=8.683 5 7.5553e-02 2.4900 *
           75) elev< 8.683 3 3.2668e-03 2.6665 *
       19) elev< 7.81 4 6.5291e-02 2.6677
         38) elev< 7.636 1 0.0000e+00 2.4742 *
         39) elev>=7.636 3 1.5352e-02 2.7323 *
    5) elev< 6.943 8 6.4361e-02 2.8377 *
  3) dist.m< 145 54 2.3431e+00 2.8696
    6) elev>=8.1455 15 4.9618e-01 2.6463
     12) elev< 8.48 4 7.5028e-02 2.4565
       24) ffreq=2 2 3.5663e-03 2.3233 *
       25) ffreq=3 2 4.8902e-04 2.5897 *
     13) elev>=8.48 11 2.2452e-01 2.7154
       26) dist.m>=65 10 1.7841e-01 2.6949
         52) elev>=8.7415 5 9.6731e-02 2.6312
          104) elev< 8.797 1 0.0000e+00 2.4031 *
          105) elev>=8.797 4 3.1724e-02 2.6882 *
         53) elev< 8.7415 5 4.1061e-02 2.7586 *
       27) dist.m< 65 1 0.0000e+00 2.9201 *
    7) elev< 8.1455 39 8.1173e-01 2.9555
     14) dist.m>=75 11 8.0012e-02 2.8487 *
     15) dist.m< 75 28 5.5715e-01 2.9974
       30) elev>=6.9915 21 4.3568e-01 2.9709
         60) elev< 7.2635 6 1.8981e-02 2.8475 *
         61) elev>=7.2635 15 2.8873e-01 3.0203
          122) dist.m>=55 2 1.2715e-02 2.8409 *
          123) dist.m< 55 13 2.0178e-01 3.0479 *
       31) elev< 6.9915 7 6.2531e-02 3.0769 *
```

Task 27 : Plot the regression tree, using the `rpart.plot` function of the `rpart.plot` package. •

The `rpart.plot` function has several options to control the display; we choose to show the values of the response variable at the interior nodes as well as at the leaves, and to show the number of observations in each split and leaf. The information is the same as given with a printout of the model

object, but easier to visualize.

```
> rpart.plot(m.lzn.rp, digits = 3, type = 4, extra = 1)
```



Q26 : *How many terminal nodes, i.e., prediction groups, does this tree have? How many internal nodes, i.e., splits?* *Jump to A26* •

We can find this by examining the fitted object; this is described in the help for `?rpart.object`. The `frame` field contains information on the tree. This includes the `var` field:

> "a factor giving the names of the variables used in the split at

each node (leaf nodes are denoted by the level '`<leaf>`'), `n`, the number of observations reaching the node, `...yval`, the fitted value of the response at the node, and `splits`, a two column matrix of left and right split labels for each node.

So we want to count the number of leaves and internal nodes, using the `==` "logical equals" and `!=` "logical not equals" binary operators:

```
> sum(m.lzn.rp$frame$var == "<leaf>")

[1] 28

> sum(m.lzn.rp$frame$var != "<leaf>")

[1] 27
```

The `rpart` function summarizes the relative importance of each predictor, roughly speaking, how often it was used in the model and how much it contributed to the reduction in sum of squares. This information is stored in the `variable.importance` field of the fitted model.

Task 28 : Display the relative variable importance, as a percentage of the total. •

```
> x <- m.lzn.rp$variable.importance
> data.frame(variableImportance = 100 * x/sum(x))

       variableImportance
dist.m            55.5876
elev              38.9996
ffreq              5.4128
```

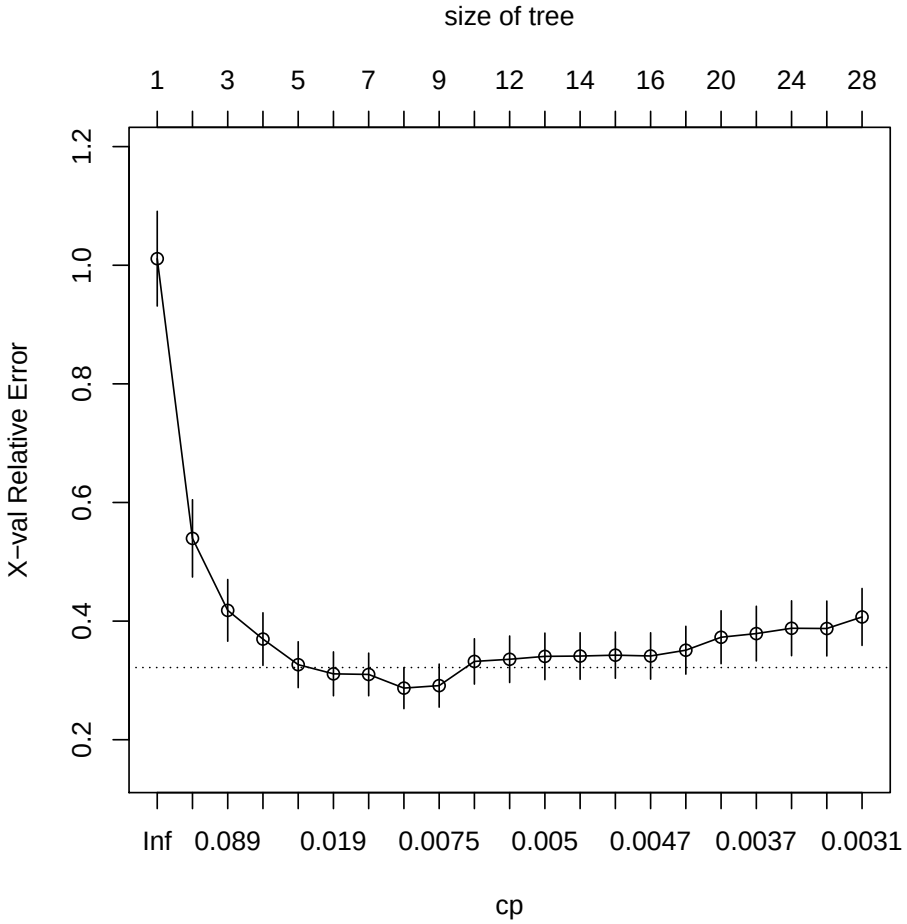
Q27 : *Which variables are most important?* *Jump to A27* •

We now examine the reduction in fitting and cross-validation error with the `printcp` "print the complexity parameter" function.

Task 29 : Print and plot the error rate vs. the complexity parameter and tree size. •

```
> printcp(m.lzn.rp)

Regression tree:
rpart(formula = logZn ~ ffreq + dist.m + elev, data = meuse,
    minsplit = 2, cp = 0.003)

Variables actually used in tree construction:
[1] dist.m elev   ffreq

Root node error: 15.1/155 = 0.0977

n= 155
```

```
        CP nsplit rel error xerror   xstd
1  0.53785      0    1.0000  1.011 0.0799
2  0.11577      1    0.4621  0.539 0.0652
3  0.06839      2    0.3464  0.418 0.0520
4  0.04671      3    0.2780  0.370 0.0443
5  0.02650      4    0.2313  0.327 0.0387
6  0.01299      5    0.2048  0.311 0.0370
7  0.01153      6    0.1918  0.310 0.0360
8  0.00902      7    0.1803  0.287 0.0347
9  0.00617      8    0.1712  0.291 0.0363
10 0.00564     10    0.1589  0.332 0.0382
11 0.00510     11    0.1532  0.336 0.0392
12 0.00498     12    0.1481  0.341 0.0393
13 0.00490     13    0.1432  0.341 0.0393
14 0.00469     14    0.1383  0.343 0.0390
15 0.00462     15    0.1336  0.341 0.0391
16 0.00412     18    0.1197  0.351 0.0403
17 0.00386     19    0.1156  0.373 0.0446
18 0.00351     20    0.1117  0.379 0.0461
19 0.00334     23    0.1012  0.388 0.0463
20 0.00330     26    0.0912  0.388 0.0463
21 0.00300     27    0.0879  0.407 0.0480

> plotcp(m.lzn.rp)
```



Note: Your results will likely be different. This is because the cross-validation makes a *random* split of the full dataset into a number of subsets for model building and evaluation. Each run gives a different random split.

The `xerror` field in the summary shows the **cross-validation error**; that is, applying the model to the original data split K-fold, each time excluding some observations. If the model is over-fitted, the cross-validation error increases; note that the fitting error, given in the `error` field, always decreases. By default, the split is 10-fold; this can be modified by the `control` argument to the `rpart` function.[10]

Q28 : *Does this model appear to be overfit? Why or why not? What appears to be the optimum number of splits, avoiding over-fitting?* *Jump to A28* •

A regression tree can be pruned back to any level of complexity. The aim is to build as complex a tree as possible without over-fitting, i.e., to have the most leaves possible, so the most different predictions.

[10] See the help for `rpart.control`.

Task 30 : Find the minimum cross-validation error and the corresponding complexity parameter. •

This information is in the `cptable` list item inside the `m.lzn.rp` model object returned by the `rpart` function; this is of class `rpart`, which we can see with the `class` function.[11]

```
> class(m.lzn.rp)

[1] "rpart"

> head(cp.table <- m.lzn.rp[["cptable"]])

        CP nsplit rel error  xerror     xstd
1 0.537851      0   1.00000 1.01108 0.079890
2 0.115767      1   0.46215 0.53947 0.065162
3 0.068393      2   0.34638 0.41811 0.052030
4 0.046707      3   0.27799 0.36970 0.044338
5 0.026502      4   0.23128 0.32655 0.038702
6 0.012991      5   0.20478 0.31109 0.037017

> cp.ix <- which.min(cp.table[, "xerror"])
> print(cp.table[cp.ix, ])

       CP    nsplit rel error    xerror      xstd
 0.009015  7.000000  0.180256  0.287056  0.034662

> cp.min <- cp.table[cp.ix, "CP"]
```

Q29 : *What is the minimum cross-validation error? At how many splits? What is the corresponding complexity parameter?* *Jump to A29* •

The minimum cross-validation error is at 0.009; this is one possibility for the complexity parameter for pruning. Another possibility is the first value of the complexity parameter at which the error is one standard deviation above the minimum; this is shown as a dashed line in the plot.

```
> (cp.min.plus.sd <- cp.table[cp.ix, "xerror"] + cp.table[cp.ix,
      "xstd"])

[1] 0.32172

> cp.ix.sd <- min(which(cp.table[, "xerror"] < cp.min.plus.sd))
> print(cp.table[cp.ix.sd, ])

       CP    nsplit rel error    xerror      xstd
 0.012991  5.000000  0.204779  0.311092  0.037017

> cp.min.sd <- cp.table[cp.ix.sd, "CP"]
```

This results in a simpler tree. Another possibility is to find by inspection where a large increase in cross-validation error first occurs; this would result in a more complex tree.

[11] ?rpart

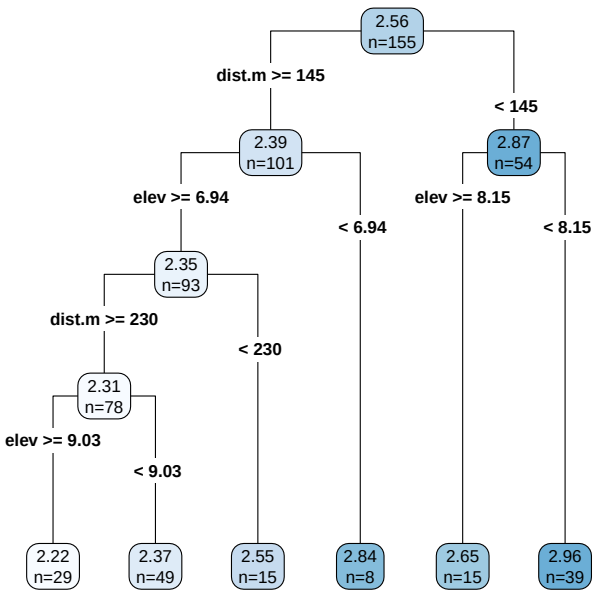
Task 31 : Prune the tree back to complexity level identified in the previous step. •

We do this with the `prune` function, specifying the `cp` "complexity parameter" argument.

```
> (m.lzn.rpp <- prune(m.lzn.rp, cp = cp.min.sd))

n= 155

node), split, n, deviance, yval
      * denotes terminal node

 1) root 155 15.136000 2.5562
   2) dist.m>=145 101  4.652100 2.3886
     4) elev>=6.943 93  2.835500 2.3500
       8) dist.m>=230 78  1.735800 2.3117
        16) elev>=9.028 29  0.365720 2.2185 *
        17) elev< 9.028 49  0.968980 2.3669 *
       9) dist.m< 230 15  0.392640 2.5488 *
     5) elev< 6.943 8  0.064361 2.8377 *
   3) dist.m< 145 54  2.343100 2.8696
     6) elev>=8.1455 15  0.496180 2.6463 *
     7) elev< 8.1455 39  0.811730 2.9555 *
```

Task 32 : Plot the pruned regression tree. •

```
> rpart.plot(m.lzn.rpp, digits = 3, type = 4, extra = 1)
```



Q30 : *How many terminal nodes, i.e., prediction groups, does the pruned*

tree have? How many internal nodes, i.e., splits? How do these compare with the original (unpruned) tree? *Jump to A30* •

Q31 : *Which predictor(s) was(were) used to build the pruned tree? What does this imply about the other(s)?* *Jump to A31* •

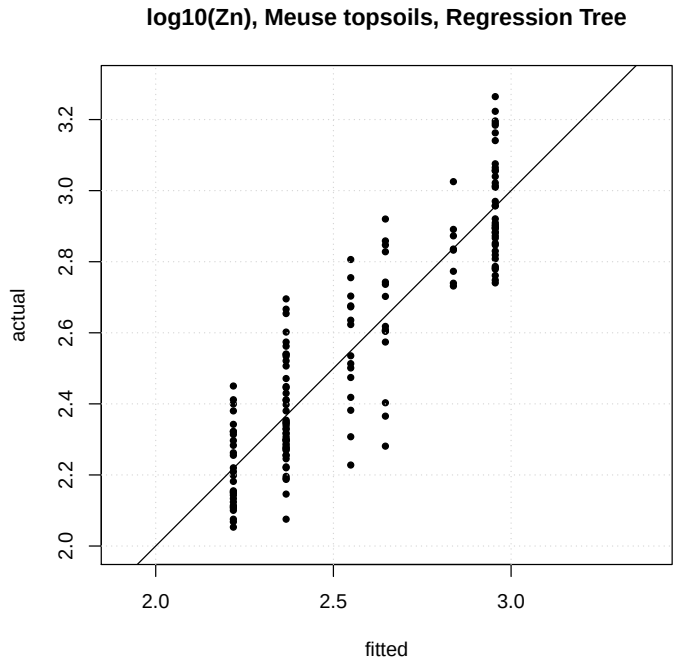
Q32 : *Interpret the structure of the tree in terms of geography.* *Jump to A32* •

A fitted model can be used for prediction. There is a major difference with the linear model: the tree model only predicts the exact values at its leaves, whereas the linear model applies its formula to any predictor value and thus predicts any number of values. Here we do not have other points to predict, so we will just predict back at the known points; these are more properly called **fitted** values.

Task 33 : Use the pruned regression tree to predict at the calibration points, and plot the actual vs. fitted values. •

We do this with the `predict` method applied to a `rpart` object; this automatically calls function `predict.rpart`. The points to predict and the values of the predictor variables at those points are supplied in a dataframe as argument `newdata`. In this case we already have the dataframe, i.e., the `meuse` object. We count the number of predicted values with the `unique` function; there is only one value per "box" in the feature space defined by the predictor variables.

```
> p.rpp <- predict(m.lzn.rpp, newdata = meuse)
> length(unique(p.rpp))

[1] 6

> summary(r.rpp <- meuse$logZn - p.rpp)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.3653 -0.0949 -0.0284  0.0000  0.1009  0.3286

> sqrt(sum(r.rpp^2)/length(r.rpp))

[1] 0.14141

> plot(meuse$logZn ~ p.rpp, asp = 1, pch = 20, xlab = "fitted",
      ylab = "actual", xlim = c(2, 3.3), ylim = c(2,
          3.3), main = "log10(Zn), Meuse topsoils, Regression Tree")
> grid()
> abline(0, 1)
```



Q33 : *Comment on the success or otherwise of this modelling approach.* *Jump to A33* •

Challenge: Build a linear model to predict $\log_{10}$Zn from the flooding frequency, distance to river, and relative elevation, and compare its performance to the regression tree model.

Challenge: Select 90% of the observations (see the `sample` function) and use these to build another regression tree. Compare it to the first one. How much difference do you expect? Do this several times to evaluate the sensitivity of the regression tree method to the data.

7.6 * Classification trees

A very similar method to the regression tree for continuous predictands can be applied for categorical (classified) predictands, in which case it is known as a **classification tree**. Again, see Hastie *et al.* [11, §9.2] or James *et al.* [15, §8.1]. The same `rpart` "Recursive Partitioning" package implements this.

There are several categorical variables in the Meuse dataset; one that would be interesting to model is flooding frequency. This is determined by historical records and time-series of air photos or satellite images, which are often not available during flooding events due to cloud cover. Perhaps flooding frequency class can be determined from the distance to river and elevation above sea level; if so a reliable map of flooding frequency could be produced.

The `rpart` function can also build trees to model categorical response variables. Instead of using reduction in residual error as a criterion (or equiva-

lently increase in R^2) it uses the increase in classification accuracy. Outputs and interpretation are somewhat different from regression trees.

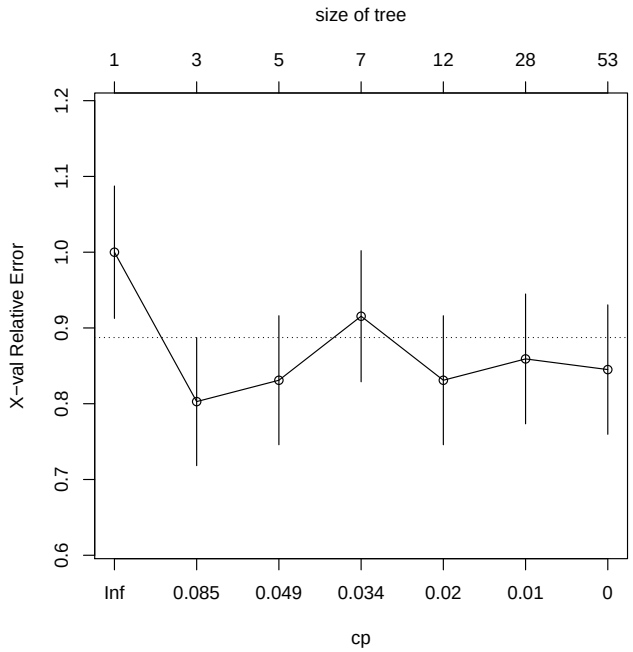
Task 34 : Build a classification tree to model flooding frequency class based on the distance to river and elevation above sea level. •

The model formula has the same form as for a continuous predictand. Since the response variable is categorical `rpart` automatically uses the `"class"` method instead of `"anova"` when determining the splits.

```
> m.ff.rp <- rpart(ffreq ~ dist.m + elev, data = meuse,
      minsplit = 2, cp = 0)
```

Task 35 : Show the cross-validation error rate vs. complexity parameter, both in text and as a graph. Find the complexity parameter at the minimum cross-validation error rate. •

```
> printcp(m.ff.rp)

Classification tree:
rpart(formula = ffreq ~ dist.m + elev, data = meuse, minsplit = 2,
    cp = 0)

Variables actually used in tree construction:
[1] dist.m elev

Root node error: 71/155 = 0.458

n= 155

       CP nsplit rel error xerror   xstd
1 0.12676      0     1.000  1.000 0.0874
2 0.05634      2     0.746  0.803 0.0846
3 0.04225      4     0.634  0.831 0.0851
4 0.02817      6     0.549  0.915 0.0865
5 0.01408     11     0.408  0.831 0.0851
6 0.00704     27     0.183  0.859 0.0857
7 0.00000     52     0.000  0.845 0.0854

> plotcp(m.ff.rp)
> cp.table.class <- m.ff.rp[["cptable"]]
> sum(cp.table.class[, "CP"])

[1] 0.27465
```



Q34 : *Considering the complexity parameter at each level to roughly correspond to the amount of explained variation added by the level: (1) how much total variability is explained by the full model?; (2) at how many splits is the cross-validation error a minimum? (3) Is there a clear choice of complexity parameter for pruning?* *Jump to A34*

•

Task 36 : Prune the tree. •

Because there is not much difference in cross-validation errors between two and three splits, we choose three splits to see a somewhat more complex tree.

