An introduction to (geo)statistics with R

D G Rossiter

Cornell University, Section of Soil & Crop Sciences

January 19, 2024

Contents

1 Installing R and RStudio **1**
1.1 Installing contributed packages 2

2 First interaction with R **3**
2.1 Using a script 3
2.2 Loading packages 4

3 Loading and examining the Meuse dataset **6**

4 * Taking a break and re-starting **9**

5 Non-spatial univariate EDA and transformation **11**
5.1 Transformation 13

6 Non-spatial bivariate EDA **15**

7 Model-based feature-space modelling **18**
7.1 Theory of linear models 19
7.1.1 * Ordinary Least Squares (OLS) solution of the linear model 20
7.2 Continuous response, continuous predictor 21
7.2.1 Model summary 22
7.2.2 Model diagnostics 23
7.3 Continuous response, categorical predictor 28
7.4 * Multivariate linear models 31
7.4.1 Additive linear model 31

7.4.2 Interaction linear model 33

8 Spatially-explicit R objects 34

9 Data-driven feature-space modelling 40
9.1 Regression trees 40
9.1.1 Predicting over the study area with the regression tree model 51
9.2 * Classification trees 54
9.3 Random forests for continuous predictands 59
9.3.1 Building and evaluating a random forest model . . . 60
9.3.2 * A deeper investigation of variable importance . . . 63
9.3.3 Prediction with the random forest 67
9.3.4 Predicting over the study area with the random forest model 69
9.4 * Random forests for categorical predictands 70

10 Local spatial structure 73
10.1 Assessing spatial dependence with the empirical variogram 73
10.1.1 * The variogram cloud 75
10.1.2 The empirical variogram 77
10.2 Building a variogram model 79

11 Mapping by Ordinary Kriging interpolation 82
11.1 Theory of Ordinary Kriging 82
11.2 Ordinary kriging on a regular grid 85

12 * Non-parameteric methods: Indicator kriging 90

13 Mixed predictors 94
13.1 Feature-space prediction 94
13.2 The residual variogram 97
13.3 Prediction by Kriging with External Drift (KED) 99
13.3.1 Displaying several maps on the same scale 100
13.3.2 KED Prediction variances 101
13.4 A mutivariate mixed predictor 103
13.4.1 Linear model diagnostics 108
13.4.2 Spatial structure of the the residuals 111
13.4.3 KED prediction 113
13.4.4 KED prediction variances 115

14 * Generalized Least Squares 117
14.1 * GLS – theory 117
14.2 GLS – practice 119
14.3 GLS prediction 124
14.4 GLS-RK 125

15 Model evaluation 129
15.1 Independent evaluation set 130
15.2 Cross-validation 130

16 * Generalized Additive Models 133
17 Final words 149
18 Answers 150
19 Assignment 162
References 169
Index of R concepts 170
A The Meuse dataset 173

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** [15, 24] and several R packages, notably `sf` [20] for spatial data structures, `gstat` [22] for conventional geostatistics, `rpart` for classification and regression trees (CART), and `ranger` [29] for random forests.

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[1] 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**. §19 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 §17, which you should consult before undertaking your own geostatistical analyses.

> **Note:** The source for this document is a text file that includes ordinary LaTeX source and "chunks" of R source code, using the Noweb[2] syntax. The formatted R source code, R text output, and R graphs in this document were automatically generated and incorporated into a LaTeX source file by running the Noweb source document through R, using the `knitr` package [30]. The LaTeX source was then compiled by LaTeX into the PDF you are reading.

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] `http://www.rstudio.org`
[2] `http://www.cs.tufts.edu/~nr/noweb/`

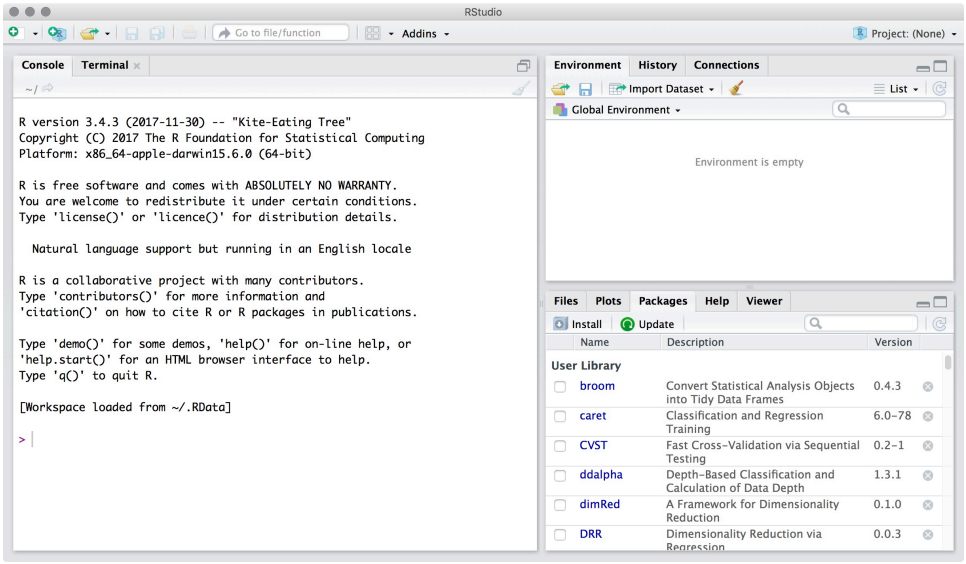


Figure 1: RStudio screen after installation

1.1 Installing contributed packages

In this exercise we will use several of the thousands of contributed R packages, in addition to the standard packages supplied with a fresh installation of R.

To get new packages, use the "Install" toolbar button of the "Packages" tab in RStudio, and ask that packages `sf`[3], the `sp`[4], the `gstat`[5] and the `ggplot2`[6] packages be installed, along with their dependencies.

Check the "Install dependencies" box; this will first install any packages on which the selected packages depend, and then the selected packages.

Installation only needs to be done once for each (group of) package(s), and then the package(s) are on your system available for use.

The first time you install new packages, you mayl be asked to specify a "mirror", i.e., one of the many servers around the world from which R and packages can be 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 – **do not do this** if you've already installed the packages using the interactive dialogue (above). To install the required packages and their dependencies, 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("sf", "sp", "gstat", "ggplot2"), dependencies=TRUE)
> ```

[3] "Simple Features for R"
[4] "Classes for spatial data"
[5] "Spatial and Spatio-Temporal Geostatistical Modelling, Prediction and Simulation"
[6] "Grammar of Graphics"

As you use R, you should periodicially check for package updates using the "Update" toolbar button of the "Packages" tab in RStudio.