```
> (m.ff.rpp <- prune(m.ff.rp, cp = cp.table.class[3,
      "CP"]))

n= 155

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 155 71 1 (0.541935 0.309677 0.148387)
   2) elev< 7.558 40  2 1 (0.950000 0.000000 0.050000) *
   3) elev>=7.558 115 67 2 (0.400000 0.417391 0.182609)
     6) elev< 9.084 87 43 1 (0.505747 0.321839 0.172414)
      12) elev>=8.812 17  3 1 (0.823529 0.117647 0.058824) *
      13) elev< 8.812 70 40 1 (0.428571 0.371429 0.200000)
        26) elev< 8.3805 45 22 1 (0.511111 0.244444 0.244444) *
        27) elev>=8.3805 25 10 2 (0.280000 0.600000 0.120000) *
     7) elev>=9.084 28  8 2 (0.071429 0.714286 0.214286) *
```

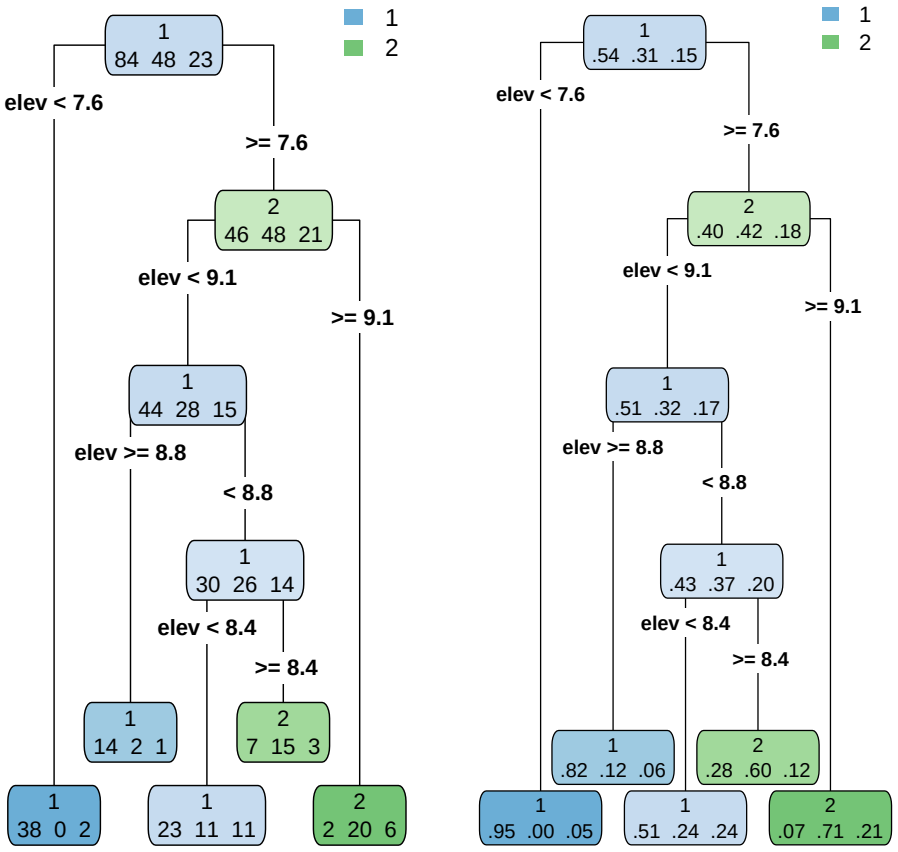
```
> printcp(m.ff.rpp)

Classification tree:
rpart(formula = ffreq ~ dist.m + elev, data = meuse, minsplit = 2,
    cp = 0)

Variables actually used in tree construction:
[1] elev

Root node error: 71/155 = 0.458

n= 155

      CP nsplit rel error xerror   xstd
1 0.1268      0     1.000  1.000 0.0874
2 0.0563      2     0.746  0.803 0.0846
3 0.0423      4     0.634  0.831 0.0851
```

The nodes and leaves of the tree now show the proportion of each class which satisfies the condition at the node. For example, at the first split (`elev < 7.558`, 95% of the observations are class 1, none are class 2, and one is class 3[12]. For the higher-elevation points, 40% are class 1, 41.7% class 1, 18.2% class 3.

Task 37 : Display the classification tree as a graph (1) showing the number of observations at each node and leaf; (2) showing the proportion of each class in each node and leaf. •

```
> par(mfrow = c(1, 2))
> rpart.plot(m.ff.rpp, type = 4, extra = 1)
> rpart.plot(m.ff.rpp, type = 4, extra = 4)
> par(mfrow = c(1, 1))
```

[12] We may wonder if that point's flooding frequency was accurately recorded.



Q35 : *How successful is this tree in modelling flooding frequency? Explain why.* *Jump to A35* •

7.7 * Random forests for continuous predictands

In this **optional** section we discuss random forests for continuous predictands; this is an extension of the regression trees explained in §7.5.

A problem with regression trees is that a small change in the sample set, for example a missing or erroneous observation, can radically change the tree.

> "One major **problem** with trees is their **high variance**. Often a **small** change in the data can result in a **very different** series of splits, making interpretation somewhat precarious. The major reason for this instability is the **hierarchical** nature of the process: the effect of an error in the top split is **propagated** down to all of the splits below it." [11, p. 312] (emphasis added)

Also, correlated predictors can appear in the tree as surrogates for each other, depending on the details of the calibration set; this makes interpretation difficult. Finally, as we have seen in the 1:1 plot of actual vs. fitted values in

the previous section, trees do not give smooth predictions; rather they only predict a single value in each box. To solve these problems, a method known as "random forests" was developed; see Hastie *et al.* [11, §15] (advanced) or James *et al.* [15, §8] (simplified). There are a lot of details to this method, but the basic idea is straightforward:

1. Build a **large number of regression trees**, independently, using *different* sets of observations; these are built by *sampling with replacement* from the actual observations; this is a technique known as *bagging* or *bootstrap aggregation*.
2. Save all these trees; when predicting, use all of them and **average their predictions**.
3. In addition we can summarize the whole set of trees to see how different they are, thus how robust is the final model.
4. Also, for each tree we can use observations that were not used to construct it for true **validation**, called *out-of-bag* validation. This gives a good idea of the true prediction error.

The first step may seem suspicious. The underlying idea is that what we observe is the best sample we have of reality; if we sampled again we'd expect to get similar values. So we simulate re-sampling by sampling from this same set, *with replacement*, to get a sample of the same size but a different sample. If this idea bothers you, read Efron & Gong [7], Shalizi [24] or Hastie *et al.* [11, §8.2].

In this section we compare the random forest to the single regression tree of the previous section.

Task 38 : Build a random forest to model $\log_{10}$Zn from the flooding frequency, distance to river, and relative elevation. •

Random forests have been implemented with the `randomForest` package. The `randomForest` function computes the model; `print` applied to an object of class `randomForest` displays the results.

The `randomForest` function has many control parameters, which you should review[13] before any serious model building, and also consult the recommended textbooks for advice on the effect of various decisions about the parameters. Here we just replace the default number of predictors to try at each split, specifying that all three should be compared, using the `mtry` argument.

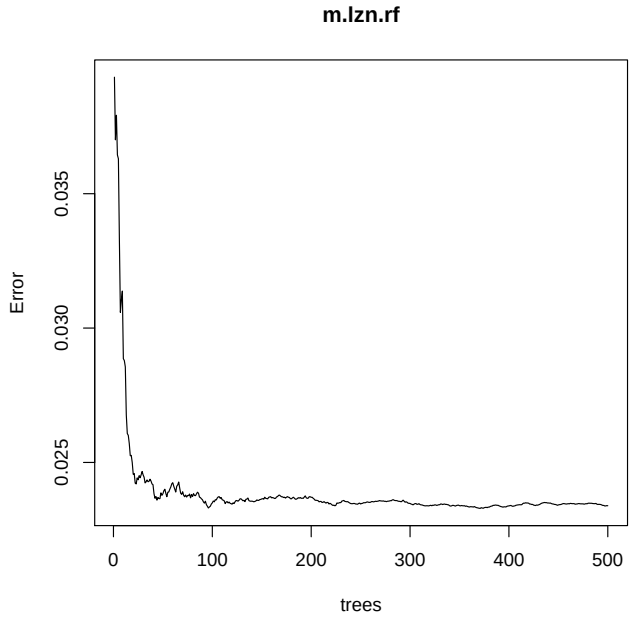
```
> library(randomForest)
> m.lzn.rf <- randomForest(logZn ~ ffreq + dist.m +
      elev, data = meuse, importance = T, na.action = na.omit,
      mtry = 3)
> print(m.lzn.rf)
```

[13] `?randomForest`

```
Call:
 randomForest(formula = logZn ~ ffreq + dist.m + elev, data = meuse,      importa
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 3

          Mean of squared residuals: 0.023388
                    % Var explained: 76.05
```

We plot the the error rates, i.e., mean square error of the predictions of the out-of-bag observations, of the randomForest object vs. the number of trees. Here the `plot` function is applied to a `randomForest` object.

```
> plot(m.lzn.rf)
```



Q36 : *About how many trees are needed to achieve a stable error rate?*
Jump to A36 •

While building the forest, if the optional `importance` argument is set to `TRUE`, we obtain the average importance of each predictor; this is similar to the regression tree but here the importances are averaged.

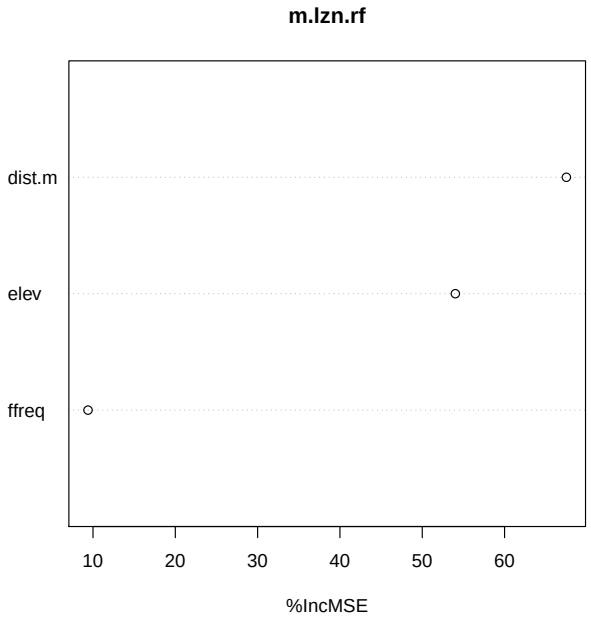
The importance is measured as an increase in mean squared error (MSE) as follows. For each tree, the prediction error on the out-of-bag observations is recorded – this is the success of tree. Then a predictor variable is permuted: that is, its values are randomly assigned to observations. The tree is again used to predict at the out-of-bag observations, and the errors are recorded. If a variable is not used much in the tree, or at lower levels, or with not much difference in the predictions, this increase in error will be small – the predictor is unimportant. The difference between the errors from the original and permuted trees are averaged over all trees and then normalized by the

standard deviation of the differences, to give an importance on a $[0 \ldots 1]$ or $[0 \ldots 100]\%$ scale.

Task 39 : Print and display the relative variable importances, as measured by the increase in mean-squared error. •

We do this with the `importance` and `varImpPlot` functions.

```
> importance(m.lzn.rf, type = 1)

       %IncMSE
ffreq   9.3959
dist.m 67.5104
elev   54.0195

> varImpPlot(m.lzn.rf, type = 1)
```



Note that the importances are calculated independently, they do not sum to 100%.

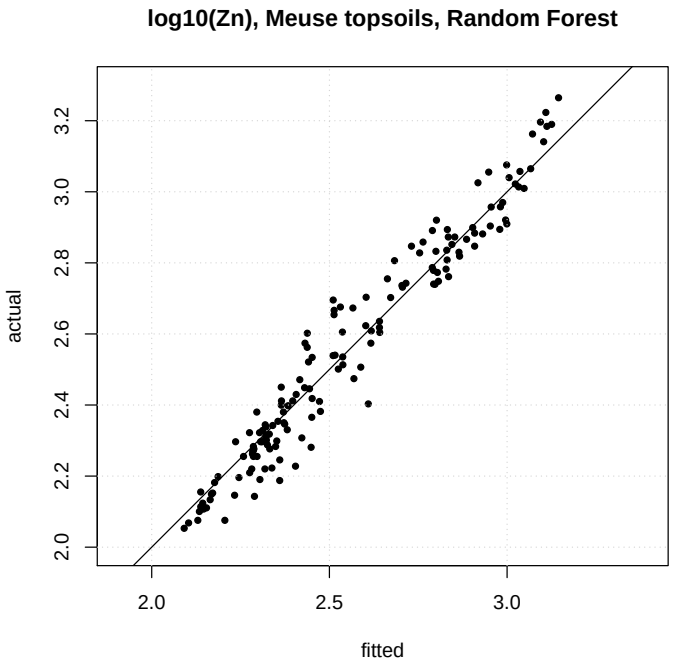
Q37 : *Which variables are most important? Compare with the single pruned regression tree.* *Jump to A37* •

And of course we want to use our model for prediction, here at the known points:

Task 40 : Use the random forest object to predict back at the calibration points, and plot the actual vs. fitted values. •

We do this with the `predict` method applied to a `randomForest` object; this automatically calls function `predict.randomForest`. The points to predict and the values of the predictor variables at those points are supplied in a dataframe with the optional argument `newdata`. In this case we already have the dataframe, i.e., the `meuse` object.

```
> p.rf <- predict(m.lzn.rf, newdata = meuse)
> length(unique(p.rf))

[1] 155

> summary(r.rpp <- meuse$logZn - p.rf)

    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.20630 -0.04188 -0.00928 -0.00134  0.03346  0.18485

> sqrt(sum(r.rpp^2)/length(r.rpp))

[1] 0.071266

> plot(meuse$logZn ~ p.rf, asp = 1, pch = 20, xlab = "fitted",
      ylab = "actual", xlim = c(2, 3.3), ylim = c(2,
          3.3), main = "log10(Zn), Meuse topsoils, Random Forest")
> grid()
> abline(0, 1)
```



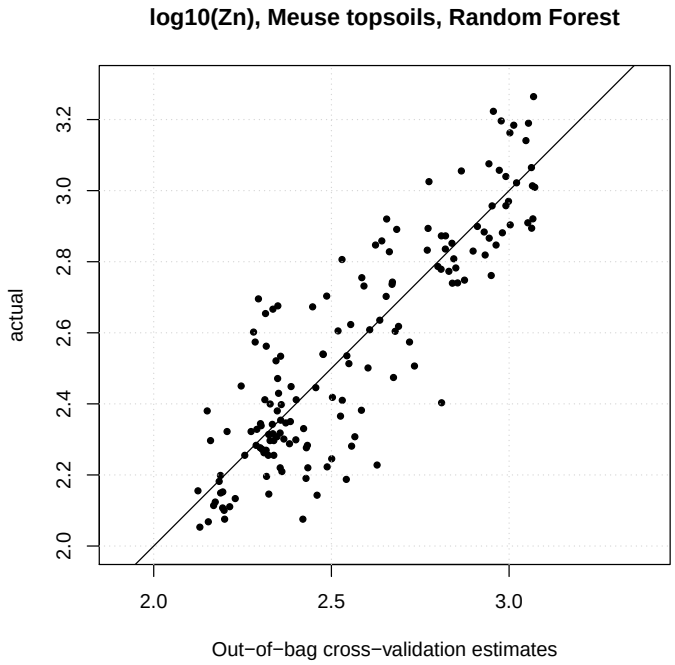
Q38 : *Compare this 1:1 plot (actual vs. fitted) to that produced by the regression tree. Which would you prefer for prediction at unknown points?*
Jump to A38 •

This graph is not a valid estimate of the accuracy of the random forest. Better is the average of the **out-of-bag** cross-validations. These are predictions at a point when it is not included in the resampled set of observations used to build a tree. We can get these by using the `predict` method applied to a `randomForest` object but with no `newdata` argument; in this case the method averages the out-of-bag cross-validations calculated during the construction of the forest.

Task 41 : Display the out-of-bag cross-validation averages vs. the known values at each observation point. Compare the evaluation statistics with those from the fitted values, above. •

Again we use the `predict` method applied to a `randomForest` object; this automatically calls function `predict.randomForest`. But now we do *not* specify a dataframe with the `newdata` optional argument. In this case `predict.randomForest` reports the average out-of-bag predictions that were stored during the computation of the forest.

```
> p.rf.oob <- predict(m.lzn.rf)
> summary(r.rpp.oob <- meuse$logZn - p.rf.oob)

    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.40695 -0.09680 -0.01934 -0.00349  0.07524  0.40084

> sqrt(sum(r.rpp.oob^2)/length(r.rpp.oob))

[1] 0.15293

> plot(meuse$logZn ~ p.rf.oob, asp=1, pch=20,
       xlab="Out-of-bag cross-validation estimates",
       ylab="actual", xlim=c(2,3.3), ylim=c(2,3.3),
       main="log10(Zn), Meuse topsoils, Random Forest")
> grid()
> abline(0,1)
```



Q39 : *How do the evaluation statistics from the out-of-bag estimates compare to those from the fits? Which is a more realistic measure of the success of this modelling approach?* *Jump to A39* •

Challenge: Build another random forest and compare its predictions and variable importances to the first one. How much difference do you expect?

7.8 * Random forests for categorical predictands

In this **optional** section we discuss random forest models for categorical (classified) response variables; this is an extension of the classification trees discussed in §7.6. The output is somewhat different that for continuous predictands.

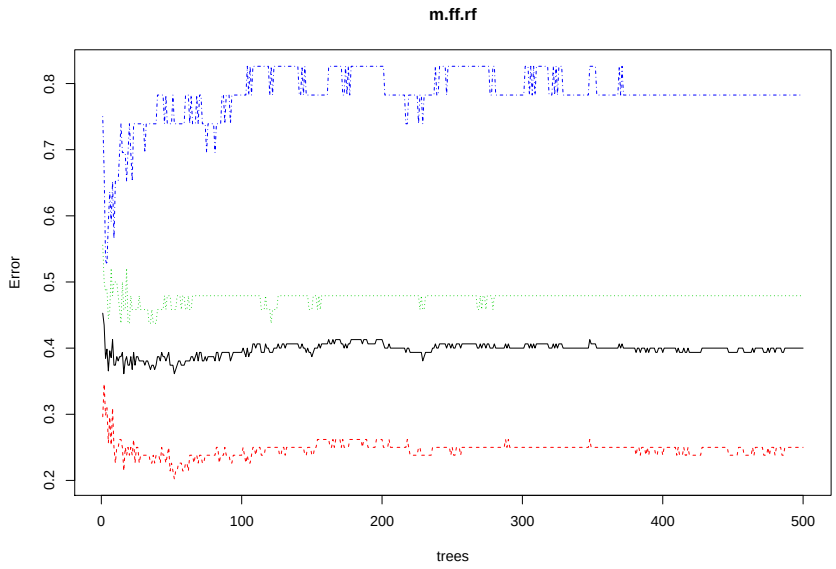
Task 42 : Build a random forest model of flooding frequency class based on distance to river and elevation, using the default number of trees. •

```
> library(randomForest)
> m.ff.rf <- randomForest(ffreq ~ dist.m + elev, data = meuse,
      importance = T, mtry = 2)
> print(m.ff.rf)

Call:
 randomForest(formula = ffreq ~ dist.m + elev, data = meuse, importance = T,
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2
```

```
        OOB estimate of  error rate: 40%
Confusion matrix:
   1  2 3 class.error
1 63 13 8     0.25000
2 18 25 5     0.47917
3 12  6 5     0.78261

> importance(m.ff.rf, type = 1)

       MeanDecreaseAccuracy
dist.m               3.1595
elev                46.9759
```

Task 43 : Display the out-of-bag cross-validation error rate vs. number of trees. •

```
> plot(m.ff.rf)
> palette()

[1] "black"   "red"     "green3"  "blue"    "cyan"    "magenta"
[7] "yellow"  "gray"
```



This shows the out-of-bag cross-validation for overall (black lines) and the three classes separately; these colours are according to the default colour palette for base graphics, which we can examine with the `palette` function.

Q40 : *Which classes have the highest and lowest error rates? Comment on their values.* *Jump to A40* •

Q41 : *Approximately how many trees are needed for a stable error rate?* *Jump to A41* •

Task 44 : Display the out-of-bag predictions of the random forest, compared to the mapped classes in a confusion matrix. •

The `table` function with two arguments produces a **confusion matrix**, also called a **contingency table**. The conventional order is for the "true" (observed) values to be columns (second argument) and the predicted values to be rows (first argument).

```
> p.rf <- predict(m.ff.rf)
> table(meuse$ffreq)

 1  2  3
84 48 23

> table(p.rf)

p.rf
 1  2  3
93 44 18

> (p.rf.cm <- table(p.rf, meuse$ffreq, dnn = c("Predicted",
      "Observed")))

         Observed
Predicted  1  2  3
        1 63 18 12
        2 13 25  6
        3  8  5  5
```

We follow Lark [16] and use the terms:

- **map unit purity** for what is usually called "user's accuracy"; this is the proportion of a map unit correctly shown by the model;
- **class representation** for what is usually called "producer's accuracy"; this is how well the mapper found the class when it was actually present.
- **overall purity** for what is often called "overall accuracy".

In this matrix the proportional row sums are map unit purity, i.e., how much of the mapped class is really in that class; this is what we want to evaluate the usefulness of a model. The map user would go to the field expecting to find a class; how often would the class as mapped be found in the field, and how often would other classes be found instead? Similarly the proportional column sums are class representations, i.e., how much of the actual class is mapped correctly, and how much as other classes?

The `diag` "matrix diagonal" function extracts the diagonal, i.e., number of correctly-classified observations. The `apply` function, applied over rows with the `MARGIN` argument set to `1`, specifying `sum` as the function to apply, sums each row, i.e., class as mapped. Their ratio gives the map unit purity. For example, 5 of the total 18 observations predicted to be in flood frequency class 3 were actually in that class. This gives a map unit purity of 27.8%.

```
> (d <- diag(p.rf.cm))

 1  2  3
63 25  5

> (row.sums <- apply(p.rf.cm, MARGIN = 1, "sum"))

 1  2  3
93 44 18

> (mu.purity <- d/row.sums)

      1       2       3
0.67742 0.56818 0.27778
```

Q42 : *(1) What are the three map unit purities? (2) Which confusions contribute most to the low purities? (3) Overall, how well does this random forest model predict flood frequency class from elevation and distance to river?* *Jump to A42* •

8 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 45 : 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 vector of the two names, here `"x"` and `"y"`; the `coordinates` method expects such a vector, to know which fields represent the coordinates.

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

Task 46 : Display the structure of the spatially-explicit object. •

```
> str(meuse)

Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
  ..@ data       :'data.frame':        155 obs. of  14 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 ...
  .. ..$ logZn  : num [1:155] 3.01 3.06 2.81 2.41 2.43 ...
  .. ..$ logCu  : num [1:155] 1.93 1.91 1.83 1.91 1.68 ...
  ..@ coords.nrs : int [1:2] 1 2
  ..@ coords     : num [1:155, 1:2] 181072 181025 181165 181298 181307 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:155] "1" "2" "3" "4" ...
  .. .. ..$ : 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 slot
  .. .. ..@ projargs: chr NA
```

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

Q43 : *Which slots in the object refer to the spatial structure and which to the attributes?* *Jump to A43* •

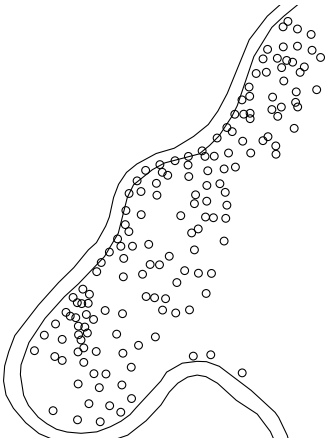
> **Note:** The `@proj4string` slot has no projection information, so we don't know what the coördinates represent. That information can be added, but for now we just consider them as meters on an arbitrary grid. In fact, these coördinates are in the Dutch national triangulation ("Rijksdriehoek", or RDH) system[14].

Task 47 : Display a map of the sample locations, along with the line of the Meuse river. •

We use the `data` function to load another built-in dataset, `meuse.riv`, which is a `matrix` of point-pairs outlining both banks of the river:

[14] See the EPSG database http://www.epsg-registry.org/, EPSG code 28992

```
> 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[15]. The `asp` optional argument specifies the plot aspect, i.e., ratio of the two scales. Since this is a map, the two scales are equal, so we specify `asp=1`.

Q44 : *How are the points distributed over the study area? For example, are they evenly-spaced (gridded)? Random? Denser in some areas?* *Jump to A44* •

Task 48 : 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)
```

[15] You can see all the possible symbols with `?pch`

Q45 : *Do nearby observations seem to more similar to each other than would be expected at random?* *Jump to A45* •

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:

$$\gamma(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{2}[z(\mathbf{x}_i) - z(\mathbf{x}_j)]^2 \tag{9}$$

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

Task 49 : 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$logZn)
> n * (n - 1)/2

[1] 11935
```

Q46 : *How many unique* pairs *of points are there in this dataset?* *Jump to A46* •

Task 50 : Compute the distance and semivariance between the first two points in the data set. •

The distance can be calculated with the `dist` function, which finds the Euclidean (straight-line) distances between rows of a matrix, in this case the first two rows of the coördinates matrix, selected by the `[]` matrix selection operator. The semivariance is computed as in Equation 9.

```
> dim(coordinates(meuse))

[1] 155   2

> coordinates(meuse)[1, ]

     x      y
181072 333611

> coordinates(meuse)[2, ]

     x      y
181025 333558

> (sep <- dist(coordinates(meuse)[1:2, ]))

       1
2 70.838

> (gamma <- 0.5 * (meuse$logZn[1] - meuse$logZn[2])^2)

[1] 0.0011441
```

Q47 : *What is the separation and semivariance for this point-pair?* *Jump to A47* •

! → 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:

$$\overline{\gamma}(\mathbf{h}) = \frac{1}{2m(\mathbf{h})} \sum_{(i,j)|\mathbf{h}_{ij}\in\mathbf{h}} [z(\mathbf{x}_i) - z(\mathbf{x}_j)]^2 \tag{10}$$

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**.

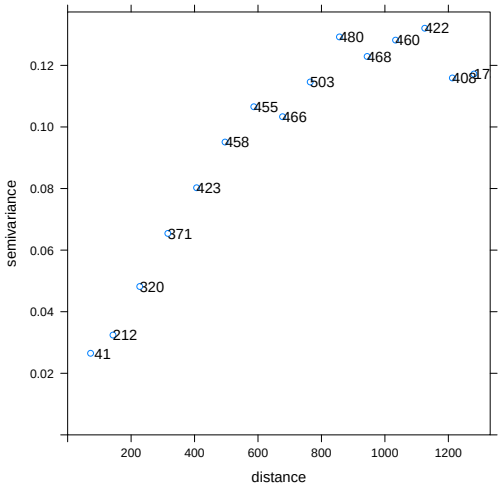
- $\mathbf{h}_{ij}$ is the distance (separation) between points i and j;
- $\mathbf{h}$ is a range of separations, similar to a histogram **bin**; $\mathbf{h}_{ij} \in \mathbf{h}$ means that this point-pair's separation vector is within this range;
- $m(\mathbf{h})$ is the number of point-pairs in the bin corresponding to separation range $\mathbf{h}$;
- the notation $(i, j)|$ reads "pairs of points indexed as i and j, *such that* ...".

Task 51 : Plot the experimental variogram of the log-Zn concentrations, i.e. the average semi-variances of the point-pairs versus average distance (also known as the "lag"), with a bin width of 90 m, to a maximum lag distance ("cutoff") of 1300 m[16]. •

The `variogram` function computes the experimental variogram:

```
> (v <- variogram(logZn ~ 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))
```

[16] By default, the `variogram` function uses a cutoff equal to 1/3 of the maximum distance across the diagonal of the bounding box (see slot `@bbox`), and divides this into 15 equal separation distance classes; this rule-of-thumb may not be the best to reveal the spatial structure. Ideally the cutoff should be about 10–15% beyond the estimated range, and bins should be as narrow as possible before the variogram becomes "too" erratic. I experimented with the `cutoff` and `width` arguments to the `variogram` command according to these criteria, and decided on these values.



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 `logZn ~ 1` specifies the dependence of the left-hand side "dependent" variable, here `logZn`, 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 `logZn` is only dependent on itself! This is why we use the term spatial **auto-** ("self") correlation.

Q48 : *What is the **average separation** and **average semivariance** in the first bin? How many **point-pairs** are averaged for this?* *Jump to A48*
•

Q49 : *What evidence does this plot give that closer points are more similar, i.e., what is the evidence for **local spatial dependence**?* *Jump to A49* •

---

range of spatial dependence

Q50 : *At what separation between point-pairs is there no more spatial dependence? (This is called the **range** of spatial dependence)* *Jump to A50* •

Q51 : *What is the semivariance at zero separation? (This is called the **nugget**). Why is it not zero?* *Jump to A51* •

nugget semivariance

The nugget semivariance is completely unexplained. It may result from measurement error and from processes at a smaller scale than the support

of the observation.

Q52 : *What is the semivariance at the range and beyond? (This is called the **total sill**)* *Jump to A52* •

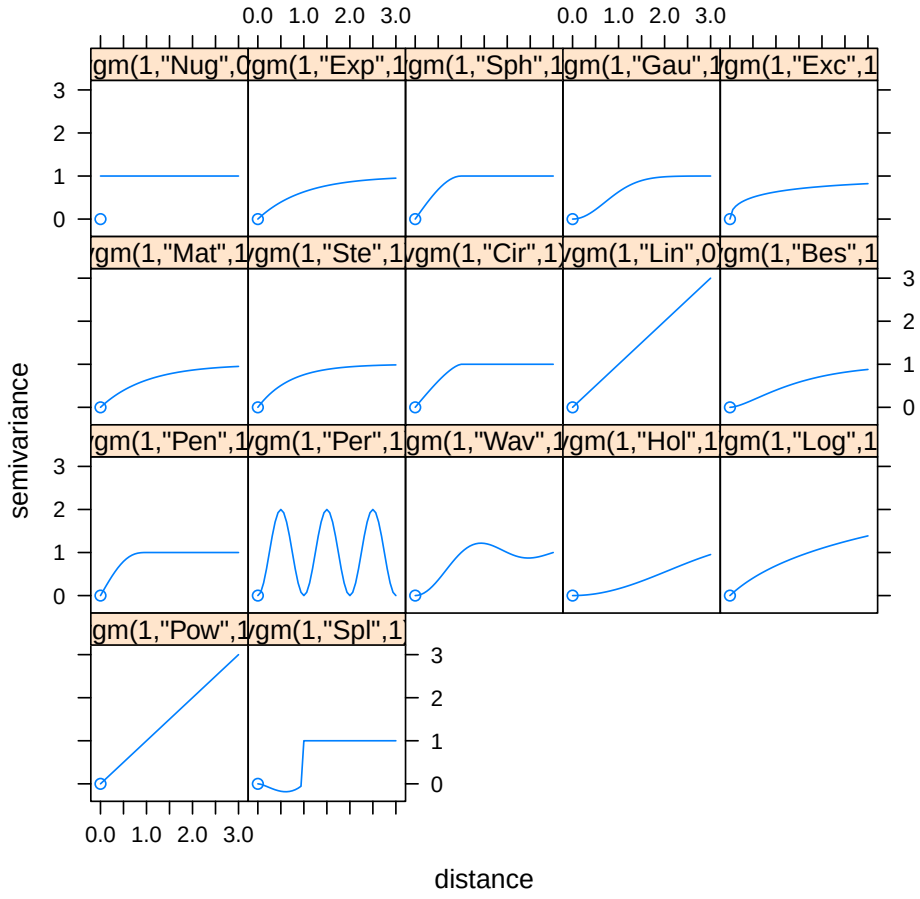
variogram sill

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

Task 52 : Display the variogram model forms which can be used in `gstat`. •

The `show.vgms` function of the `gstat` package displays these:

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



Selecting a model form is an art; the best way is from knowledge of the assumed **spatial process** which produced this realization of the assumed **random field**.

From many studies we know that a common model for soil attributes is the

spherical model. This model reaches a definite **sill** at some **range**; that is, if point-pairs are separated by more than this distance, their covariance is a constant. At shorter separations the semivariance increases almost linearly from zero separation, and then near the range there is a "shoulder" which transitions to the sill. This is consistent with "patchy" spatial random fields with more or less abrupt transitions; such fields are often found to represent the spatial covariance structure of soil properties.

$$\gamma(h) = \begin{cases} c \cdot \left[\frac{3}{2}\frac{h}{a} - \frac{1}{2}\left(\frac{h}{a}\right)^3 \right] & : \quad h < a \\ c & : \quad h \geq a \end{cases} \tag{11}$$

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.

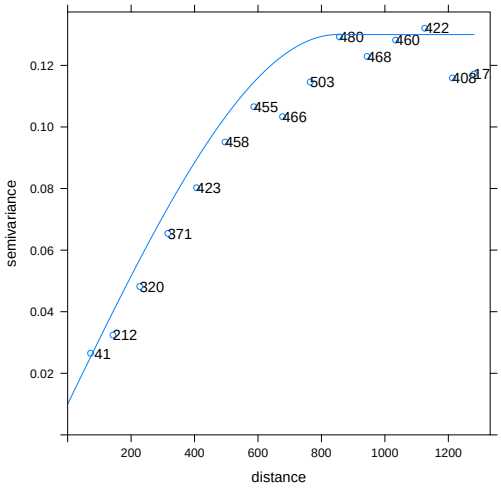
In addition, the whole variogram is raised by a **nugget** variance.

> **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.
>
> A third method is by fitting various models and comparing their goodness-of-fit, typically by the weighted sum of squared errors (discrepency between model and observed at each variogram bin) of the fitted model. This ignores prior information about the model form.

Task 53 : Fit a spherical variogram model by eye to the experimental semivariogram and plot it; then adjust it with `gstat` automatic fit (`fit.variogram` function). •

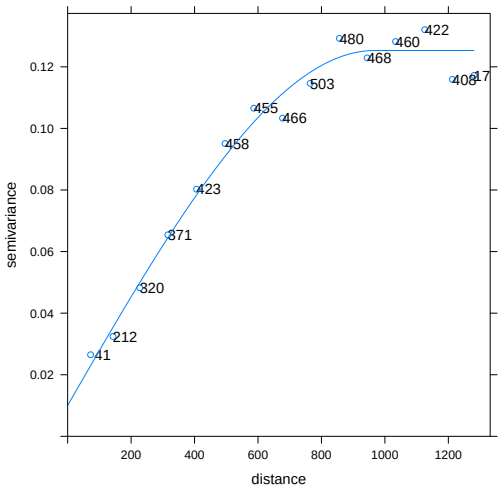
The `vgm` function specifies a variogram model. In the previous set of questions we estimated the these parameters from looking at the empirical variogram, so we supply them as the model parameters. Note that the `psill` "partial sill" model parameter is the total sill (which we estimated as 0.13), less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



Q53 : *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 A53* •

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

9 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**.

9.1 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.

Here is the Ordinary Kriging system, in matrix form:

$$\mathbf{A}\lambda = \mathbf{b} \quad (12)$$

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

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

This system shows that the semivariance γ must be known for the prediction point and all observation points (this is the $\mathbf{b}$ vector) and also between all pairs of known points (this is the $\mathbf{A}$ 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 $\mathbf{A}$ is positive definite; this is guaranteed by using authorized models. This has the solution (in matrix notation):

$$\lambda = \mathbf{A}^{-1}\mathbf{b} \quad (13)$$

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

$$\hat{z}(\mathbf{x}_0) = \sum_{i=1}^{N} \lambda_i z(\mathbf{x}_i) \quad (14)$$

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:

$$\hat{\sigma}^2(\mathbf{x}_0) = \mathbf{b}^T\lambda \quad (15)$$

! → 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.

9.2 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 54 : 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 55 : 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(logZn ~ 1, locations = meuse, newdata = meuse.grid,
      model = vmf)

[using ordinary kriging]
```

As with the empirical variogram plot (§8) and feature-space modelling (§7), 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`.

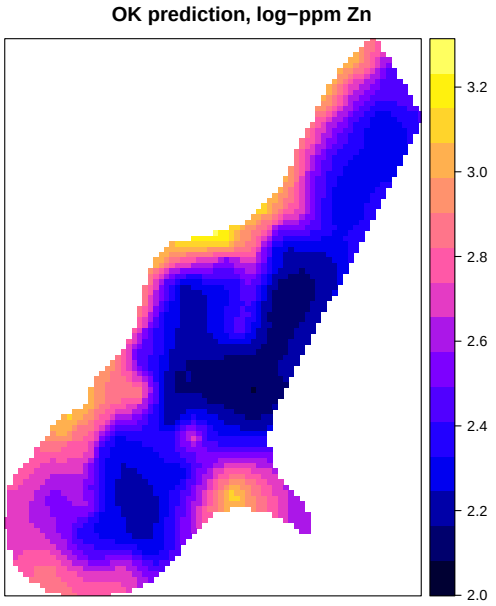
Task 56 : Display the structure of the kriging prediction object. •

```
> 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
  .. .. ..$ : chr [1:3103] "1" "2" "3" "4" ...
  .. .. ..$ : 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 slot
  .. .. ..@ 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 57 : Display the map of predicted values, usng a blue-pink-yellow colour ramp (smooth transition among colours showing different 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.

The `bpy.colors` "blue-pink-yellow colour scheme" argument to the `col.regions` function colour ramp is a function that produces a vector of colours. [17]

```
> head(bpy.colors(64))

[1] "#000033FF" "#000042FF" "#000050FF" "#00005FFF" "#00006DFF"
[6] "#00007CFF"

> bpy.colors(64)[32]

[1] "#C225DAFF"
```

Each colour is made of four numbers, representing Red, Green, Blue and alpha (transparency) on $[0 \ldots 255]$, represented as hexidecimal characters, so `FF` is $(16 \cdot 16) - 1 = 255$. So for example the 32^{nd} colour, which is used for the midpoint of the scale, is `C225DAFF`; its red component is `C2`, which is $(12 \cdot 16) + 2 = 194$, i.e., $\approx 76\%$ saturated (255) Red.

Q54 : *Describe the kriged map with respect to its (1) smoothness, (2) values at observed data points.* *Jump to A54* •

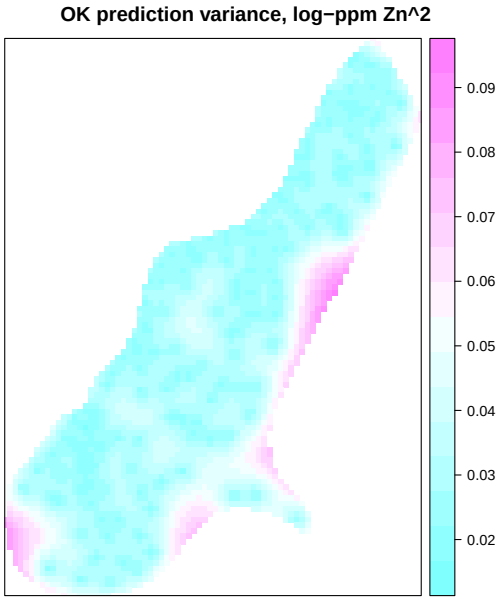
Task 58 : Display the map of kriging prediction variances. •

To emphasize that these are variances and not predictions, we specify the `cm.colors` "cyan-to-magenta colour scheme" colour ramp.

```
> print(spplot(k40, "var1.var",
                col.regions=cm.colors(64),
```

[17] This colour ramp also displays well if printed in gray scale.

```
        asp=1,
        main="OK prediction variance, log-ppm Zn^2"))
```



Q55 : *Describe the variances map. Where is the prediction variance lowest? Does this depend on the data value?* *Jump to A55* •

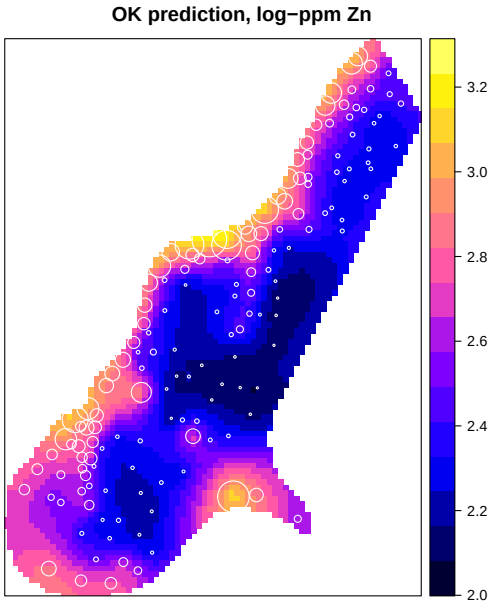
It may be helpful to visualize the predictions and their variances in relation to the observations. To do this, we first specify a list of extra items to add to `spplot` as so-called "panel functions", in this case just one: `sp.points`, and then add that to the `spplot` function call with the `sp.layout` argument.

> **Note:** Trellis graphics such as `spplot`, as implemented by the `lattice` package, have a complicated syntax which takes some time to learn; the thing to remember here is that we are adding a plot of the points on top of the original plot of the predictions or their variances, shown as a grid.

For the kriging prediction, we show the post-plot: value proportional to circle size[18].