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] 0.01745329
```

This means: when R prompts for input:

1. You type an expression, in this case `2*pi/360`, and press the "Enter" key.
   Do *not* type the > prompt; this is from the R system, telling you it is ready for input.
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.

> **Note:** If you type a command that is not complete, the R console will prompt you for more input with a + "continuation" prompt. If you made a mistake input, just press the `Esc` key and start over.

Challenge: At the R command line, write and execute an expression to compute the number of seconds in a standard, 365-day, year. How many are there?

2.1 Using a script

Although you can use R from the console, it is much better to write your R code in an **R script**. This is a text file, conventionally saved with file extension `.R`, with a list of R commands.

Task 2 : Create a new R script, with the `File | New File | R Script` menu item, or the icon at the top left of the toolbar; see Figure 2. •

Task 3 : Enter the R code from the previous § in the script, and execute

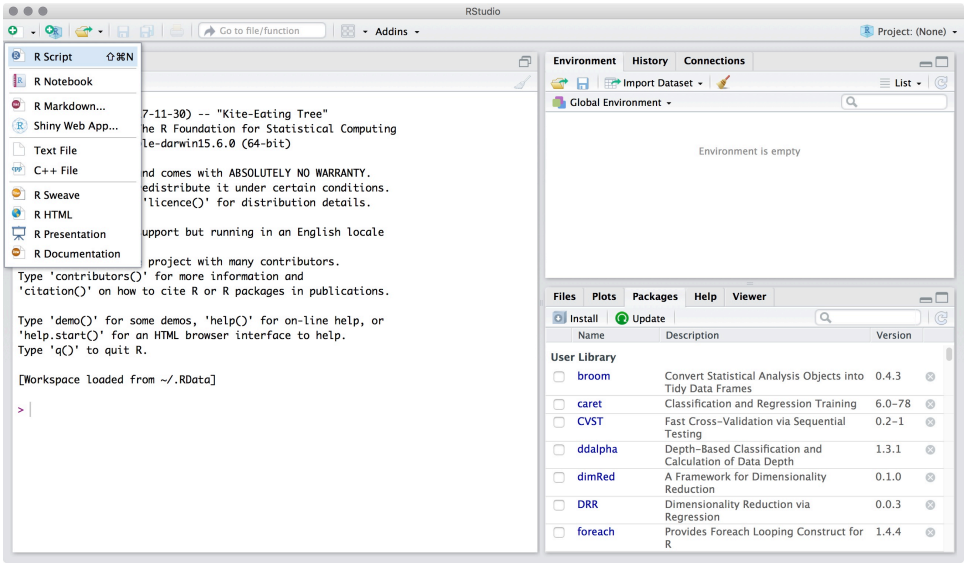


Figure 2: Creating a new R script in RStudio

these commands in the console by selecting them and pressing the `Run` button in the toolbar, or by pressing `Command + Return` (Windows) or `Command + Return` (Mac). The code will be executed, and the results will be shown in the Console. See Figure 3. •

TASK 4 : Save the R script, with the `File | Save` menu item or the small disk icon in the toolbar of the script window. •

The default file extension, which you should not change, is `.R`.

For the remainder of the tutorial, write your R code in a script (you can use several, to break up the work, if you wish), and execute the code as shown above.

2.2 Loading packages

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. We installed two of these (`sf` and `gstat`) with their dependcies in §1, above. In this exercise we will use the `gstat` package for geostatistical modelling, prediction and simulation, contributed by Pebesma [21] and the `sf` package for representing spatial data.

TASK 5 : Load the `sf` 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 4.

You only need to load packages one time in each session, although it will

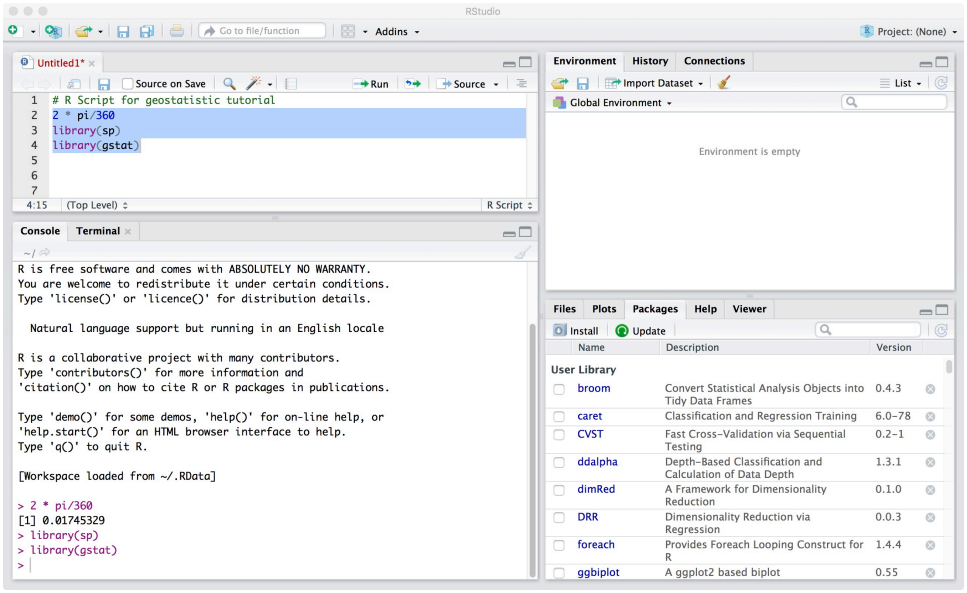


Figure 3: Executing commands from an R script in RStudio

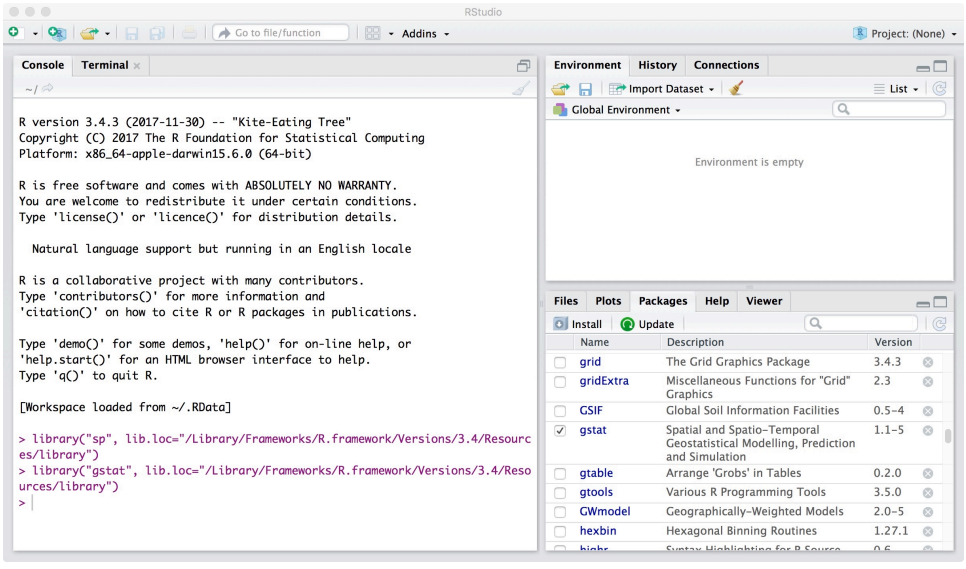


Figure 4: Loading installed Packages with RStudio

not cause any problems if you load them more than once.

You can also load packages from the R console with the `library` function:

```
library(sf)
```

```
Linking to GEOS 3.11.0, GDAL 3.5.3, PROJ 9.1.0; sf_use_s2() is TRUE
```

```
library(gstat)
```

The `require` function is almost identical to `library`.

You can see the loaded packages with the `search` "search path" function:

```
search()
```

```
 [1] ".GlobalEnv"        "package:gstat"     "package:sf"
 [4] "package:knitr"     "ESSR"              "package:stats"
 [7] "package:graphics"  "package:grDevices" "package:utils"
[10] "package:datasets"  "package:methods"   "Autoloads"
[13] "package:base"
```

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 topsoil samples which were analyzed for their **concentration of toxic heavy metals**, along with the sample **location**.

The `data` function displays the built-in datasets available with currently-loaded packages:

```
data()
```

```
Data sets in package `datasets':

AirPassengers          Monthly Airline Passenger Numbers 1949-1960
BJsales                Sales Data with Leading Indicator
BJsales.lead (BJsales) Sales Data with Leading Indicator
BOD                    Biochemical Oxygen Demand
CO2                    Carbon Dioxide Uptake in Grass Plants
ChickWeight            Weight versus age of chicks on different diets
...
```

The datasets in one package can be shown with the optional `package` argument:

```
data(package="sp")
```

```
Data sets in package `sp':

Rlogo           Rlogo jpeg image
gt (Rlogo)      Rlogo jpeg image
meuse           Meuse river data set
meuse.area      River Meuse outline
meuse.grid      Prediction Grid for Meuse Data Set
meuse.grid_ll   Prediction Grid for Meuse Data Set, geographical coordinates
meuse.riv       River Meuse outline
```

Challenge: Display the datasets available in the `gstat` package.

> **Note:** R also provides functions for loading data from external sources, e.g., `read.table` to read delimited text files and and `read.csv` to read comma-separated values (CSV) text files. In this exercise we will not show to how to load your own datasets; see the R Data Import/Export Manual [25].

TASK 6 : Load the `meuse` dataset into the workspace. •

The `data` function also loads a named dataset. We don't need to load the `sp` package just to access one of its example datasets, if we use the optional `package` argument to the `data` function. We show the contents of the workspace before and after with the `ls` "list objects" function:

```
ls()

character(0)

data("meuse", package = "sp")
ls()

[1] "meuse"
```

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

TASK 7 : 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**.

Note that the matrix is display rotated: the vertical list of data fields is actually the **columns** of the underlying matrix, and the horizontal lists of values are per-row values. We can see this by displaying the data frame as a matrix, using the `as.matrix` "convert to matrix" function:

```
dim(meuse)

[1] 155  14

dim(as.matrix(meuse))

[1] 155  14

str(as.matrix(meuse))

 chr [1:155, 1:14] "181072" "181025" "181165" "181298" "181307" ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:155] "1" "2" "3" "4" ...
  ..$ : chr [1:14] "x" "y" "cadmium" "copper" ...

head(as.matrix(meuse))

  x        y        cadmium copper lead  zinc   elev     dist
1 "181072" "333611" "11.7"  " 85"  "299" "1022" " 7.909" "0.00135803"
2 "181025" "333558" " 8.6"  " 81"  "277" "1141" " 6.983" "0.01222430"
3 "181165" "333537" " 6.5"  " 68"  "199" " 640" " 7.800" "0.10302900"
4 "181298" "333484" " 2.6"  " 81"  "116" " 257" " 7.655" "0.19009400"
5 "181307" "333330" " 2.8"  " 48"  "117" " 269" " 7.480" "0.27709000"
6 "181390" "333260" " 3.0"  " 61"  "137" " 281" " 7.791" "0.36406700"
  om     ffreq soil lime landuse dist.m
1 "13.6" "1"   "1"  "1"  "Ah"     "  50"
2 "14.0" "1"   "1"  "1"  "Ah"     "  30"
3 "13.0" "1"   "1"  "1"  "Ah"     " 150"
4 " 8.0" "1"   "2"  "0"  "Ga"     " 270"
5 " 8.7" "1"   "2"  "0"  "Ah"     " 380"
6 " 7.8" "1"   "2"  "0"  "Ga"     " 470"
```

The `dim` "dimensions" function shows the matrix dimensions; the `head` function shows the first few lines of large objects.

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 8 : 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 5 is a Google Earth view of the Meuse observation points, with their Zn concentrations[7]. 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.

Another way to examine any R object is with the `summary` method. This shows summary information appropriate to the type of R object.

TASK 9 : Display the summary of the Meuse dataframe. •

```
summary(meuse)
```

```
       x                y              cadmium          copper      
 Min.   :178605   Min.   :329714   Min.   : 0.200   Min.   : 14.00  
 1st Qu.:179371   1st Qu.:330762   1st Qu.: 0.800   1st Qu.: 23.00  
 Median :179991   Median :331633   Median : 2.100   Median : 31.00  
 Mean   :180005   Mean   :331635   Mean   : 3.246   Mean   : 40.32  
 3rd Qu.:180630   3rd Qu.:332463   3rd Qu.: 3.850   3rd Qu.: 49.50  
 Max.   :181390   Max.   :333611   Max.   :18.100   Max.   :128.00  

      lead             zinc            elev             dist        
 Min.   : 37.0   Min.   : 113.0   Min.   : 5.180   Min.   :0.00000  
 1st Qu.: 72.5   1st Qu.: 198.0   1st Qu.: 7.546   1st Qu.:0.07569  
 Median :123.0   Median : 326.0   Median : 8.180   Median :0.21184  
 Mean   :153.4   Mean   : 469.7   Mean   : 8.165   Mean   :0.24002  
 3rd Qu.:207.0   3rd Qu.: 674.5   3rd Qu.: 8.955   3rd Qu.:0.36407  
 Max.   :654.0   Max.   :1839.0   Max.   :10.520   Max.   :0.88039  