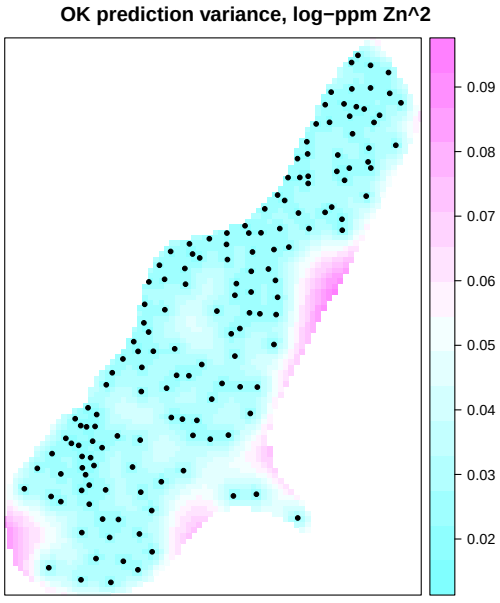
```
> pts.s <- list("sp.points", meuse, col="white",
                pch=1, cex=4*meuse$zinc/max(meuse$zinc))
> print(spplot(k40, "var1.pred", asp=1, col.regions=bpy.colors(64),
               main="OK prediction, log-ppm Zn",
               sp.layout = list(pts.s)
               ))
```

[18] To emphasize the differences, we use the original Zn values, not the transformed ones, to size the circles.



For the kriging prediction variance, we show only the observation locations:

```
> pts.s <- list("sp.points", meuse, col="black", pch=20)
> print(spplot(k40, zcol="var1.var",
                col.regions=cm.colors(64), asp=1,
                main="OK prediction variance, log-ppm Zn^2",
                sp.layout = list(pts.s)
                ))
```



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[19], 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 59 : Convert the observations for Zn to an indicator, and add to the data frame; summarize them. •

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:

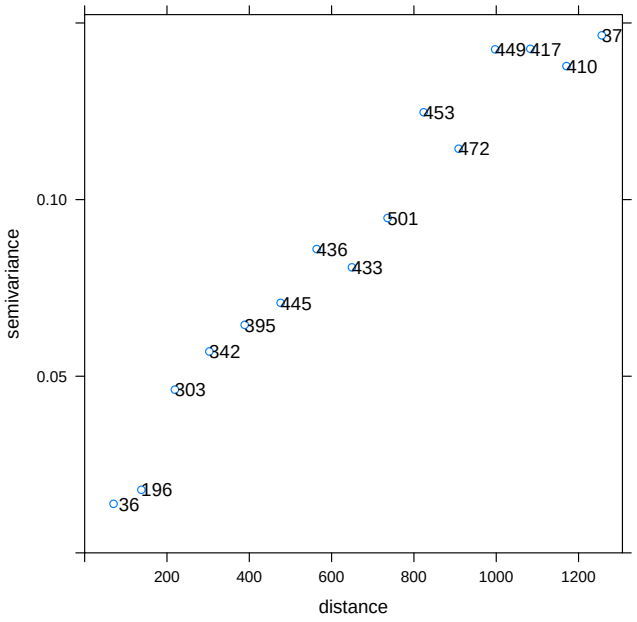
```
> meuse$zn.i <- (meuse$zinc < 150)
> summary(meuse$zn.i)

   Mode   FALSE    TRUE
logical     140      15
```

Q56 : *How many observations are above the threshold?* *Jump to A56* •

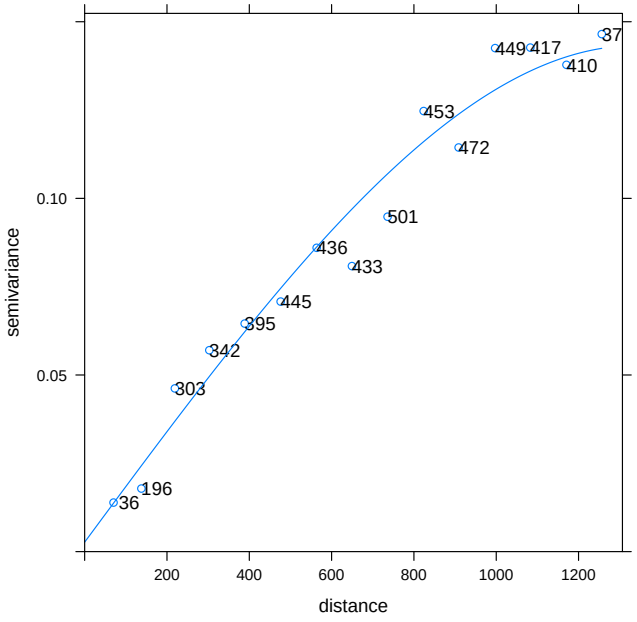
Task 60 : Make an empirical **indicator variogram** and model it. •

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

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




```
> (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))
```



Q57 : *What is the range of the indicator? Does this match that for the*

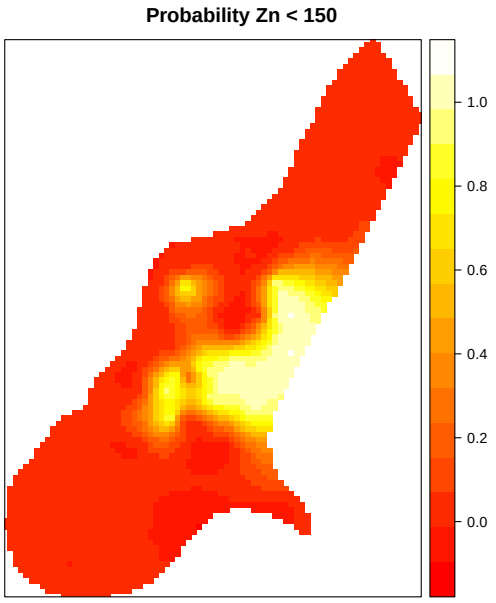
original variable? *Jump to A57* •

Q58 : *What is the sill? What are its units?* *Jump to A58* •

Task 61 : 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`.

Q59 : *What parts of the study area are safe for agricultural land use?* *Jump to A59* •

11 Mixed predictors

In §7 we saw that the feature-space attribute "flooding frequency" explained about 25% of the variation in $\log_{10}$Zn concentration. Yet, we ignored this

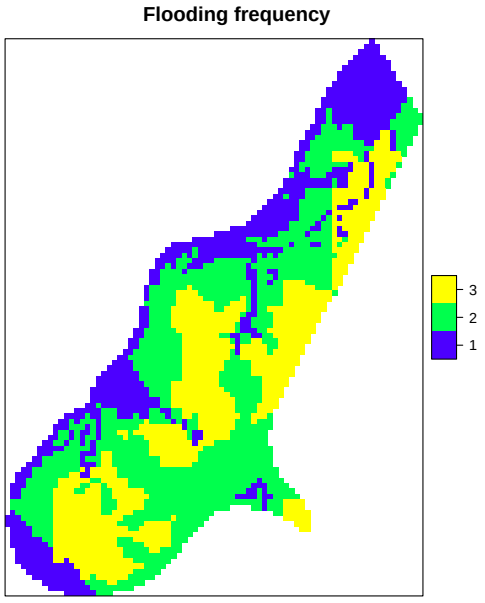
information when predicting by Ordinary Kriging (§9.2). In this section we examine a method to combine the two.

11.1 Feature-space prediction

In §7 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"))
```



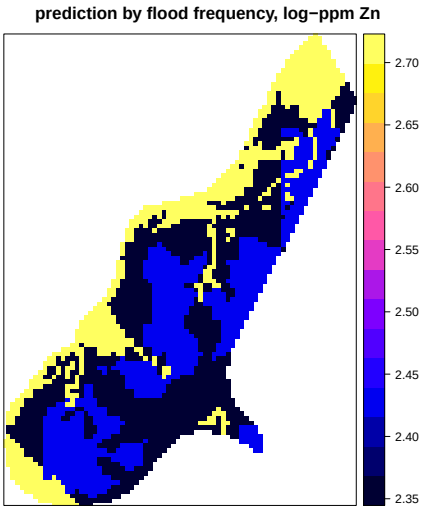
Here we use yet another colour ramp, using three colours from the `topo.colors` function.

Task 62 : 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

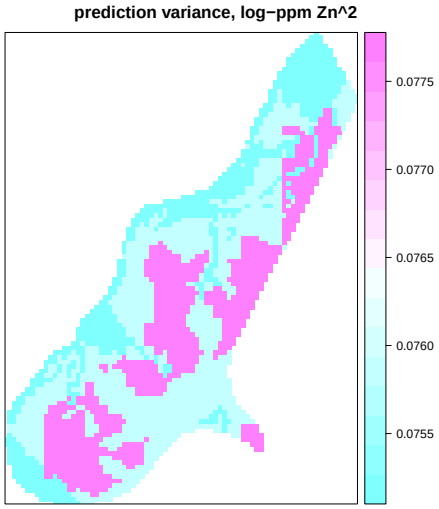
OLS regression exactly as does the `lm` function, and then uses that regression to fill the interpolation grid.

```
> k.ffreq <- krige(logZn ~ 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. As in §9.2 for Ordinary Kriging, we emphasize that this is not the prediction, by displaying the map of the prediction variance with a different colour ramp, here `cm.colors`:

```
> print(spplot(k.ffreq, zcol="var1.var",
               col.regions=cm.colors(64),
               main="prediction variance, log-ppm Zn^2"))
```



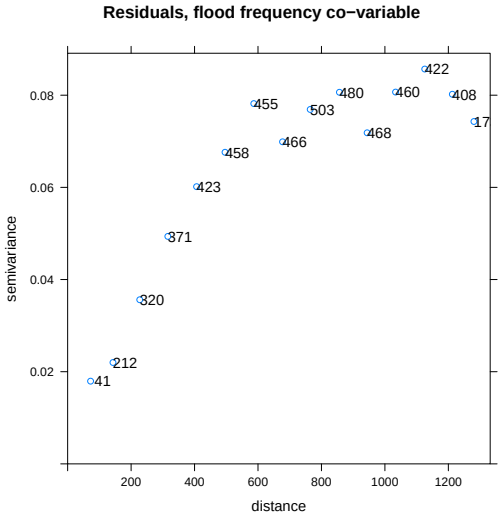
Q60 : *Explain the spatial pattern of this prediction and its variance.* *Jump to A60* •

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 63 : Compute and display the empirical residual variogram of $\log_{10}$Zn, after accounting for flooding frequency. •

```
> (vr <- variogram(logZn ~ 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"))
```



Q61 : *How does this empirical variogram compare to the original (non-residual) empirical variogram?* *Jump to A61*

•

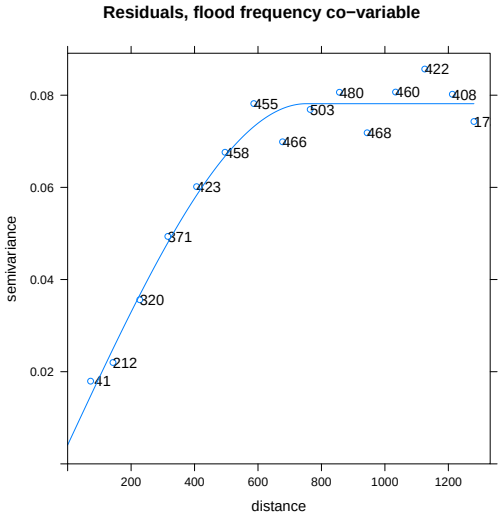
Clearly, accounting for the flood frequency has lowered the total variability (as expected from the results of the linear modelling), but it has also reduced the range of spatial dependence. Some of the apparent range in the original variogram was due to the spatial extent of the flooding classes; this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

Task 64 : 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, model=vrmf,
              main="Residuals, flood frequency co-variable"))
```



Q62 : *How does this variogram model compare to the original (non-residual) variogram model?* *Jump to A62* •

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, so the first histogram bin has a lower value, which pulls down the fit – although in theory the nugget should not change.

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

Now the prediction. We use the same `feature-space dependence` formula `logZn ~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.

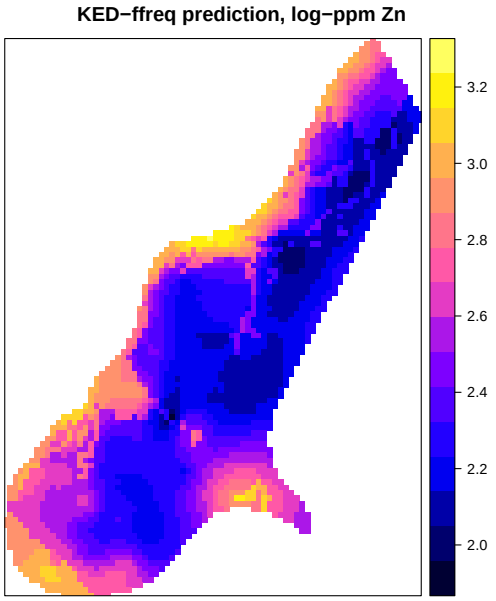
! → In KED, the formula for kriging *must* match that for the residual variogram.

```
> kr40 <- krige(logZn ~ 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 66 : 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"))
```



Q63 : *How does this KED map compare to the OK map? Where is the effect of flood frequency class reflected in the prediction?* *Jump to A63* •

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.

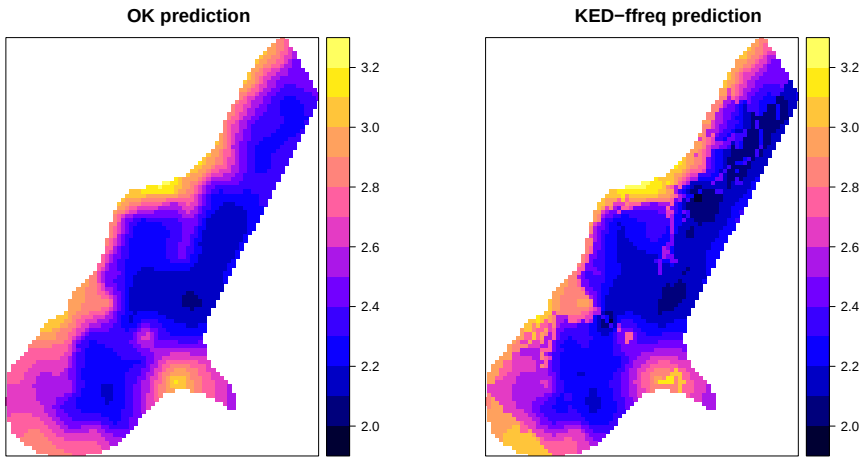
However, there is one important detail before we can do this – the scales of the two plots must be the same, for correct visual comparison. So, we detemine the overall maximum range and then use this for both the plots. The `max` and `min` functions find the extremes; we round these up and down to the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

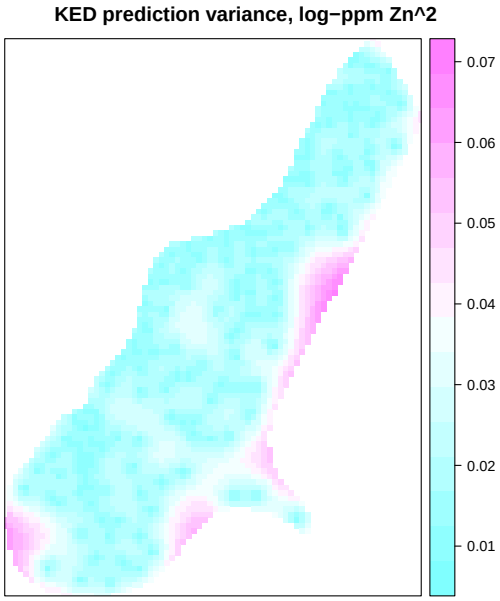
```
> plot(p1, split = c(1, 1, 2, 1), more = T)
> plot(p2, split = c(2, 1, 2, 1), more = F)
```



11.3.2 KED Prediction variances

Task 67 : Display the map of the KED prediction variances. •

```
> print(spplot(kr40, "var1.var", asp=1,
                col.regions=cm.colors(64),
                main="KED prediction variance, log-ppm Zn^2"))
```

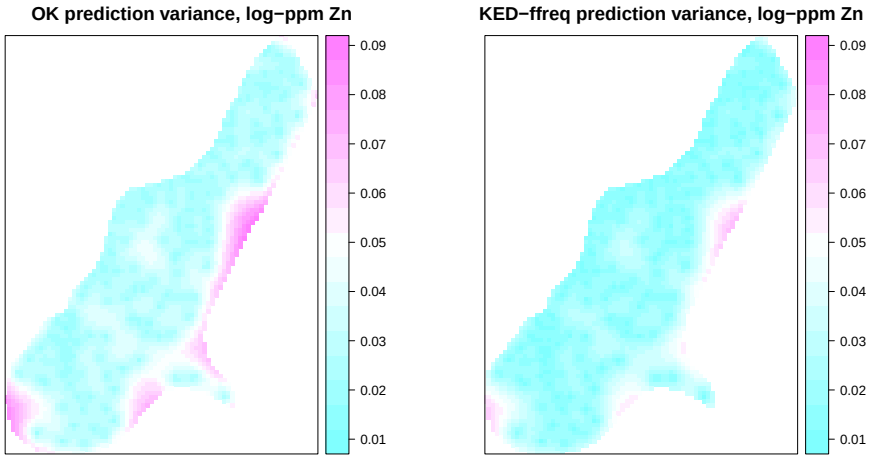


Task 68 : Compare these prediction variances to those for OK, both numerically and graphically. •

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

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00808 0.01628 0.02045 0.02364 0.02807 0.06861

> 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",
               col.regions=cm.colors(64),
               asp=1, at=ramp,
               main="OK prediction variance, log-ppm Zn")
> p2 <- spplot(kr40, "var1.var",
               col.regions=cm.colors(64),
               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 of the linear model part of KED is the same everywhere[20]. 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 69 : 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"
```

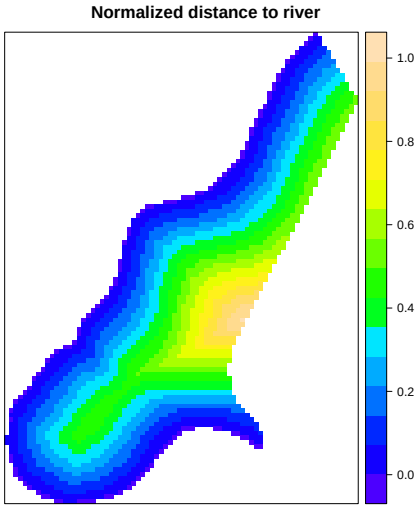
Q64 : *What covariables are available?* *Jump to A64* •

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.

[20] Of course, the prediction variance of the residual part, i.e., not accounted for by the linear model, does vary according to the observation points' relative locations.