       om          ffreq   soil    lime        landuse       dist.m      
 Min.   : 1.000   1:84    1:97    0:111   W       :50   Min.   :  10.0  
 1st Qu.: 5.300   2:48    2:46    1: 44   Ah      :39   1st Qu.:  80.0  
 Median : 6.900   3:23    3:12            Am      :22   Median : 270.0  
 Mean   : 7.478                           Fw      :10   Mean   : 290.3  
 3rd Qu.: 9.000                           Ab      : 8   3rd Qu.: 450.0  
 Max.   :17.000                           (Other):25   Max.   :1000.0  
 NA's   :2                                NA's    : 1                   
```

Q5 : *What are the minimum, median and maximum concentrations of copper in the topsoil?* *Jump to A5* •

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. If you want to continue on now, just jump to §5.

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`.

[7] source: Hengl [14]

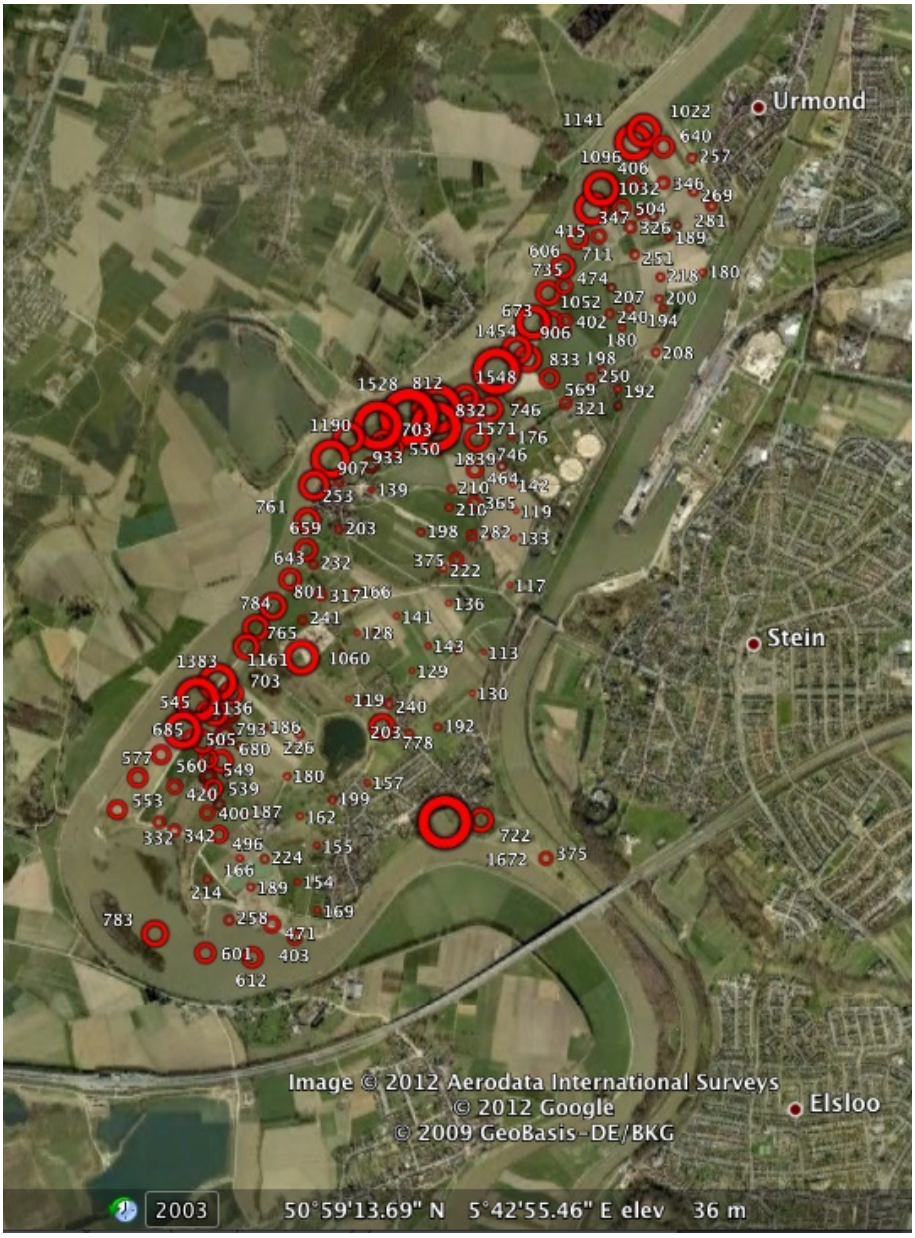


Figure 5: Meuse sample points, with zinc concentrations, shown in Google Earth

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

> **Note:** By default Windows and Mac OS/X do not show file extensions, and so the `.Rdata` file is not visible in the file manager.

When you are ready to continue:

TASK 10 : 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.

Any R scripts that were shown in the Scripts window should also be automatically re-loaded; if not, re-load with the `File | Open File...` or `File | Open Recent` menu items.

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

```
library(sf)
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 11 : Display the actual data values for zinc (Zn) content, both in sample and sort order. •

the $ field separator

One way to reference a field in a dataframe is with 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 1096
 [14]  504  326 1032  606  711  735 1052  673  402  343  218  200  194
 [27]  207  180  240  180  208  198  250  192  213  321  569  833  906
 [40] 1454  298  167  176  258  746  746  464  365  282  375  222  812
 [53] 1548 1839 1528  933  432  550 1571 1190  907  761  659  643  801
 [66]  784 1060  119  778  703  676  793  685  593  549  680  539  560
 [79] 1136 1383 1161 1672  765  279  241  317  545  505  420  332  400
 [92]  553  577  155  224  180  226  186  198  187  199  157  203  143
[105]  136  117  113  130  192  240  221  140  128  166  191  232  203
[118]  722  210  198  139  253  703  832  262  142  119  152  415  474
```

```
[131]  126  210  220  133  141  158  129  206  451  296  189  154  169
[144]  403  471  612  601  783  258  214  166  496  342  162  375
```

```
sort(meuse$zinc)
```

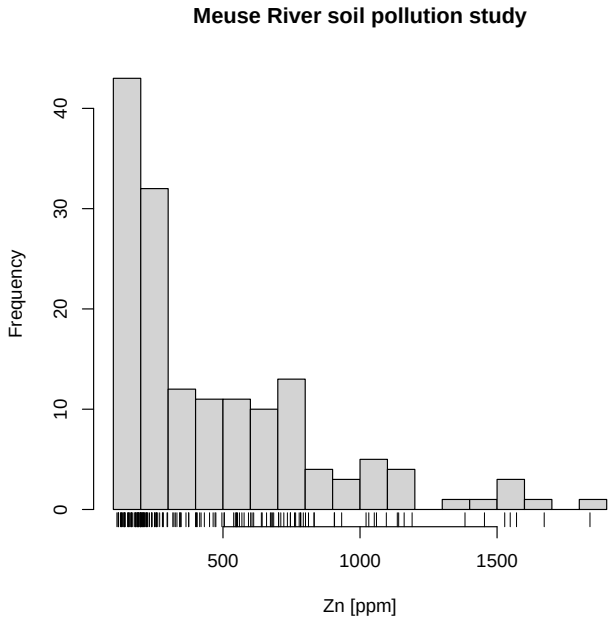
```
  [1]  113  117  119  119  126  128  129  130  133  136  139  140  141
 [14]  142  143  152  154  155  157  158  162  166  166  167  169  176
 [27]  180  180  180  183  186  187  189  189  191  192  192  194  198
 [40]  198  198  199  200  203  203  206  207  208  210  210  213  214
 [53]  218  220  221  222  224  226  232  240  240  241  250  251  253
 [66]  257  258  258  262  269  279  281  282  296  298  317  321  326
 [79]  332  342  343  346  347  365  375  375  400  402  403  406  415
 [92]  420  432  451  464  471  474  496  504  505  539  545  549  550
[105]  553  560  569  577  593  601  606  612  640  643  659  673  676
[118]  680  685  703  703  711  722  735  746  746  761  765  778  783
[131]  784  793  801  812  832  833  906  907  933 1022 1032 1052 1060
[144] 1096 1136 1141 1161 1190 1383 1454 1528 1548 1571 1672 1839
```

TASK 12 : Display a **histogram** and a **five-number summary** of the Zn content. Show the individual values of Zn with a "rug" plot under the histogram. •

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

```
hist(meuse$zinc, breaks=16, main="Meuse River soil pollution study",
     xlab="Zn [ppm]")
rug(meuse$zinc)
summary(meuse$zinc)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  113.0   198.0   326.0   469.7   674.5  1839.0
```



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

Q6 : *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 A6*
•

Q7 : *What are the minimum, first quartile, median, third quartile, and maximum Zn concentrations in the sample set?* *Jump to A7* •

Q8 : *Compare the mean and median. What does this imply about the distribution of the variable?* *Jump to A8* •

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.

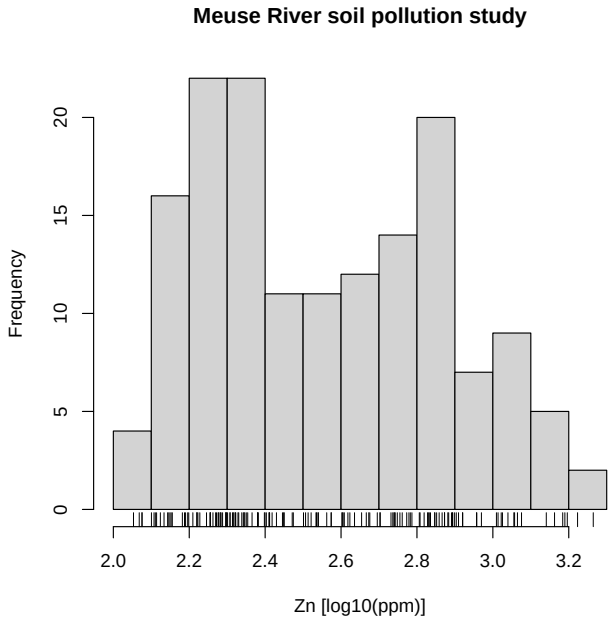
TASK 13 : Log-transform the variable `zinc` and save it as a new field in the data frame.

Then, repeat repeat the summary, histogram, and related questions.

Use base-10 logarithms, as they are easy to understand in the original units (e.g. if $\log_{10}\text{Zn} = 3$, then $\text{Zn} = 10^3 = 1000$). . •

```
meuse$logZn <- log10(meuse$zinc)
hist(meuse$logZn, breaks=16, main="Meuse River soil pollution study",
     xlab="Zn [log10(ppm)]")
rug(meuse$logZn)
summary(meuse$logZn)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.053   2.297   2.513   2.556   2.829   3.265
```



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] 3.009451 3.057286 2.806180 2.409933 2.429752 2.448706
```

Q9 : *Does the transformation make the variable more symmetric? Does it remove presumed outliers? Is there now evidence for more than one population?* *Jump to A9* •

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 (Cu) concentration.

Task 14 : Log-transform the Cu 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.

Challenge: Display a histogram of the point elevations above local river base level[8]. Display the numeric summary. Discuss the form of the distribution. Are there any unusual or extreme values?

6 Non-spatial bivariate EDA

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

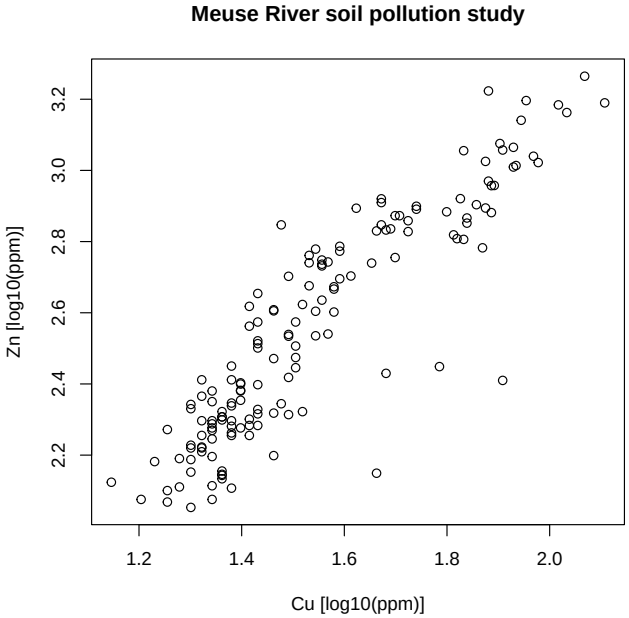
TASK 15 : Show a **scatterplot** of the relation between log-transformed Zn and Cu. •

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,
     xlab = "Cu [log10(ppm)]",
     ylab = "Zn [log10(ppm)]",
     main = "Meuse River soil pollution study")
```

[8] See `?meuse` for a description of the attributes



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.

Q10 : *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 A10* •

TASK 16 : 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  29
[17]  30  31  32  33  34  35  36  41  42  43  44  48  49  50  51  68
[33]  84  85  86  90  94  95  96  97  98  99 100 101 102 103 104 105
[49] 106 107 108 109 110 111 112 113 114 115 116 117 119 120 121 122
[65] 125 126 127 128 131 132 133 134 135 136 137 138 140 141 142 143
[81] 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  39
[17]  40  45  46  52  53  54  55  56  59  60  61  62  63  64  65  66
[33]  67  69  70  71  72  73  75  76  79  80  81  82  83  88 118 124
[49] 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 ffreq
4   181298 333484     2.6     81  116  257 7.655 0.190094 8.0     1
5   181307 333330     2.8     48  117  269 7.480 0.277090 8.7     1
6   181390 333260     3.0     61  137  281 7.791 0.364067 7.8     1
140 179917 331325     0.8     46   42  141 9.970 0.445580 4.5     3
    soil lime landuse dist.m    logZn    logCu
4      2    0      Ga    270 2.409933 1.908485
5      2    0      Ah    380 2.429752 1.681241
6      2    0      Ga    470 2.448706 1.785330
140    2    0      Am    540 2.149219 1.662758
```

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.

> **Note:** The numbers shown to the left of each row 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. The creator of the data set can use any character string as a row name, in the

same way that any character string can be used as a column name.

A `data.frame` object, e.g., the `meuse` data frame, has column names, which you can see with the `colnames` function, and may have row names, which you can see with the `row.names` function.

In this dataset the row names are observation ID numbers, as strings. 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 matrix.

Here you can see that the row names are character strings, i.e., ID's, not matrix row numbers:

```
print(row.names(meuse))
```

```
  [1] "1"   "2"   "3"   "4"   "5"   "6"   "7"   "8"   "9"   "10"
 [11] "11"  "12"  "13"  "14"  "15"  "16"  "17"  "18"  "19"  "20"
 [21] "21"  "22"  "23"  "24"  "25"  "26"  "27"  "28"  "29"  "30"
 [31] "31"  "32"  "33"  "34"  "35"  "37"  "38"  "39"  "40"  "41"
 [41] "42"  "43"  "44"  "45"  "46"  "47"  "48"  "49"  "50"  "51"
 [51] "52"  "53"  "54"  "55"  "56"  "57"  "58"  "59"  "60"  "61"
 [61] "62"  "63"  "64"  "65"  "66"  "67"  "69"  "75"  "76"  "79"
 [71] "80"  "81"  "82"  "83"  "84"  "85"  "86"  "87"  "88"  "89"
 [81] "90"  "123" "160" "163" "70"  "71"  "91"  "92"  "93"  "94"
 [91] "95"  "96"  "97"  "98"  "99"  "100" "101" "102" "103" "104"
[101] "105" "106" "108" "109" "110" "111" "112" "113" "114" "115"
[111] "116" "117" "118" "119" "120" "121" "122" "124" "125" "126"
[121] "127" "128" "129" "130" "131" "132" "133" "134" "135" "136"
[131] "161" "162" "137" "138" "140" "141" "142" "143" "144" "145"
[141] "146" "147" "148" "149" "150" "151" "152" "153" "154" "155"
[151] "156" "157" "158" "159" "164"
```

Challenge: Plot the elevation of each observation point against its distance from the river in meters. What relation do you expect? Do you see this relation? Is it consistent? Are there any unusual points that do not fit the overall pattern? If so, identify them.