Task 70 : Display the normalized distance to river. •

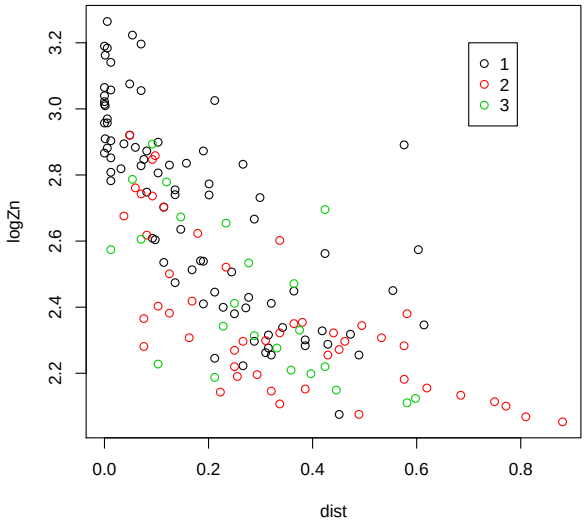
```
> print(spplot(meuse.grid, zcol="dist",
                main="Normalized distance to river",
                col.regions=topo.colors(64)))
```



In §7 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 71 : Display a feature-space scatterplot of metal concentration vs. distance from river. •

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



Q65 : *Does there appear to be a linear relation between distance and metal concentration? How strong does it appear to be?* *Jump to A65* •

Task 72 : 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(logZn ~ dist, data = meuse)
> summary(m.lzn.dist)

Call:
lm(formula = logZn ~ 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
```

Q66 : *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 A66* •

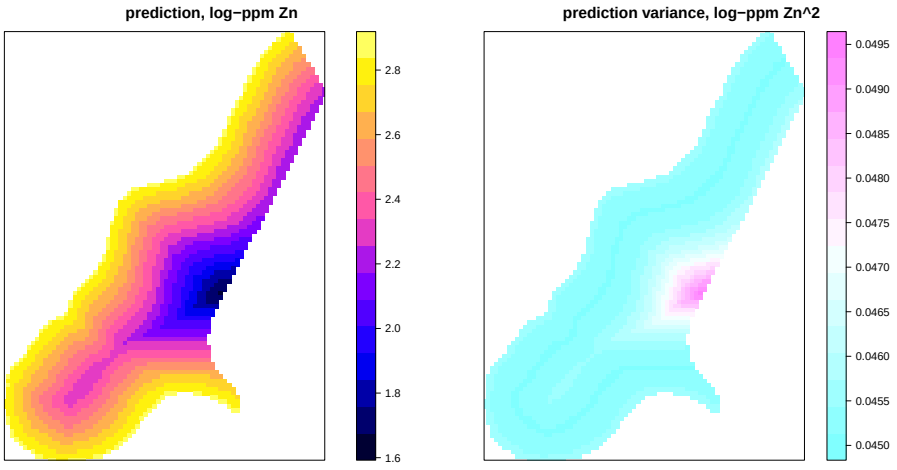
Task 73 : Predict the metal concentration over the study area, from the distance to river. •

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(logZn ~ 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",
               col.regions=cm.colors(64),
               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)
```



Q67 : *Explain the spatial pattern of this prediction and its variance.* *Jump to A67* •

Task 74 : 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 §7; that model should still be in the workspace.

```
> m.lzn.ff.dist <- lm(logZn ~ ffreq + dist, data=meuse)
> m.lzn.ff.dist.i <- lm(logZn ~ 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: logZn ~ ffreq * dist
Model 2: logZn ~ ffreq + dist
Model 3: logZn ~ dist
Model 4: logZn ~ 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.

Q68 : *Do the two-predictor models give significantly lower residual sum-of-squares?* *Jump to A68*

•

Q69 : *Does the interaction model give significantly lower residual sum-of-squares than the additive model?* *Jump to A69*

•

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 75 : 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
```

Q70 : *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 A70* •

Task 76 : Display the model summary for the best model. •

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

Call:
lm(formula = logZn ~ 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
```

Q71 : *How much of the variability in metal concentration is explained by this model?* *Jump to A71* •

11.4.1 Linear model diagnostics

Recall that a linear model assumes the form:

$$z_i = \beta_0 + \sum_{j=1}^{k} \beta_i x_{ij} + \varepsilon_i \tag{16}$$

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.

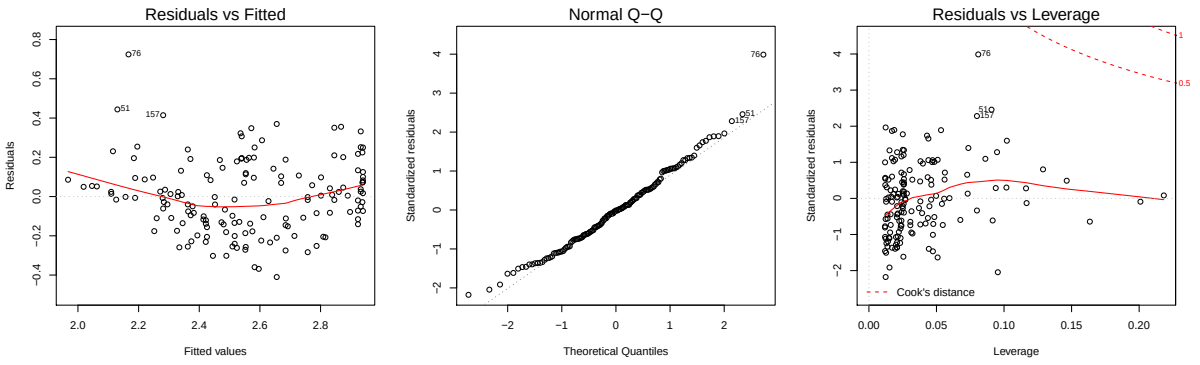
As explained in §7.2.2 linear model must satisfy several assumptions [8, 5], among which are:

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.

Task 77 : Display the model diagnostics for the interaction model. •

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



Q72 : *Looking at the "Residuals vs. fitted values" plot, 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 A72* •

Q73 : *Looking at the "Normal Q-Q" plot, do the residuals appear to be normally distributed?* *Jump to A73* •

Q74 : *Looking at the "Residuals vs. leverage" plot, do the high-leverage residuals have a high Cook's distance (a measure of how much the observation influences the model)?* *Jump to A74*
•

There are three poorly-modelled points, labelled 51, 157, and especially 76, 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 numbers shown are the observation names, given by the `row.names` function, they are *not* necessarily the matrix row numbers of the observations in the data frame, i.e., the indices that are used to access a given row using the `[]` selection operator. In this dataset some field observations were excluded, so rows were omitted, thus there are gaps in the row name sequence, but of course not in the row numering for a matix..

To find the matrix indices we use the `which` function with a logical condition that is `TRUE` for this given row names.

```
> (ix <- which(row.names(meuse@data) == "76"))

[1] 69

> meuse@data[ix, ]

   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 logZn  logCu  zn.i
76      Bw    750 2.891 1.7404 FALSE
```

So, in the case of matrix row `69` the data frame row name is 76.

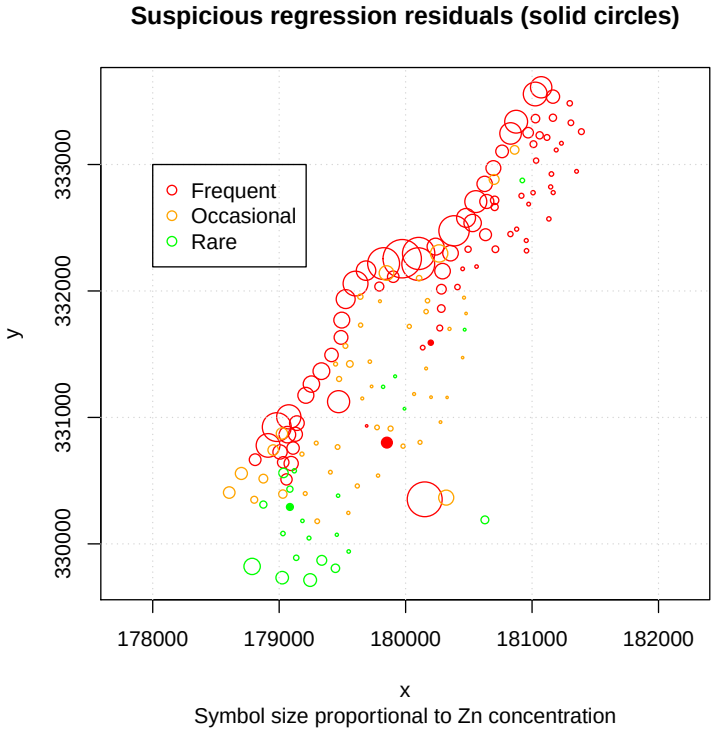
Task 78 : Plot the suspicious regression residuals in geographic space. •

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 <- which(row.names(meuse@data) %in% c("76",
      "51", "157")))

[1]  50  69 152
```

```
> bad.row <- (row(meuse@data)[, 1] %in% bad.pt)
> which(bad.row)

[1]  50  69 152

> 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, 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)
```

11.4.2 Spatial structure of the the residuals

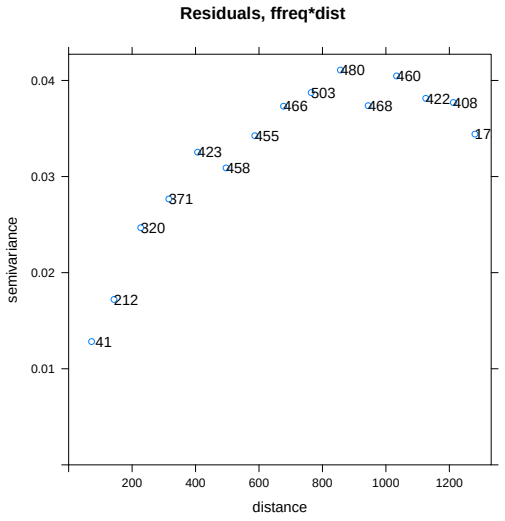
Now that we've identified a good model (substantially better than the single-predictor model with just flooding frequency), we continue with the local structure of the residuals.

Task 79 : 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(logZn ~ ffreq * dist, location = meuse,
      cutoff = 1300, width = 90))
```

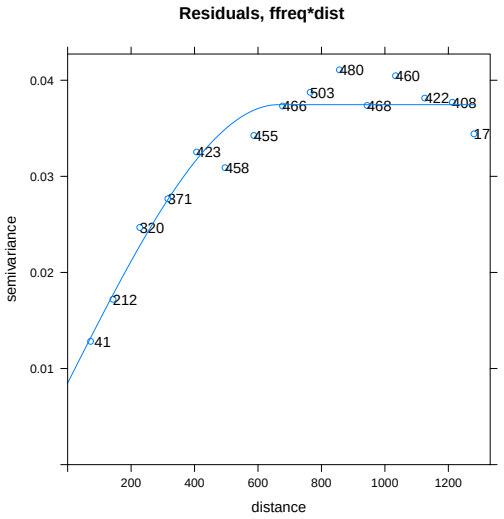
```
    np     dist    gamma dir.hor dir.ver   id
1   41   72.248 0.012830       0       0 var1
2  212  142.880 0.017211       0       0 var1
3  320  227.322 0.024683       0       0 var1
4  371  315.855 0.027669       0       0 var1
5  423  406.448 0.032533       0       0 var1
6  458  496.094 0.030899       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"))
```



Q75 : *What happens to the values of the ranges and partial (structural) sills as the model includes more predictors?* *Jump to A75* •

11.4.3 KED prediction

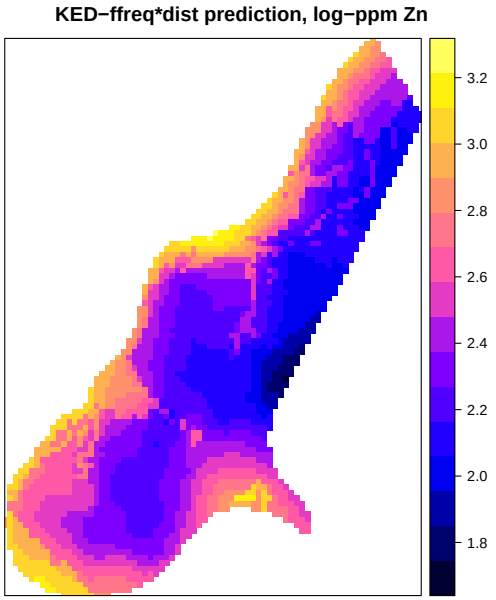
Task 80 : Predict by KED using the best feature-space model as covariables. •

```
> kr240 <- krige(logZn ~ 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 81 : 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"))
```



Q76 : *How does this KED map compare to the OK map, and the single-predictor (flood frequency) KED map?*

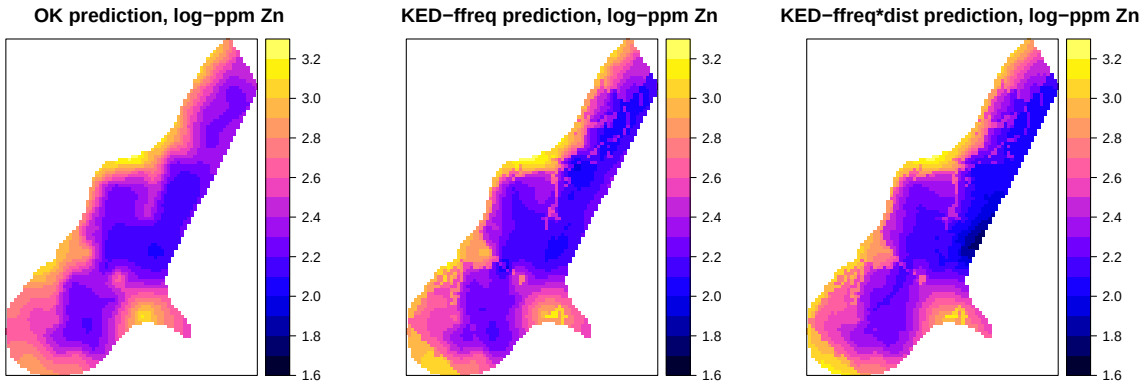
Where is the effect of flood frequency class and distance to river reflected in the prediction? *Jump to A76* •

Task 82 : 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)
```



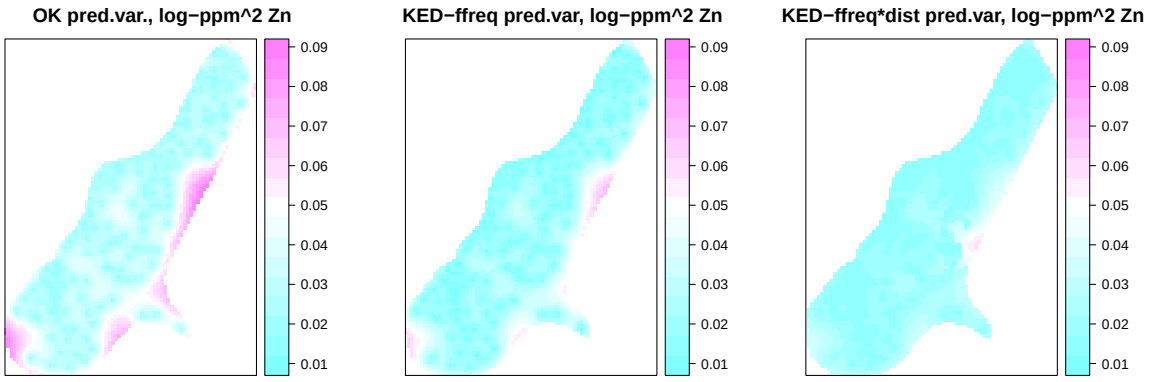
11.4.4 KED prediction variances

Task 83 : 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.
0.00808 0.01628 0.02045 0.02364 0.02807 0.06861

> 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",
               col.regions=cm.colors(64),
               asp=1, at=ramp,
```

```
                main="OK pred.var., log-ppm^2 Zn")
> p2 <- spplot(kr40, "var1.var",
                col.regions=cm.colors(64),
                asp=1, at=ramp,
                main="KED-ffreq pred.var, log-ppm^2 Zn")
> p3 <- spplot(kr240, "var1.var",
                col.regions=cm.colors(64),
                asp=1, at=ramp,
                main="KED-ffreq*dist pred.var, log-ppm^2 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)
```



Q77 : *What is the effect of adding the distance to river as a predictor on the spatial pattern of the prediction variances?* *Jump to A77* •

12 Model evaluation

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**.[21] One aspect of model evaluation is to assess its predictive power: how well is it expected to perform when predicting at unmeasured points *in the target population*? We have made these predictions over the Meuse grid; how accurate (close to the true value) and precise (uncertain) are they?

One measure is the **kriging prediction variance** at each prediction point and their summary statistics; see §9.2 (OK) and §11.3.2 (KED). This is *internal* to the kriging procedure, and depends on the correctness of the model of spatial dependence, i.e., the variogram model.

A model-free approach to assessing predictive power is comparing the predictions at points that were *not* used to **build** the model, e.g., to select the

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

variogram model form and fit it, and *not* used as data points to make the **predictions**, e.g., by kriging. Thus many studies have both:

1. a **calibration** data set, used to build the model and predict;
2. a so-called **validation** (evaluation, independent) set, where the model predicts, without using their known values.

Then the model predictions $\hat{y}_i$ are compared with the actual values y_i, and summarized by descriptive and inferred population statistics. Brus *et al.* [4] give an extensive discussion of sampling for map validation, and point out that the resulting validation statistics are only correct for the entire target population if the validation points are a **probability** (random) sample. In our study we have an obvious non-probability sample, which is no problem for a *model-based* mapping approach (e.g., kriging), where the randomness comes from the model, not the data.

12.1 Independent validation set

In some studies there is one sampling campaign, and then the whole dataset is randomly *split* into calibration and validation sets. But we used all the observations to build our model and predict; we do not have an independent validation set. If we split it, re-fit the model with only the calibration set, and predict at the validation points (left out of the calibration set), we would have some measure of predictive accuracy but (1) probably a poorly-fit model, because of the small number of points overall; (2) not a good measure of the population predictive accuracy.

So we turn to another "external" (more or less) approach, **cross-validation**.

12.2 Cross-validation

One of the characteristics 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:** Strictly speaking, we should re-fit the variogram model without each point in turn; in practice this makes almost no difference to fitted model, because only a very small proportion of the point-pairs would not be used to

fit the variogram model. So we use the single fitted variogram model from all the points for the entire cross-validation.

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.

Task 84 : Perform LOOCV for the OK and KED predictions. •

The `krige.cv` method performs this.

```
> kcv.ok <- krige.cv(logZn ~ 1, locations = meuse,
      model = vmf)
> kcv.rk <- krige.cv(logZn ~ ffreq, locations = meuse,
      model = vrmf)
> kcv.rk2 <- krige.cv(logZn ~ 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 85 : 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 86 : 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.
-0.42847 -0.09643 -0.00452 -0.00015  0.08528  0.64139

> summary(kcv.rk$residual)

    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.56124 -0.08345 -0.01365  0.00094  0.08987  0.47683

> summary(kcv.rk2$residual)

    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.56126 -0.08464  0.00142  0.00141  0.07841  0.59062
```

Q78 : *Is any prediction **biased**? (Compare the mean to zero). Which has the narrower overall and inter-quartile range?* *Jump to A78* •

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
```

Q79 : *Which prediction is, on average, more precise?* *Jump to A79* •

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.

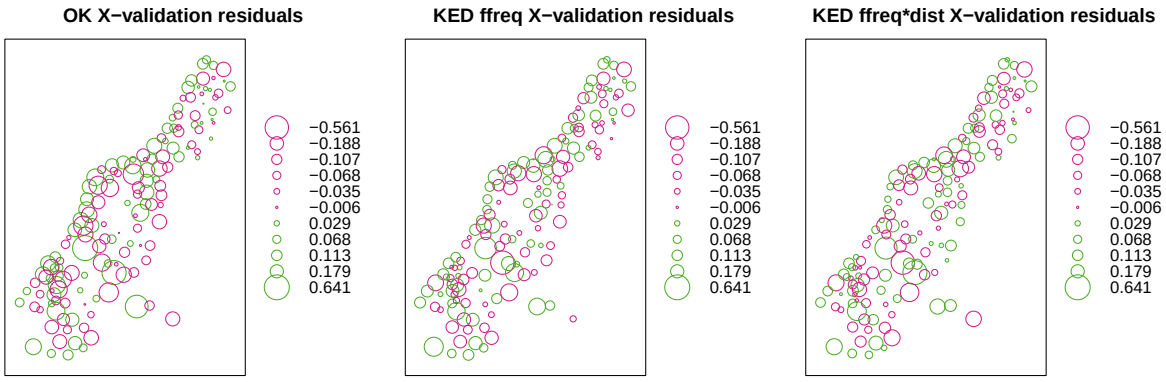
Task 87 : Display bubble plots of the OK and KED (flood frequency and distance interaction) cross-validations. •

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 residuals",
               key.entries=ramp, pch=1)
> p3 <- bubble(kcv.rk2, zcol="residual",
               main="KED ffreq*dist X-validation residuals",
               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)
```



Q80 : *Is there any pattern to the positive and negative residuals? Is there a difference between the OK and KED patterns?* *Jump to A80* •

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 * Generalized Additive Models

Generalized Additive Models (GAM) are similar to multiple linear regression, except that each term in the linear sum of predictors need not be the predictor variable itself, but can be an empirical smooth function of it. So instead of the linear model of k predictors:

$$y = \beta_0 + \sum_k \beta_k x_k \tag{17}$$

we allow *functions* f_k of these:

$$y = \beta_0 + \sum_k f_k(x_k) \tag{18}$$

The advantage is that non-linear relations in nature can be fit; the disadvantage is that there is no single equation to describe the relation, it is just an empirical fit.

> **Note:** Further, the GAM should never be extrapolated (there is no data to support it), whereas a polynomial can, with caution, be extrapolated, on the theory that the data used to fit the model extends outside the range. This is of course very dangerous for higher-order polynomials, which are a main competitor to GAM.

Hastie *et al.* [11, §9.1] give a thorough explanation of GAM; a simplified explanation of the same material is given in James *et al.* [15, §7.7]. In a geostatistical setting, we can choose the coördinates as the predictors (as in a trend surface) but fit these with smooth functions, rather than polynomials. We can also fit any other predictor this way.

To illustrate this with the Meuse dataset, we'll fit a model of Zn concentration in the soil based on two predictors: distance to river and elevation. However, we do not assume linear, linearizable or higher-order polynomial relations with either of these; rather we assume they vary smoothly but not according to any single equation.

Q81 : *What theory of the origin of the Zn is this model testing?* *Jump to A81* •

Notice that we are not using metric coördinates in space (here, the Dutch grid system), rather, we are using the distance from river as a "coördinate" to express the spatial relation. In the `meuse` dataset distance is available as distance in meters or as a normalized distance; we choose the latter because

it is available in the interpolation grid, see below. Since there is no statistical model, there is no issue with spatial dependence of residuals.

GAM can be fitted in R with the `mgcv` "Mixed GAM Computation Vehicle" package.

Task 88 : Load the `mgcv` package into the workspace. •

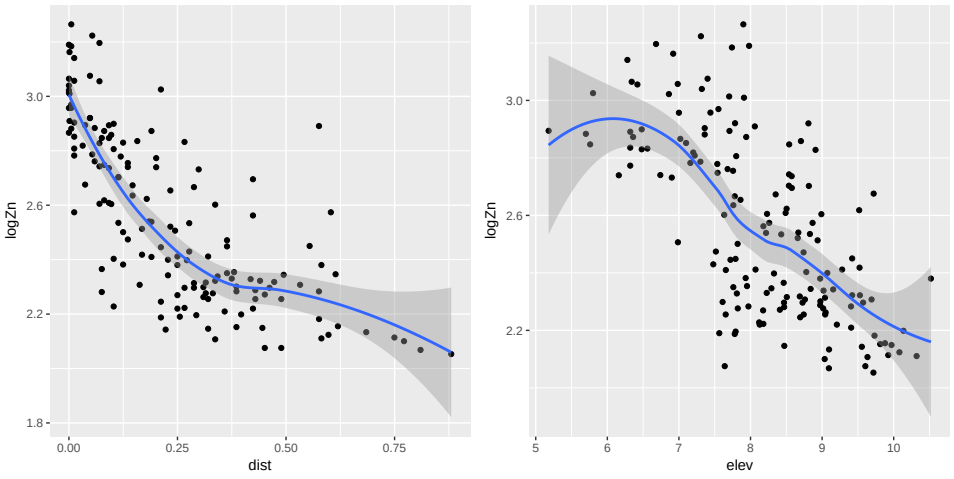
```
> library(mgcv)
```

Task 89 : Display a scatterplot of the two predictors against the $\log_{10}$Zn, with an empirical smoother provided by the `ggplot2` graphics package. •

The `qplot` "quick plot" function is the `ggplot2` equivalent of base graphics flexible `plot` function.

> **Note:** The `gridExtra` package provides a `grid.arrange` function to arrange saved `ggplot2` or `lattice` graphics objects on a page.

```
> library(ggplot2)
> p1 <- qplot(x = dist, y = logZn, data = meuse@data,
      geom = c("point", "smooth"))
> p2 <- qplot(x = elev, y = logZn, data = meuse@data,
      geom = c("point", "smooth"))
> require(gridExtra)
> grid.arrange(p1, p2, ncol = 2)
```



Q82 : *Do these relations look linear? Do they look as if they could be well-fit with some transformation such as inverse, quadratic, logarithmic?* *Jump to A82* •

Task 90 : Build two GAM, one with each of these two predictors (so, not

yet additive), using the default smoothing function. Examine the model summaries. •

The empirical smooth function is specificed with the `s` "smooth" function provided by the `mgcv` package. The degree of smoothness is estimated as part of fitting, by using regression splines penalized for the number of knots (i.e., effective degrees of freedom). The best choice is selected by cross-validation.

```
> library(mgcv)
> m.g.dist <- gam(logZn ~ s(dist), data = meuse)
> summary(m.g.dist)

Family: gaussian
Link function: identity

Formula:
logZn ~ s(dist)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.5562     0.0148     173   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
         edf Ref.df    F p-value
s(dist) 3.56   4.42 65.9  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.655   Deviance explained = 66.3%
GCV = 0.034927  Scale est. = 0.033899  n = 155

> summary(residuals(m.g.dist))

    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.49586 -0.11387 -0.00769  0.00000  0.09085  0.61384

> m.g.elev <- gam(logZn ~ s(elev), data = meuse)
> summary(m.g.elev)

Family: gaussian
Link function: identity

Formula:
logZn ~ s(elev)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.5562     0.0187     137   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
        edf Ref.df  F p-value
```

```
s(elev) 3.8   4.76 27  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.451   Deviance explained = 46.5%
GCV = 0.055675  Scale est. = 0.053951  n = 155

> summary(residuals(m.g.elev))

    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.5911 -0.1498 -0.0194  0.0000  0.1375  0.6682
```

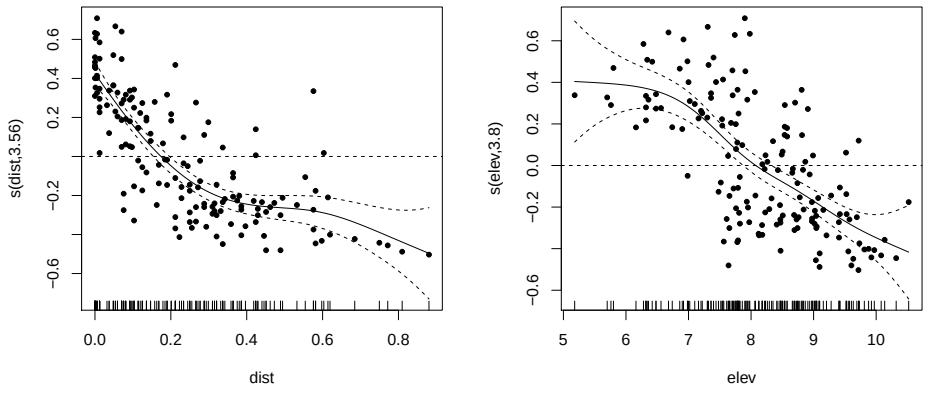
The model summary gives the adjusted R^2, i.e., proportion of variance explained, and a closely-related statistic, the deviance explained.

Q83 : *Do both predictors provide useful information about the metal concentration? Which of the two predictors is better?* *Jump to A83*
•

Task 91 : Repeat the scatterplots, but showing the fitted functions. •

The `plot.gam` function plots the smoother, relative to the mean of the response variable:

```
> par(mfrow = c(1, 2))
> plot.gam(m.g.dist, residuals = T, pch = 20)
> abline(h = 0, lty = 2)
> plot.gam(m.g.elev, residuals = T, pch = 20)
> abline(h = 0, lty = 2)
> par(mfrow = c(1, 1))
```



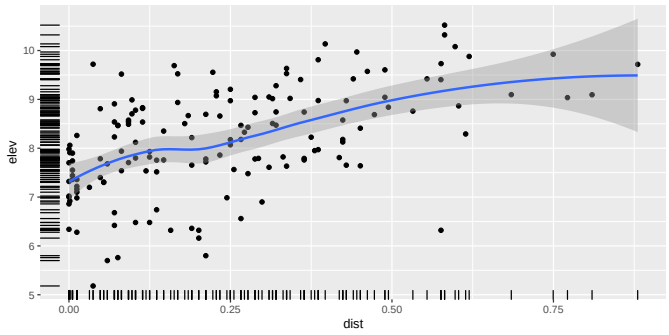
Q84 : *Describe the fitted relations. Do these fit the hypothesis of the origin of the metals?* *Jump to A84* •

Challenge: Build linear models, maybe with some transformation of the predictors, and compare their success to the single-predictor GAM. Examine the regression residuals to see if a linear model is justified.

We suspect thay there may be an **interaction** between the predictors: higher areas tend to be further from the river, although there are some high points near the river (on the dikes), so that the two predictors are not simply additive in their effects.

Task 92 : Display a scatterplot of elevation against distance from the river. •

```
> qplot(x = dist, y = elev, data = meuse@data, geom = c("point",
      "smooth", "rug"))
> cor(meuse$elev, meuse$dist, method = "pearson")

[1] 0.5301

> cor(meuse$elev, meuse$dist, method = "spearman")

[1] 0.54637
```



Q85 : *How are elevation and distance related? Is it a strong relation?* *Jump to A85* •

Task 93 : Build an additive GAM with the two predictors distance and elevation. Also build an additive GAM with the two predictors and an interaction term. Compare their model summaries and residuals. •

The additive model just uses the `+`, as in a linear model. To specify the interaction, we don't use `*`, which is for a linear interaction. Instead the `ti` "tensor product interaction between smoothers" function is used; this is the multi-dimension extension of the one-dimensional `s` "smoother" function.

```
> m.g.dist.elev <- gam(logZn ~ s(dist) + s(elev), data = meuse)
> m.g.dist.elev.i <- gam(logZn ~ s(dist) + s(elev) +
      ti(dist, elev), data = meuse)
> summary(m.g.dist.elev)
```

```
Family: gaussian
Link function: identity

Formula:
logZn ~ s(dist) + s(elev)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.5562     0.0121     212   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
         edf Ref.df    F p-value
s(dist) 3.92   4.84 41.7 < 2e-16 ***
s(elev) 6.14   7.31 10.7 2.2e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =   0.77   Deviance explained = 78.5%
GCV = 0.024327  Scale est. = 0.022592  n = 155

> summary(m.g.dist.elev.i)

Family: gaussian
Link function: identity

Formula:
logZn ~ s(dist) + s(elev) + ti(dist, elev)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    2.556      0.016     160   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
               edf Ref.df     F p-value
s(dist)       3.77   4.68 36.87 < 2e-16 ***
s(elev)       4.30   5.47 15.12 1.1e-12 ***
ti(dist,elev) 4.72   5.20  3.12   0.011 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.785   Deviance explained = 80.3%
GCV = 0.023195  Scale est. = 0.021133  n = 155

> summary(residuals(m.g.dist.elev))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.4160 -0.0857 -0.0156  0.0000  0.0912  0.3752

> summary(residuals(m.g.dist.elev.i))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.4152 -0.0762 -0.0142  0.0000  0.0854  0.3494
```

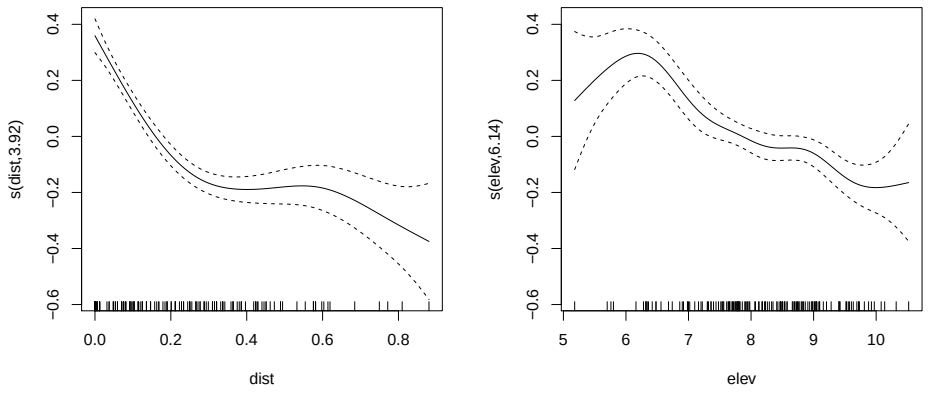
Q86 : *Are these models better than either of the two single-predictor models? Is the interaction model better than the additive model? (Hint: look at the approximate significance of the interaction term.)* *Jump to A86* •

Task 94 : Plot the fitted smooth functions. •

The `pages` optional argument to the `plot.gam` function specifies the number of pages over which to spread the output; to see all graphs at once we specify a single page.

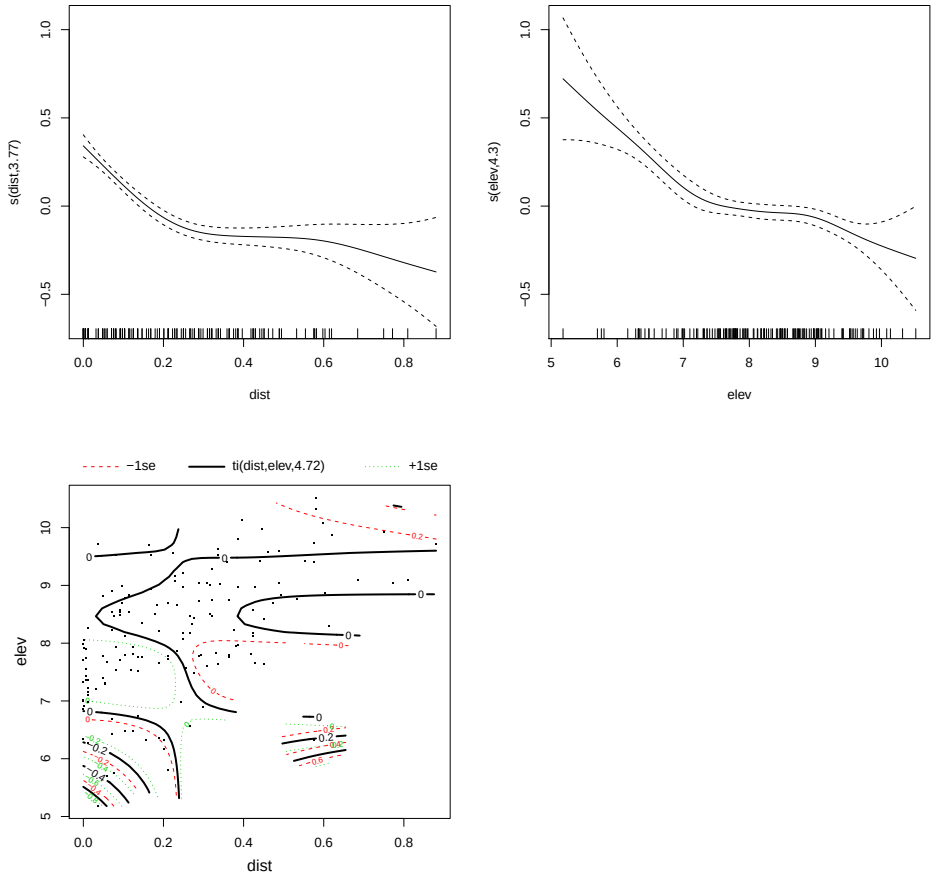
First the additive model:

```
> plot.gam(m.g.dist.elev, pages = 1)
```



Then the interaction model:

```
> plot.gam(m.g.dist.elev.i, pages = 1)
```

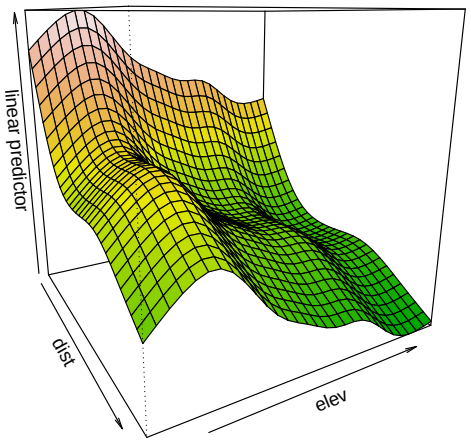


Q87 : *Compare the additive model to the two single-factor models. How do the smooth fits differ? Where does the interaction tensor most modify the additive relation?* *Jump to A87* •

Task 95 : Display a 3D plot of the response surface to distance and elevation. •

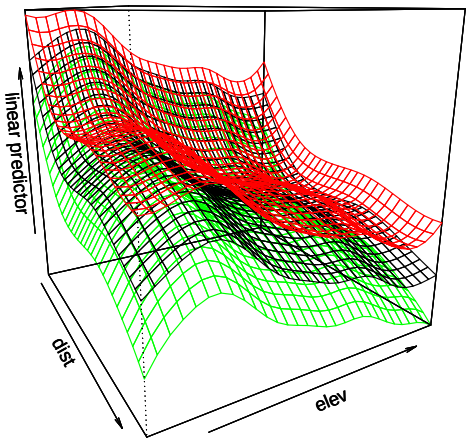
A very nice plot to show this is `vis.gam`.

```
> vis.gam(m.g.dist.elev, theta=+60,
          plot.type="persp", color="terrain")
```



The standard errors of prediction can also be shown, along with the prediction; the `se` argument specifies how many standard errors away from the prediction to display.

```
> vis.gam(m.g.dist.elev, theta=+60, color="terrain", se=1.96)
```



red/green are +/– 1.96 s.e.

Of course once we have a model we can use it for prediction, i.e., to show the predicted metal content everywhere in the study area. Unfortunately the prediction grid `meuse.grid` does not include the elevation predictor.

Although this has been included in a different version of the Meuse dataset[22] it's not in the dataset included in the `sp` package. So, we have to create it by interpolation from the known points.

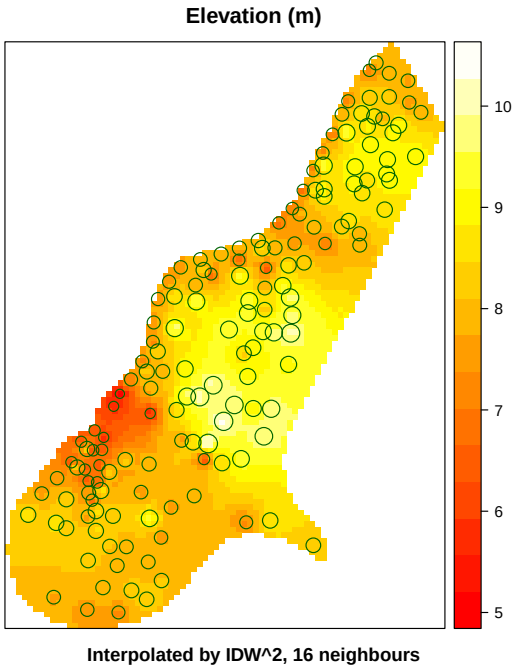
We just need to "fill in the holes" between the observed points.

Task 96 : Predict the elevation over the prediction grid by inverse distance squared interpolation. •

The `idw` function is a special case of the `krige` function. Instead of a variogram model it need the inverse distance power to be specified by the `idp` argument. We also specify the maximum number of neighbours to consider, with the `nmax` argument; in practice this doesn't make much difference since far-away points would receive much lower weights.

```
> tmp <- idw(elev ~ 1, locations=meuse,
             nmax=16, idp=2, newdata=meuse.grid)

[inverse distance weighted interpolation]

> meuse.grid$elev <- tmp$var1.pred
> pts.s <- list("sp.points", meuse, col="darkgreen",
                pch=1, cex=2*meuse$elev/max(meuse$elev))
> print(spplot(meuse.grid, zcol="elev",
               col.regions=heat.colors(64),
               main="Elevation (m)",
               sub="Interpolated by IDW^2, 16 neighbours",
               sp.layout = list(pts.s)))
> summary(meuse$elev)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   5.18    7.55    8.18    8.17    8.96   10.52

> summary(meuse.grid$elev)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   5.20    7.87    8.21    8.32    8.86   10.28
```

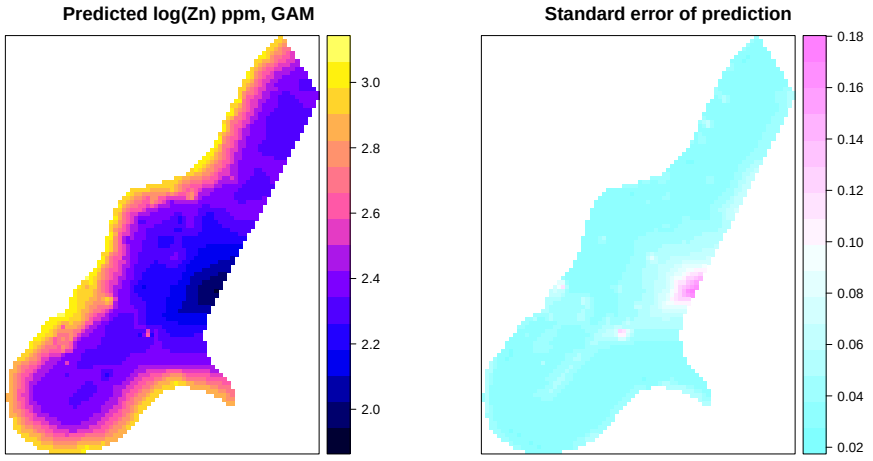
[22] http://spatial-analyst.net/book/meusegrids



Task 97 : Predict the $\log_{10}$Zn concentration over the grid using the best fitted GAM. •

The `predict.gam` function predicts using a fitted GAM.