7 Model-based feature-space modelling

A common non-spatial (feature-space) approach to prediction is to **model** one variables' distribution (the **dependent** or **response** variable) by one or more other variables (the **independent** or **predictor** variables). This is sometimes called "regression modelling" in the general sense. All variables in regression modelling can be either continuous or categorical.

There are two general modelling approaches [3]: **model-based** and **data-driven**. As Breiman [3] explains:

> "There are two cultures in the use of statistical modeling to reach conclusions from data. One assumes that the data are generated by a given stochastic data model. The other uses algorithmic models and treats the data mechanism as unknown."

The two approaches are thorougly investigated and compared in the text of Hastie *et al.* [13] and the simplified version of that text by James *et al.* [17].

In this section we deal with model-based approache, specifically **linear** models (§7.2 and §7.3). In §9, below, we deal with data-driven approaches.

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. The value predicted by the linear model is called the **fitted** value, because it results from the model fit to the entire set of calibration points. For the OLS fit to be valid, the residuals ε_i must 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 §13.

The first three assumptions can be examined after the regression parameters are estimated, using **regression diagnostics**, see §7.2.2, below. The fourth assumption can be examined with the **residual variogram**, see §13.2, below.

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 * Ordinary Least Squares (OLS) 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:[9]

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

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:

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

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

Note: $\mathbf{y}^T\mathbf{X}\beta$ is a 1×1 matrix, i.e., a scalar[10], so it is equivalent to its transpose: $\mathbf{y}^T\mathbf{X}\beta = [\mathbf{y}^T\mathbf{X}\beta]^T = \beta^T\mathbf{X}^T\mathbf{y}$. 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 β, setting this equal to $\mathbf{0}$, and solving:

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

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 17 : 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)
```

In a linear model from a continuous predictor the design matrix has a column of 1's (for the intercept) and the values of the predictor at each observation point. Here are the first few rows, computed with the `model.matrix` function:

```
model.matrix(m.lzn.lcu)[1:6,]
```

```
  (Intercept)    logCu
1           1 1.929419
2           1 1.908485
3           1 1.832509
4           1 1.908485
5           1 1.681241
6           1 1.785330
```

```
meuse$logCu[1:6]
```

```
[1] 1.929419 1.908485 1.832509 1.908485 1.681241 1.785330
```

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

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

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 18 : 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)

[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 19 : 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
-0.60973 -0.07904 -0.00027  0.08739  0.37686

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.58820    0.07913   7.434    7e-12 ***
logCu        1.27403    0.05071  25.122   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1389 on 153 degrees of freedom
Multiple R-squared:  0.8049,Adjusted R-squared:  0.8036
F-statistic: 631.1 on 1 and 153 DF,  p-value: < 2.2e-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.

Q11 : *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 A11*
•

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.

Q12 : *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 A12* •

7.2.2 Model diagnostics

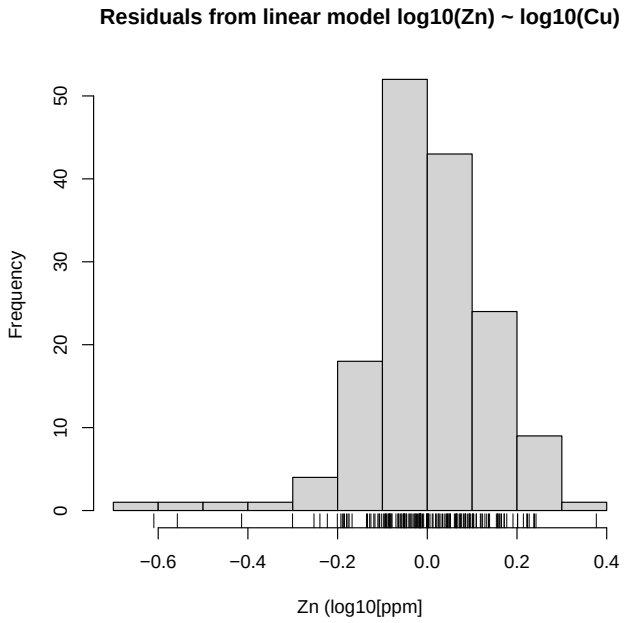
Linear modelling is a complicated topic, covered in many texts, e.g., [8, 11]. A fundamental requirement for the ordinary least squares (OLS) fit such as computed by the `lm` function is that the **residuals** must be **independent and identically normally-distributed**; if this assumption is not met various adjustments must be made or other methods used.

> **Note:** A thorough discussion of regression residuals, and various diagnostic techniques using them, is given by Cook & Weisberg [7] and is also covered in the regression texts listed in the previous paragraph.

TASK 20 : 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),
     main = "Residuals from linear model log10(Zn) ~ log10(Cu)",
     xlab = "Zn (log10[ppm]")
rug(residuals(m.lzn.lcu))
```



Q13 : *Do the residuals appear to be normally-distributed?* *Jump to A13* •

A linear model must satisfy several assumptions [7, 11], among which are:

1. no relation between predicted values and residuals;
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.

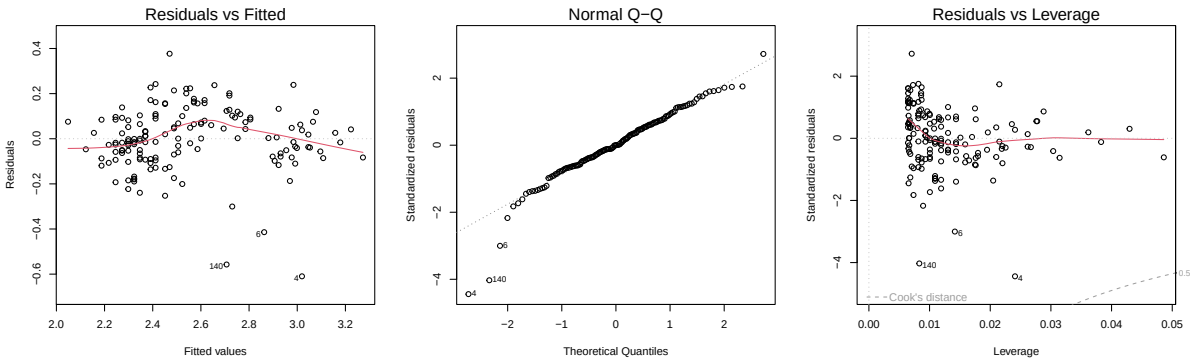
We can view these graphically, with the `plot` method, which specializes to the `plot.lm` function if it is called on objects of class `lm`. This function produces six different plots; the most useful are `1` "Residuals vs. fitted values", `2` "Normal Q-Q", and `5` "Residuals vs. leverage"; see `?plot.lm`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