```
> tmp <- predict.gam(object=m.g.dist.elev.i, newdata=meuse.grid, se.fit=TRUE)
> names(tmp)

[1] "fit"    "se.fit"

> meuse.grid$k.g.i <- tmp$fit
> meuse.grid$k.g.i.se <- tmp$se.fit
> p1 <- spplot(meuse.grid, zcol="k.g.i",
               col.regions=bpy.colors(64),
               main="Predicted log(Zn) ppm, GAM")
> p2 <- spplot(meuse.grid, zcol="k.g.i.se",
               col.regions=cm.colors(64),
               main="Standard error of prediction")
> grid.arrange(p1, p2, ncol=2)
```



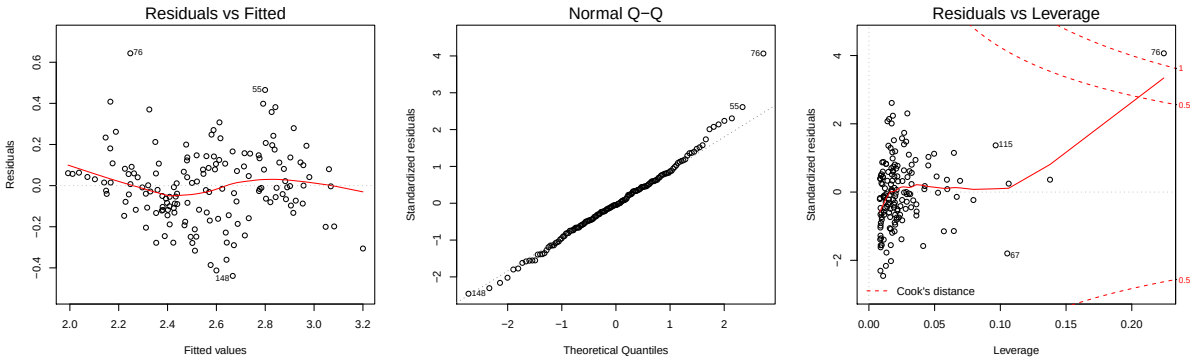
Q88 : *Comment on the suitability of this GAM.* *Jump to A88* •

We can compare this result with a linear model using the same formula.

Task 98 : Build a linear model with the same structure as the GAM, i.e., predictors elevation, distance, and their interaction. Summarize the model and plot its diagnostics. •

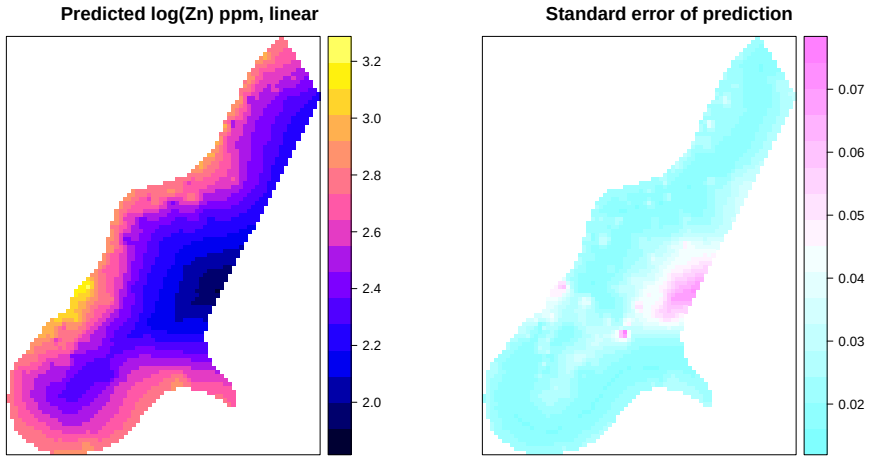
```
> summary(m.dist.elev.i <- lm(logZn ~ dist * elev,
      data = meuse))

Call:
lm(formula = logZn ~ dist * elev, data = meuse)

Residuals:
    Min      1Q  Median      3Q     Max
-0.4391 -0.1133 -0.0084  0.1060  0.6434

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   4.1390     0.1789   23.14  < 2e-16 ***
dist         -3.0256     0.6283   -4.82  3.5e-06 ***
elev         -0.1689     0.0227   -7.45  6.8e-12 ***
dist:elev     0.2523     0.0722    3.49  0.00062 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.18 on 151 degrees of freedom
Multiple R-squared:  0.678,        Adjusted R-squared:  0.671
F-statistic:  106 on 3 and 151 DF,  p-value: <2e-16
```

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



Task 99 : Plot the predictions and their standard errors. •

```
> tmp <- predict.lm(object=m.dist.elev.i,
                    newdata=meuse.grid, se.fit=TRUE)
> meuse.grid$k.i <- tmp$fit
> meuse.grid$k.i.se <- tmp$se.fit
> p1 <- spplot(meuse.grid, zcol="k.i",
               col.regions=bpy.colors(64),
               main="Predicted log(Zn) ppm, linear")
> p2 <- spplot(meuse.grid, zcol="k.i.se",
               col.regions=cm.colors(64),
               main="Standard error of prediction")
> plot(p1, split=c(1,1,2,1), more=T)
> plot(p2, split=c(2,1,2,1), more=F)
```

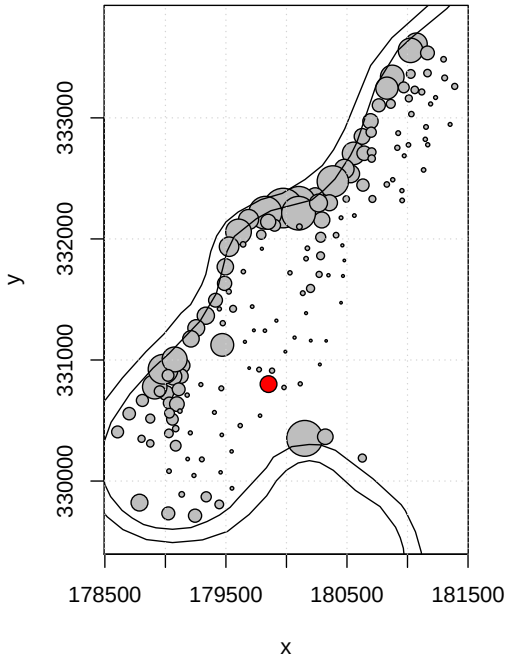


The residuals-vs.-fitted plot shows the difficulty that the linear model has with the lowest and highest values; there is one very poorly predicted point with high leverage.

Task 100 : Display the fitted and actual values for the high-leverage, high-residual point. •

The `row.names` function gives a vector of row names; these are displayed on the diagnostic plots. We find the row number for this and examine its values.

```
> ix <- which(row.names(meuse) == "76")
> meuse[ix, c("zinc", "elev", "dist")]

        coordinates zinc elev    dist
76 (179850, 330800)  778 6.32 0.57588

> log10(meuse@data[ix, "zinc"])

[1] 2.891

> fitted(m.dist.elev.i)[ix]

    76
2.2475

> fitted(m.g.dist.elev.i)[ix]

[1] 2.9206

> plot(coordinates(meuse), asp = 1, pch = 21, cex = 4 *
      meuse$zinc/max(meuse$zinc), bg = ifelse(row.names(meuse) ==
      "76", "red", "gray"))
> data(meuse.riv)
> lines(meuse.riv)
> grid()
```

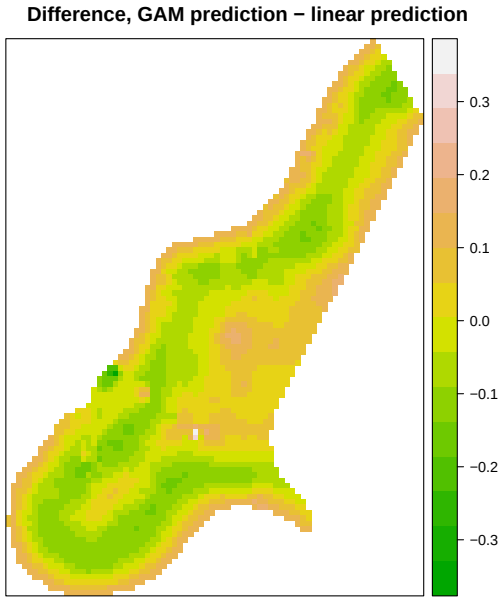


Q89 : *What is the anomaly at this observation?* *Jump to A89* •

Task 101 : Plot the difference between the GAM and linear model predictions. •

We compute the difference and add as a field to the prediction grid, then display with the usual spatial plot:

```
> meuse.grid$diff <- meuse.grid$k.g.i - meuse.grid$k.i
> print(spplot(meuse.grid, zcol="diff",
              col.regions=terrain.colors(64),
              main="Difference, GAM prediction - linear prediction"))
```



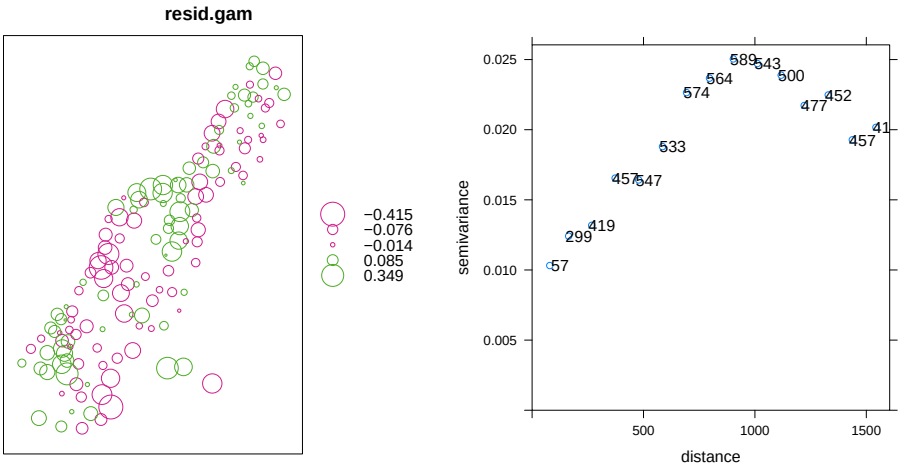
Q90 : *What are the differences between the two models? Which appears more realistic?* *Jump to A90* •

Challenge: Compare the GAM predictions with those from OK and KED.

Now that we've seen that the GAM does a fairly good job, let's see if there is any residual spatial correlation.

Task 102 : Display a bubble plot of the GAM model residuals and the residual variogram •

```
> meuse$resid.gam <- residuals(m.g.dist.elev.i)
> p1 <- bubble(meuse, zcol = "resid.gam", pch = 1)
> vr <- variogram(resid.gam ~ 1, locations = meuse)
> p2 <- plot(vr, plot.numbers = T)
> plot(p1, split = c(1, 1, 2, 1), more = T)
> plot(p2, split = c(2, 1, 2, 1), more = F)
```



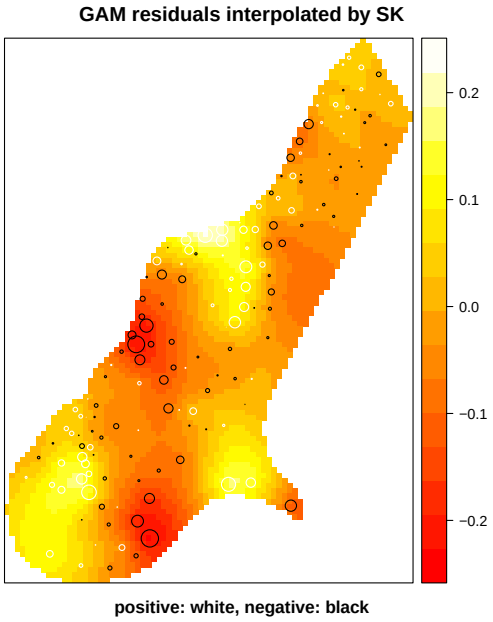
Q91 : *Do the residuals show spatial dependence?* *Jump to A91* •

Note however that most of the overall variability in $\log_{10}$Zn has been removed by the GAM; the proportion of the total remaining is less than 20%:

```
> max(vr$gamma)/max(variogram(logZn ~ 1, locations = meuse)$gamma)

[1] 0.1888
```

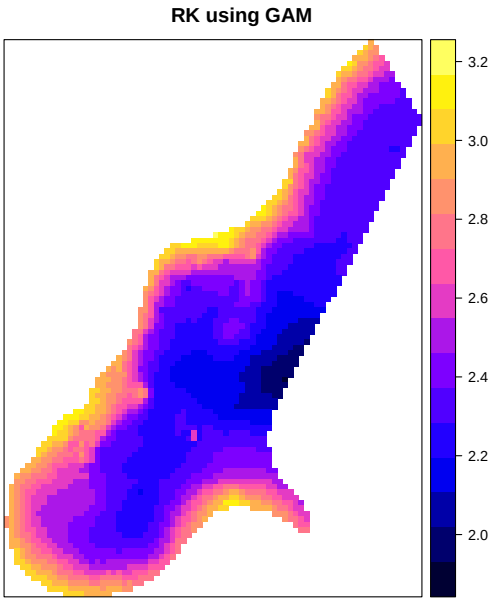
Challenge: Fit a variogram model to the GAM residuals, interpolate them over the grid by simple kriging (SK) with mean 0, add them to the GAM predictions. Hint: use the `beta` argument to `krige` to specify a known mean.

The Simple Kriging residuals look like this:



The Regression Kriging using the GAM and SK of the residuals look like

this:



14 Final words

There is much, much more to geostatistics than this simple introduction. Some good reference texts are by Goovaerts [10], Webster & Oliver [25] and Isaaks & Srivastava [14]. For spatial analysis in R, the essential reference is Bivand *et al.* [1]. For general modelling in R, a good text is Fox [9].

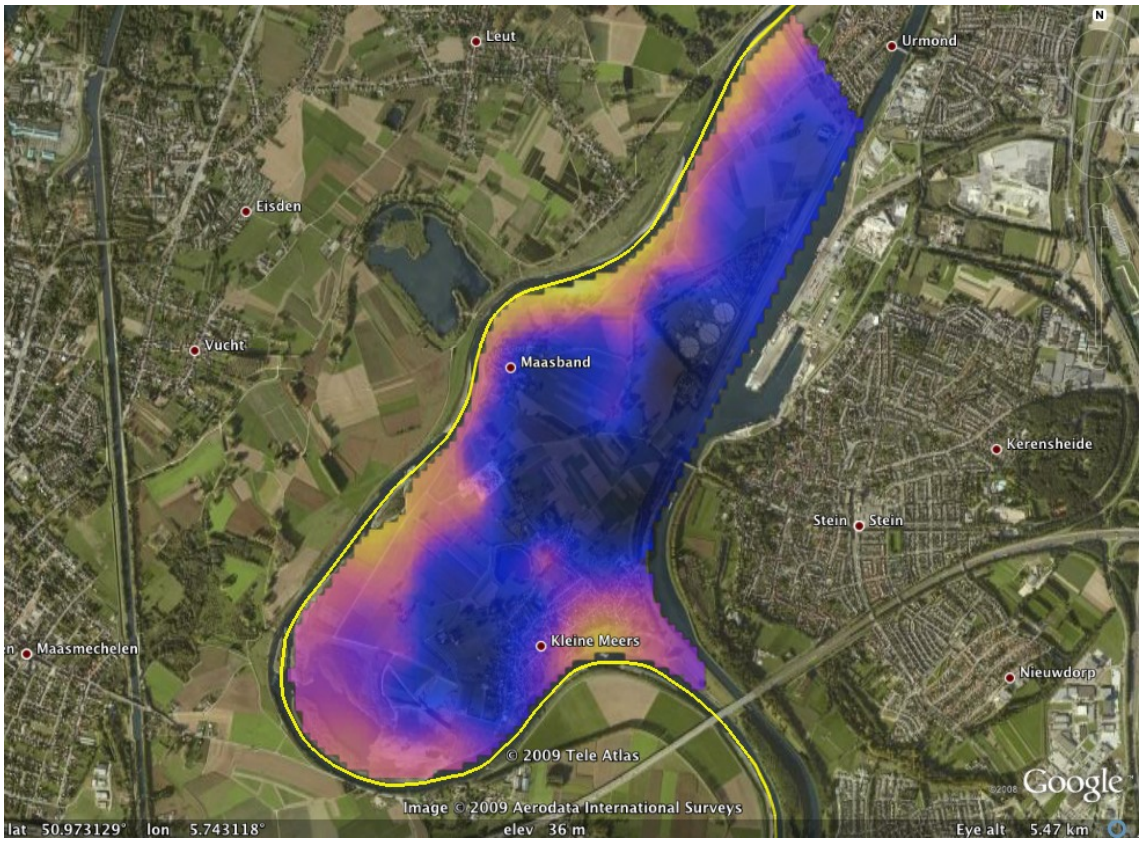
The approach to mixed modelling taken here, kriging with external drift (KED) (§11.3), has proven to perform well in many studies; however its theoretical basis is weak. This is because the feature-space model does not take into account the spatial correlation of the residuals, which is clearly present 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. Use generalized least-squares (GLS) instead of OLS to fit the linear model, considering the covariance structure of the residuals to assign weights.

   > **Note:** The `krige` function does use GLS for Universal Kriging/KED but it uses the covariance structure estimated from an OLS fit. A typical approach is to use the initial estimate of the spatial covariance (from the OLS model) to re-estimate the linear model parameters and then re-compute the residual spatial covariance; if this variogram model is close to the first estimate, the first KED estimate is good; if not, recompute the KED with the newly-estimated variogram model. This process can be iterated if necessary. But it is not theoretically-sound, unlike the next approach.

2. Fit the spatial covariance and linear model parameters at the same time using Restricted Maximum Likelihood (REML) [17].
3. Use model-based geostatistics [6] that explicitly include the feature space.

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:


Leut
Urmond
Eisden
Vucht
Maasband
Kerensheide
Stein
Stein
Maasmechelen
Kleine Meers
Nieuwdorp
© 2009 Tele Atlas
Image © 2009 Aerodata International Surveys
Google
lat 50.973129° lon 5.743118°
elev 36 m
Eye alt 5.47 km

15 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: 674.5, maximum: 1839, mean: 469.716129032258.* *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 but low to moderate Zn.*

Among the reasons could be: (1) field procedure error; (2) lab error; (3) recording error. However this was a well-conducted study and we have no reason to suspect these.

Another reason could be (4) use of Cu-containing agro-chemicals at these sites, so some of the Cu comes from polluted river water, along with the Zn, but some additional from the agro-chemical. We have no evidence for this since we do not know the land-use history. *Return to Q9* •

A10 : *Variance explained: 80.4%.* *Return to Q10* •

A11 : *The slope (gain) is 1.274 log_{10}(mg kg^{-1}), i.e., log_{10}Zn increases at a higher rate than log10-Cu. The standard error of this coefficient is 0.051, a small proportion of the coefficient.* *Return to Q11* •

A12 : *The histogram shows a normal shape except for the negative "tail", i.e., the log_{10}Zn values at a few points are strongly over-predicted by the model, leading to a large negative residual.* *Return to Q12* •

A13 : *Yes, there is some pattern. Medium values of log10-Zn, around 2.6 log_{10}(mg kg^{-1}), are on average slightly under-predicted, whereas at higher and (especially) lower values they are on average over-predicted. However this is not a strong pattern; most residuals are within a fairly narrow range* $[-0.2 \ldots +0.2]$ *log_{10}(mg kg^{-1}).* *Return to Q13* •

A14 : *The observation with ID 129 has the highest positive residual, i.e., its actual value is much higher than that predicted by log_{10}Cu*[23]*. This point does have a high Zn concentration, 703 mg kg^{-1}.* *Return to Q14* •

A15 : *No, and we saw this in the histogram of the residuals. The negative residuals are much more negative that would be expected by chance, if the residuals were normally distributed. This is because of the log_{10}Zn values at a few points that are strongly over-predicted by the model, due to the high log_{10}Cu values at these points.* *Return to Q15* •

A16 : *No, the high-leverage points do not have high Cook's distance, that is, they are consistent with the overall model fit.* *Return to Q16* •

A17 : *The model is in general fairly successful: it explains about 4/5 of the variance, the coefficients have small relative standard errors, and the residuals are mostly within a narrow range and mostly normally-distributed. Most of the poorly-fit points are over-predicted, so a pollution map made with them would be too cautious; however one high-Zn point is strongly under-predicted.* *Return to Q17* •

A18 : *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 Q18* •

A19 : *Variance explained: 24.3%* *Return to Q19* •

A20 : *To answer this, look at the summary table of the linear model. The number*

[23] This observation is row 123 in the data frame, but has been assigned a different observation ID, probably because some original observations were dropped from the dataset.

at the `(Intercept)` *row and* `Estimate` *column shows the best estimate for the mean of the first-listed class, in this case with the name* `1`*; this is Flood Frequency class 1, prediction 2.7* log_{10}*(mg kg*$^{-1}$*).*

The other two classes have the coefficients named `ffreq2` *and* `ffreq3`*; their* `Estimate` *is then the* difference *from the first-listed class (the intercept). So their mean concentrations are: class 2: 2.368* log_{10}*(mg kg*$^{-1}$*) (i.e., 2.7 + -0.332); class 3: 2.425* log_{10}*(mg kg*$^{-1}$*).* Return to Q20 •

A21 : *The variance explained is given by the* adjusted R^2*: 80.2%; this is a bit* less *than the same as that explained by Cu only: 80.4%, but much more than that explained by flooding frequency only: 24.3%.* Return to Q21 •

A22 : *There are 2 fewer degrees of freedom; these correspond to the two flooding frequency class differences from the first (base), which is equivalent to the intercept in the Cu-only model.*

The residual sum of squares is only reduced by 0.01 $log_{10}Zn^2$*.*

There is a quite high probability, 0.692, of a Type I error if the additive model is considered better than the single-predictor model. Thus we prefer the single-predictor model. Return to Q22
•

A23 : *Variance explained: 81.1%, a slight improvement over the additive and single-factor models. The residual sum of squares is reduced by 0.19* $log_{10}Zn^2$*. We have only a 0.043 probability of a Type I error if the interaction model is considered better than the single-predictor model.*

So the improvement is ***statistically significant*** *but* ***practically unimportant****. We conclude that a map of flooding frequency is not needed if we know the Cu at a location to be predicted.* Return to Q23 •

A24 : *The slope considering just the observations in flooding frequency class 2 (every 2–5 years) is considerably steeper than the others; this can also be seen in the model coefficient* `ffreq2:logCu`*.* Return to Q24 •

A25 : *The heavy metal comes from river flooding: closer distances, lower elevation and more frequent flooding are expected to increase its concentration in soil. If the metal came from air pollution, we would not expect any of these predictors to be important, except perhaps elevation if there would be local movement of surface soil. If the metal was from agricultural practices, these predictors would only be important if they would be correlated to the spatialization of agricuture.* Return to Q25 •

A26 : *The tree has 28 leaves and 27 internal nodes.* Return to Q26 •

A27 : *Distance to river is by far the most important, followed by elevation.*

Flooding frequency is unimportant. Return to Q27 •

A28 : *The model appears to be overfit; after 7 splits the cross-validation error increases. Note this may be different in different runs, because the random split of the full dataset for cross-validation will be different each time. There is very little reduction in this error between eight and nine splits, so eight seems to be the best choice for a parsimonious model.* Return to Q28 •

A29 : *The minimum cross-validation error is 0.287; this corresponds to 7 splits and a complexity parameter of 0.009.* Return to Q29 •

A30 : *The pruned tree has 6 leaves and 5 internal nodes. These are 21.4% and 18.5% of the original number, respectively. The tree is much smaller. with many fewer unique fitted values.* Return to Q30 •

A31 : *Only distance to river and elevation were used. This implies that any effect from flooding can be well-explained by distance and elevation; the recorded flooding frequency adds no useful information.* Return to Q31 •

A32 : *The first split is on distance: closer than 145 m to the river leads to much higher metal concentrations, on average, than further. Within each distance class elevation is then important. In both cases, lower elevations have substantially higher metal concentrations. The third split is different; one group is not split, two based on distance and one on elevation. The final split, for one third-level group, makes a small distinction based on elevation at the furthest distance (*> 230 *m). The interpretation is that the distance floodwater travels is the most important, and the elevation then separates deeper from shallower floods.* Return to Q32 •

A33 : *The parsimonious model only predicts 6 different values for the 155 points. Each predicted value is associated with a wide range of actual values. Thus the model is not very precise. It does, however, separate the highest and lowest values fairly well. This implies either missing predictors (but we did try flooding frequency, which did not improve the model) or local spatial variabiity not captured by these factors.* Return to Q33 •

A34 : *(1) The full model correctly classifies 27% of the observations.*

(2) It is minimum at 2.

(3) But, there is not a clear choice, because the more complicated trees do not differ much in their cross-validation error.

Return to Q34 •

A35 : *The model is not at all successful; it only explains about 20% of the classification. We can see this from the close proportions of different classes in most*

leaves. In particular, class 3 (rarely flooded) is never predicted. *Return to Q35* •

A36 : *Each run of* `randomForest` *will give different results; in most cases about 200 trees are needed to stabilize the out-of-bag MSE.* *Return to Q36* •

A37 : *As with the single regression tree, distance to river is the most important, followed by elevation; flooding frequency is much less important. However the variables are much closer to equally important here – in the single tree distance was responsible for about 58% of the single tree's succcess, whereas in the forest permuting elevation causes almost as much increase in error as permuting distance. Even permuting flood frequency causes a 9% increase in error; this implies that flooding frequency was used in some trees, whereas it was never used in the single regression tree.* *Return to Q37* •

A38 : *The random forest prediction is essentially continuous: each point has a separate prediction. By contrast, the tree only predicts one value for all points in the same "rectangle". The random forest fit seems more realistic.* *Return to Q38* •

A39 : *The out-of-bag RMSE is 0.153 log(mg kg^{-1}), about double the fits RMSE, 0.071 log(mg kg^{-1}). The out-of-bag RMSE is a more realistic estimate of the prediction error when applied at an unknown point, since when predicting at an unknown point, it could of course not be included in the model building. So this value is a more honest way to report the precision of the random forest model.* *Return to Q39* •

A40 : *Class 1 (annual flooding, red line) has the lowest rate and class 3 (rarely flooded, blue line) by far the most. For this class errors stabilize at about 80% incorrect allocation. None are good; even class 1 has about 1/4 of the observations incorrectly allocated.* *Return to Q40* •

A41 : *About 200 trees are needed.* *Return to Q41* •

A42 : *(1) Class 1: 67.7%; class 2: 56.8%; class 3: 27.8%. The least frequently flooded areas are often mapped as classes 1 and 2; this means if the map user trusts these as not often flooded, he or she will be facing flooding more frequently than expects.*

(2) Classes 1 and 2 are often confused and class 3 is very poorly mapped.

(3) This model is quite poor. One reason might be that the original records of flooding frequency are poor. Another is the geomorphology: an area can be flooded by concentrated flow even at a far distance and higher elevation. *Return to Q42* •

A43 : *Slots* `coords`, `coords.nrs`, `bbox`, `proj4string` *refer to the spatial structure; slot* `data` *to the feature-space (attribute) data. Note that the* `data` *slot has two fewer variables than before the conversion to a spatial object; these are the two*

coördinates which are now in the special `coords` *slot.* *Return to Q43* •

A44 : *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 Q44* •

A45 : *Yes, big circles tend to be near other big ones, same for small circles.* *Return to Q45* •

A46 : *There are (155 * (155-1))/2 = 11935 point-pairs.* *Return to Q46* •

A47 : *Separation is 70.84 m, semivariance is 0.001144 log(mg kg^{-1})^2.* *Return to Q47* •

A48 : *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 Q48* •

A49 : *The evidence is that at closer separations the semivariance is, on average, lower. This increases steadily until the range.* *Return to Q49* •

A50 : *At about 850 m separation there is not any reduction in average semivariance; this is the range.* *Return to Q50* •

A51 : *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 Q51* •

A52 : *At the range and beyond the average semivariance is about 0.13 log(mg kg^{-1})^2; this is the total sill.* *Return to Q52* •

A53 : *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 Q53* •

A54 : *The kriged map is very smooth. "Hot" and "cold" spots are clearly controlled by observations with especially high and low values.* *Return to Q54* •

A55 : *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 Q55* •

A56 : *Only 140 of the 155 observations above the threshold.* *Return to Q56* •

A57 : *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 Q57* •

A58 : *The total sill is 0.1432; units are dimensionless.* *Return to Q58* •

A59 : *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 Q59* •

A60 : *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. The variance is lowest in the annually flooded class, because this has more observations. *Return to Q60* •

A61 : *This empirical variogram has a shorter range (about 700 instead of about 950 m) and a lower total sill (about 0.08 instead of about 0.125 log(mg kg^{-1})^2; the estimated nugget is 0.01 log(mg kg^{-1})^2 about the same (indeed, theory requires that the nugget be exactly the same, since no model could remove noise at a point).* *Return to Q61* •

A62 : *Confirming the eyeball estimate: the range has reduced by -110 m (about 30%), the total sill by about 0.102 log(mg kg^{-1})^2 (about 25%). The modelled nugget is lower, although by theory it should not be any different.* *Return to Q62* •

A63 : *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 Q63* •

A64 : *Distance from river* `dist`, *flooding frequency class* `ffreq`, *and soil type* `soil`. *Return to Q64* •

A65 : *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 Q65* •

A66 : *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 Q66* •

A67 : *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 Q67* •

A68 : *Yes, the two-predictor models give significantly lower residual sum-of-squares.* *Return to Q68* •

A69 : *Yes, the interaction model has lower RSS than the additive model. The probability this is due to chance is only 0.0025.* *Return to Q69* •

A70 : *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 Q70* •

A71 : *Variance explained: 63.5%.* *Return to Q71* •

A72 : *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 Q72* •

A73 : *Yes, normally-distributed except for the three most positive residuals.* *Return to Q73* •

A74 : *The high-leverage residuals do not have high influence as shown by Cook's distance, this is good.* *Return to Q74* •

A75 : *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 Q75 •

A76 : *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 Q76 •

A77 : *Adding distance to river reduces the prediction variances and makes them almost uniform across the area.* Return to Q77 •

A78 : *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: 0.1631. Return to Q78 •

A79 : *The one-factor KED prediction, with flood frequency as co-variable, is most precise; RMSE is 0.141 log(mg kg^{-1}).* Return to Q79 •

A80 : *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 Q80 •

A81 : *The metal comes from flood water. Concentrations are higher where there has been more flooding and for longer times. But it's not the water, it's the sediment in the water, that carries the metals. As water flows over the flooded areas it drops sediment, so we expect higher concentrations nearer the rivers. Less water covers higher elevations, and less frequently, so we expect lower concentrations at high elevations.* Return to Q81 •

A82 : *Neither relation looks linear. The relation with distance appears to be inverse linear at short range, but then inverse squared at medium range and almost constant at long range. The relation with elevation appears to be very noisy but constant at short range, and then inverse linear at medium and long ranges. Perhaps a higher-order polynomial could fit these.* Return to Q82 •

A83 : *Both models are useful; distance (R^2 = 0.655) more so than elevation (R^2 = 0.451). Also the range of residuals and the inter-quartile range is much narrower when distance is the predictor.* Return to Q83 •

A84 : *The fits are smooth functions of the noisy data. They match well with*

the hypothesis. However it's clear that there is a lot of variability not explained by either of these. *Return to Q84* •

A85 : *Elevation is weakly related to distance: higher elevations tend to be further from the river, but there is quite a spread at all distances. The correlation coefficients, both parametric and non-parametric, show that about 25% of the variability is common between the two predictors. Thus the two predictors are mostly independent.* *Return to Q85*
•

A86 : *Both models are better than the single-factor models. The model without interaction explains $R^2 = 0.77$ of the variance, the model with interaction explains $R^2 = 0.785$. The residuals are also considerably smaller than those from the single-factor models. The interaction term is significant but does not add much to the overall fit.* *Return to Q86* •

A87 : *The interaction term (tensor) makes the two marginal smoothers (distance and elevation) more regular; in particular it removes the anomaly at the lowest and highest elevations, and the "hump" at medium-long distances.* *Return to Q87* •

A88 : *The GAM prediction clearly mostly depends on the distance from river, with some adjustments for elevation.* *Return to Q88* •

A89 : *This point has much higher Zn concentration than predicted, and quite different from nearby points. It is medium distance from the river and at a moderately low elevation, but has a concentration similar to points a shorter distance back from the river. Notice how the GAM fit is much better at this point. This is because it is not forced to predict with a single linear model over the whole range of the predictors.* *Return to Q89* •

A90 : *The GAM predicts higher near the river and in the highest area (E of map). It predicts lower in the middle. Thus the smoothly-adjusting fit of the GAM matches the pattern better than the linear model.* *Return to Q90* •

A91 : *Yes, there is strong spatial dependence.* *Return to Q91* •

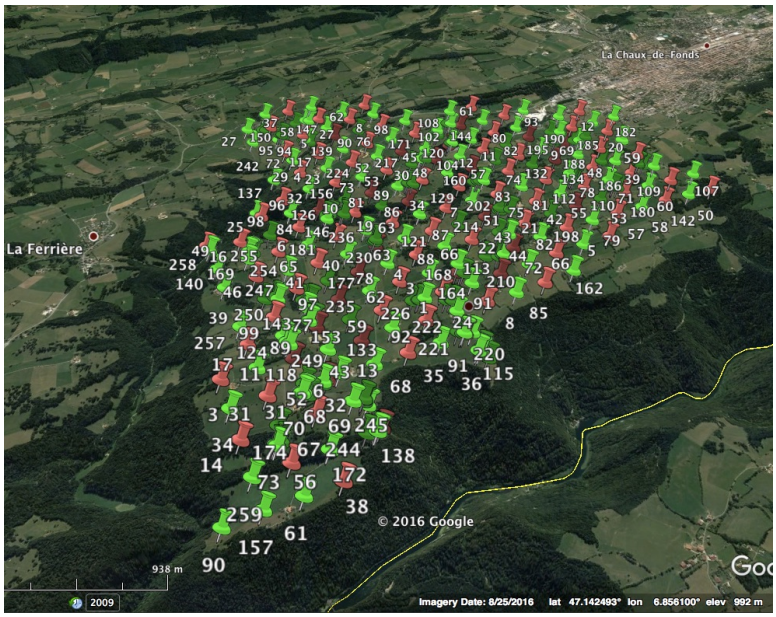
16 Assignment

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. This includes several workspace objects. Examine the structure of the `jura.pred` dataframe; this is the "calibration" dataset. •

For your information, here is a perspective view of the calibration (green) and evaluation (red) datasets, on the landscape near La Chaux-de-Fonds (CH).



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

Task 2 : Display histograms of the copper (Cu) concentration at the points in the calibration dataframe, and its log transform. •

Q2 : *Describe the two distributions.* •

Task 3 : Model the log(Cu) concentration at the points in the calibration

dataframe as a linear function of rock type and land use separately, and then additively. •

Q3 : *Why might these be reasonable (real-world) predictors of Cu concentration? In other words, why might land use and/or rock type affect Cu concentration in the soil?* •

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 for the additive model: (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 : Display a cross-classification table of the two factors (rock type and land use) using the `table` function. •

Q6 :

(1) Are all classes of each factor represented more or less equally?

(2) Are the two factors independent? That is, are the numbers of observations in each cross-classification cell as expected from the marginal totals of each factor?

***Hint**: use the `outer` 'array outer product' function on the two one-way tables to make a table of the expected values of the two-way table.*

(3) Is it advisable to build an interaction model, i.e., to look for synergies between the two factors in explaining the log(Cu) concentrations? •

Task 6 : Build a regression tree to model the log(Cu) concentration modelled by rock type, land use, and the two coördinates (`Xloc` and `Yloc`).

1. Build the tree with default parameters;

2. Display the tree;
3. Print the variable importance;
4. Print and plot the cross-validation error vs. the complexity parameter;
5. Prune the tree back to the appropriate complexity parameter and display it.

•

Q7 :

(1) Which variables were used in the tree? How many leaves does it have?

(2) Which variable was most important?

(3) What is the appropriate complexity parameter with which to prune the tree? Why?

(4) Which variables were used in the pruned tree? How many leaves does it have? •

Task 7 : Build a random forest to model the log(Cu) concentration modelled by rock type, land use, and the two coördinates (`Xloc` and `Yloc`). •

Q8 :

(1) How much of the variance in log(Cu) is explained by the random forest?

(2) Which variables are most important?

(3) How does the RMSE of the out-of-bag cross-validation compare to the mean value of the variable being modelled?

(4) Do you consider this model successful? •

Optional: predict with the regression tree and random forest over the prediction grid and display the maps. (This will require predicting in the data frame, and then converting to a spatial grid for display.)

Task 8 : Convert the calibration dataframe `jura.pred` into a spatial object, using the local metric coördinates(fields `Xloc` and `Yloc`, see `?jura`). Add the model residuals from your best model from Task 3, as determined in previous steps, as a field, and display a bubble plot of the residuals. •

Q9 : *Does there appear to be spatial dependence in the residuals?* •

Task 9 : 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). •

Q10 : *Describe the empirical variogram qualitatively. What are the approximate partial (structural) sill, range, and nugget variance?* •

Task 10 : Fit a spherical variogram model to the empirical variogram. •

Q11 : *What are the fitted partial (structural) sill, range, and nugget variance?* •

Task 11 : **Optional**: Check the robustness of this fit to the noisy empirical variogram by fitting to different bin widths and comparing the fitted variogram parameters. •

Q12 : ***Optional**: How robust is the variogram fit?* •

Task 12 : Convert the prediction grid dataframe `jura.grid`, which was loaded into the workspace with the `jura` dataset, into a gridded spatial object, and display maps of the two covariates (land use and rock type). •

Task 13 : Predict over the prediction grid by Kriging with External Drift (KED) from the calibration points, using the fitted variogram model of the linear model residuals. Plot the kriging predictions and their variances (or standard deviations). •

Q13 : *Where are the highest and lowest predicted concentrations? How do these relate to the covariables (if at all)?* •

Q14 : *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 14 : Cross-validate the KED predictions at the calibration points set `jura.pred`, summarize the cross-validation statistics, and plot the cross-validation residuals. •

Q15 : *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/ 5, 118

[2] Breiman, L.; Friedman, J. H.; Olshen, R. A.; & Stone, C. J. 1983. *Classification and regression trees*. Wadsworth 31

[3] Brus, D.; de Gruijter, J.; Walvoort, D.; Bronswijk, J.; Romkens, P.; de Vries, F.; & de Vries, W. 2002. *Mapping the risk of exceeding critical thresholds for cadmium concentrations in soils in the Netherlands*. *Journal of Environmental Quality* **31**:1875–1884 140

[4] Brus, D.; Kempen, B.; & Heuvelink, G. 2011. *Sampling for validation of digital soil maps*. *European Journal of Soil Science* **62**:394–407. ISSN 13510754 96

[5] Cook, R. & Weisberg, S. 1982. *Residuals and influence in regression*. New York: Chapman and Hall 21, 88

[6] Diggle, P. J. & Ribeiro Jr., P. J. 2007. *Model-based geostatistics*. Springer. ISBN 987-0-387-32907-9 118

[7] Efron, B. & Gong, G. 1983. *A leisurely look at the bootstrap, the jackknife & cross-validation*. *American Statistician* **37**:36–48 46

[8] Fox, J. 1997. *Applied regression, linear models, and related methods*. Newbury Park: Sage 20, 21, 88

[9] Fox, J. 2002. *An R and S-PLUS Companion to Applied Regression*. Newbury Park: Sage 118

[10] Goovaerts, P. 1997. *Geostatistics for natural resources evaluation*. Applied Geostatistics. New York; Oxford: Oxford University Press 118

[11] Hastie, T.; Tibshirani, R.; & Friedman, J. H. 2009. *The elements of statistical learning data mining, inference, and prediction*. Springer series in statistics. New York: Springer, 2nd ed edition. ISBN 9780387848587 31, 41, 45, 46, 100

[12] Hengl, T. 2009. *A Practical Guide to Geostatistical Mapping*. Amsterdam. ISBN 978-90-9024981-0
URL http://spatial-analyst.net/book/ 139

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

[14] Isaaks, E. H. & Srivastava, R. M. 1990. *An introduction to applied geostatistics*. New York: Oxford University Press 118

[15] James, G.; Witten, D.; Hastie, T.; & Tibshirani, R. 2013. *An introduction to statistical learning: with applications in R*. Number 103 in Springer texts in statistics. Springer. ISBN 9781461471370 31, 41, 46, 100

[16] Lark, R. M. 1995. *Components of accuracy of maps with special reference to discriminant analysis on remote sensor data. International Journal of Remote Sensing* **16**(8):1461–1480 53

[17] 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):799–813 118

[18] Pebesma, E. J. 2004. *Multivariable geostatistics in S: the gstat package. Computers & Geosciences* **30**(7):683–691 5

[19] Pebesma, E. J. & Bivand, R. S. 2005. *Classes and methods for spatial data in R. R News* **5**(2):9–13
URL http://CRAN.R-project.org/doc/Rnews/ 3

[20] 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

[21] R Core Team. 2015. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria
URL https://www.R-project.org 3

[22] R Development Core Team. 2015. *R Data Import/Export*. The R Foundation for Statistical Computing, version 3.2.2 (2015-08-14) edition
URL http://cran.r-project.org/doc/manuals/R-data.pdf 6

[23] 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 139

[24] Shalizi, C. 2010. *The bootstrap. American Scientist* **98**(3):186–190
URL http://www.americanscientist.org/issues/pub/2010/3/the-bootstrap/3 46

[25] Webster, R. & Oliver, M. A. 2008. *Geostatistics for environmental scientists*. John Wiley & Sons Ltd., 2nd edition 118

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 [23].

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; the EPSG code for this system is 28992
`cadmium`	Cd concentration in the soil, in weight mg kg^{-1}; zero cadmium values have been shifted to 0.2 (half the lowest non-zero value, likely the detection limit)
`copper`	Cu concentration in the soil, in mg kg^{-1}
`lead`	Pb concentration in the soil, in mg kg^{-1}
`zinc`	Zn 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 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: once in 10 years, 3: once in 50 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

Metals were determined by digestion in 25% HNO_3 followed by atomic absorption spectroscopy.

Related datasets Tomislav Hengl has extended the Meuse dataset[24] for his " Practical Guide to Geostatistical Mapping" [12]; this includes a digital

[24] http://spatial-analyst.net/book/meusegrids

elevation model with cm vertical resolution obtained from the LiDAR survey of the Netherlands, and a 2 m vertical resolution contour map from the topographic survey of the Netherlands.

Soil pollution thresholds According to the Berlin Digital Environmental Atlas[25], 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.

There is much more to soil pollution risk than a simple threshold; the path to the human must be specified and modelled. See for example the study on risk for Cd in Dutch soils by Brus *et al.* [3].

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