TASK 21 : 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[11]. The `mfrow` "multiple frames, draw row-wise" argument to the `par` "graphics parameters" function specifies the layout of multiple graphs; here we want one row and three columns, i.e., a 1x3 array of graphs. After producing this graph, we re-set the graphics device to only produce one plot, i.e., a 1x1 array.

```
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 value - 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

[11] see `?plot.lm`

observation[12], against its standardized residual r_{S_i}[13]. 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".

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.[14] 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.

Q14 : *Is there any pattern with respect to the fitted values (see the first diagnostic plot)?* *Jump to A14* •

TASK 22 : 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))

  4   5   6 129 140
  4   5   6 123 135

# the records in the data frame that correspond to large absolute residuals
meuse[which.ix,]

         x      y cadmium copper lead zinc  elev      dist  om ffreq
4   181298 333484     2.6     81  116  257 7.655 0.1900940 8.0     1
5   181307 333330     2.8     48  117  269 7.480 0.2770900 8.7     1
6   181390 333260     3.0     61  137  281 7.791 0.3640670 7.8     1
129 179849 332142     1.2     30  244  703 8.540 0.0921353 8.3     2
```

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

[13] 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

[14] $D_i = \frac{r_i}{\text{trace}(\mathbf{H})} \frac{h_i}{1-h_i}$, 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.

```
140 179917 331325     0.8     46   42  141 9.970 0.4455800 4.5     3
    soil lime landuse dist.m    logZn    logCu
4      2    0      Ga    270 2.409933 1.908485
5      2    0      Ah    380 2.429752 1.681241
6      2    0      Ga    470 2.448706 1.785330
129    1    0      Fw     70 2.846955 1.477121
140    2    0      Am    540 2.149219 1.662758

# these large absolute residuals
residuals(m.lzn.lcu)[which.ix]

         4          5          6        129        140
-0.6097276 -0.3003938 -0.4140514  0.3768635 -0.5573786

# the largest one
which.max(abs(residuals(m.lzn.lcu))[which.ix])

4
1
```

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.

Q15 : *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 A15* •

Q16 : *Looking at the "Normal Q-Q" plot, do the residuals appear to be normally distributed?* *Jump to A16* •

Q17 : *Looking at the "Residuals vs. leverage" plot, do the high-leverage residuals have a high Cook's distance?* *Jump to A17* •

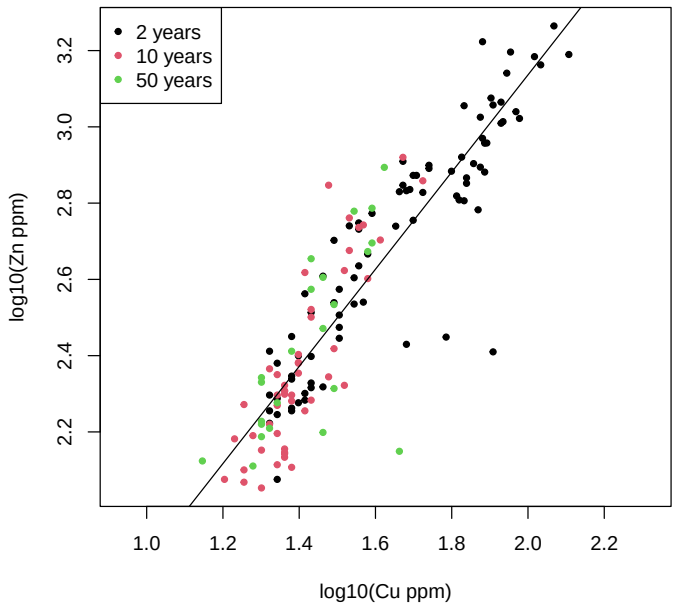
TASK 23 : Repeat the scatterplot of the relation between log-transformed Zn and Cu, adding the OLS regression line. •

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).

Note: You can see the list of printing characters at the help for the `points` base graphics function, and the list of basic colours with the `palette` function:

```
palette()

[1] "black"   "#DF536B" "#61D04F" "#2297E6" "#28E2E5" "#CD0BBC"
[7] "#F5C710" "gray62"
```

```
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)
```



Q18 : *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 A18* •

Challenge: Repeat this analysis but using distance in meters from the river as the continuous predictor of topsoil log_{10}Zn content. What is the hypothesis for this model? Does your analysis give support to this hypothesis? Are there any particularly poorly-modelled observations? Do they have a large influence on the model coefficient?

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 concentra-

tion; this would be evidence that the metals are brought from upstream industry.

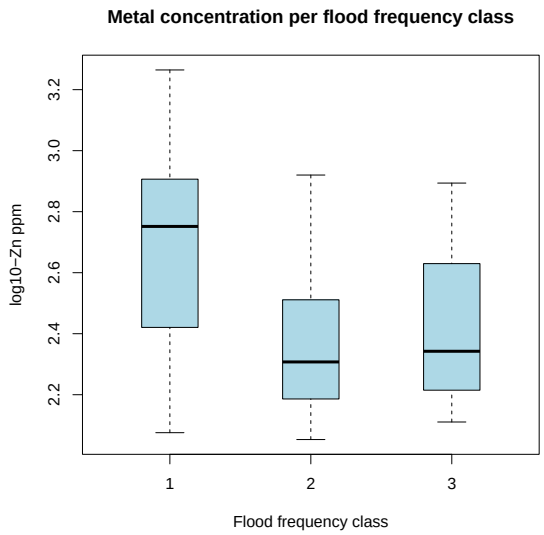
TASK 24 : Model the concentration of $\log_{10}$Zn from the flooding frequency. •

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=0.4, col="lightblue")
```



This example shows how **optional function arguments** (here, to the `boxplot` function) can be used to enhance a plot. We specify labels, a title, and the relative width of the boxes.

> **Note:** The `notch` argument can be set to `TRUE` to show the approximate confidence interval of the median (the heavy horizontal line in the boxplot).

Q19 : *Describe the relation between the different flood frequencies and the metal concentration.* *Jump to A19* •

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)
```

In a linear model from a categorial predictor the model matrix uses "dummy" variables. Here are the first few rows, all from flood frequency class 1. Note that the first class is considered as the intercept, and the other two as contrasts.

```
model.matrix(m.lzn.ff)[1:6,]

  (Intercept) ffreq2 ffreq3
1           1      0      0
2           1      0      0
3           1      0      0
4           1      0      0
5           1      0      0
6           1      0      0
```

TASK 25 : View the model summary. •

```
summary(m.lzn.ff)


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

Residuals:
     Min       1Q   Median       3Q      Max
-0.62419 -0.22330 -0.01762  0.20171  0.56484

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.69974    0.02976  90.727  < 2e-16 ***
ffreq2      -0.33187    0.04935  -6.725 3.34e-10 ***
ffreq3      -0.27501    0.06418  -4.285 3.23e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2727 on 152 degrees of freedom
Multiple R-squared:  0.2531,Adjusted R-squared:  0.2433
F-statistic: 25.75 on 2 and 152 DF,  p-value: 2.338e-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;

Q20 : *How much of the total variation in metal concentration is explained by the flooding frequency?* *Jump to A20*
•

Q21 : *What is the modelled mean log concentration of the metal in each class? (Hint: look at the "Estimate" for the model's coefficients).* *Jump to A21* •

We can see the predicted means and prediction variances by using the `predict` method applied to the fitted linear model, also specifying the optional `se.fit` argument. The `newdata` argument is just a list of the three flood frequency levels, since these are the only predictors, and apply at all locations in that class.

```
pred.ff <- predict(m.lzn.ff, newdata=data.frame(ffreq=as.factor(c(1:3))), se.fit=TRUE,
                   interval="prediction")
pred.ff$fit

       fit      lwr      upr
1 2.699739 2.157719 3.241760
2 2.367871 1.823465 2.912278
3 2.424733 1.874321 2.975144

pred.ff$se.fit

         1          2          3
0.02975682 0.03936457 0.05686725
```

Notice how the width of the prediction interval and the standard error of the fit increase as the number of observations per class decreases.

Clearly, this prediction is not so good: (1) only three values, no variation within a flood frequency class; (2) high variability of predictions because of poor model fit. So, we turn to **spatial** analysis (below, §10), and later to **mixed predictors** (§13).

Challenge: Repeat this analysis but with soil type as the categorical predictor of $\log_{10}$Zn.

Challenge: The soil type may well be related to the amount of flooding and the type of sediment brought by the river. Use the `table` function to examine the cross-classification table of these two categorical predictors. Are they independent? Check this with the `chisq.test` χ^2 test function.

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 26 : 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
-0.60925 -0.07893  0.00043  0.08951  0.38770

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.59605    0.10260   5.809 3.59e-08 ***
ffreq2      -0.01224    0.02958  -0.414    0.679
ffreq3       0.01806    0.03575   0.505    0.614
logCu        1.26966    0.06124  20.733  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1395 on 151 degrees of freedom
Multiple R-squared:  0.8058,Adjusted R-squared:  0.802
F-statistic: 208.9 on 3 and 151 DF,  p-value: < 2.2e-16
```

Q22 : *How much of the variance is explained? How does this compare with the two single-factor models?* *Jump to A22* •

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 2.9390
2    153 2.9534 -2 -0.014361 0.3689 0.6921
```

Q23 : *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 A23* •

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 concentration.

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 27 : 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.

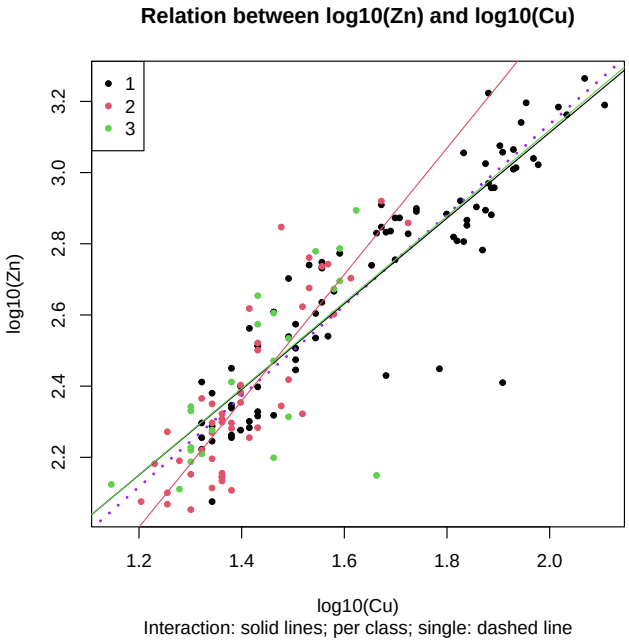
```
m.lzn.ff.lcu.i <- lm(logZn ~ ffreq * logCu, data=meuse)
summary(m.lzn.ff.lcu.i)


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

Residuals:
     Min       1Q   Median       3Q      Max
-0.59244 -0.07008  0.00991  0.08418  0.35131

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.706791   0.111244   6.354 2.42e-09 ***
ffreq2       -0.833507   0.271659  -3.068  0.00256 **
ffreq3       -0.008742   0.328816  -0.027  0.97883
logCu         1.202828   0.066538  18.077  < 2e-16 ***
ffreq2:logCu  0.572494   0.187992   3.045  0.00275 **
ffreq3:logCu  0.007974   0.226076   0.035  0.97191
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1362 on 149 degrees of freedom
Multiple R-squared:  0.8173,Adjusted R-squared:  0.8112
F-statistic: 133.3 on 5 and 149 DF,  p-value: < 2.2e-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 2.7654
2    153 2.9534 -4  -0.18798 2.532 0.04278 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Q24 : *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 A24* •

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)
```



Q25 : *In which flooding frequency class is the relation between $log_{10}Zn$ and $log_{10}Cu$ different from the overall relation?* *Jump to A25* •

We return to this theme in §13.4, when we use a multivariate model as part of kriging with external drift.

8 Spatially-explicit R objects

In much of the remainder of this exercise we will work with spatially-explicit objects. To work with such objects, we will use the `sf` "Simple Features" R package. This is an example of a package that provides special-purpose R **classes** for objects, and **methods** that use the provided class structures.

We installed this package in §1.1, above, so it is available to be loaded

into the workspace. Recall, in that section we showed how to load packages into the workspace with the `library` function. If you have already loaded them, you can skip this step, but it does no harm to do it again.

```
library(sf)
library(gstat)
```

R classes

All objects in R have a class, reported by the `class` function. For scalars and simple vectors this is just the mode of the data types, reported by the `mode` function, e.g., `numeric`, `logical` or `list`. More complicated structures have their own classes, which control how functions and methods work on them. Examples are `matrix`, `data.frame`, and `factor`.

Here are some examples:

```
class(1); class("A"); class(TRUE)

[1] "numeric"
[1] "character"
[1] "logical"

class(list(1, "A", TRUE))

[1] "list"

class(as.matrix(1:9, nrow=3))

[1] "matrix" "array"

class(meuse)

[1] "data.frame"

class(meuse$ffreq)

[1] "factor"
```

These are all defined in base R; now we will use classes defined in the additional package `sf`, which has classes for spatial "simple features".

Task 28 : Convert the `meuse` data frame to 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 `sf` package provides classes for spatially-explicit data; we just need to tell it which fields represent the coordinates. The `class` function shows the class name of an object.

Almost all `sf` functions begin with `st_`, standing for "space-time". The command to convert from a `data.frame` to an `sf` object is `st_as_sf`, meaning "convert this object to a spatial object". For this to work, we must identify which of the fields in the `data.frame` represent the coordinates. This we do with a list of them (here, 2D) and the `coords` argument:

We choose to keep the original data frame, and make a new version that

is explicitly spatial.

```
class(meuse)

[1] "data.frame"

meuse.sf <- st_as_sf(meuse, coords = c("x","y"))
class(meuse.sf)

[1] "sf"         "data.frame"
```

Note the use of the `c` "catenate" (meaning "make a chain") function to create a vector of the two field names that represent the coordinate values, here `"x"` and `"y"`.

The class of the spatial object is now `sf`, as an extension of class `data.frame`.

TASK 29 : Display the structure of the spatially-explicit object. •

```
str(meuse.sf)

Classes 'sf' and 'data.frame': 155 obs. of  15 variables:
 $ 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 ...
 $ geometry:sfc_POINT of length 155; first list element:  'XY' num  181072 333611
 - attr(*, "sf_column")= chr "geometry"
 - attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA NA NA NA NA N
  ..- attr(*, "names")= chr [1:14] "cadmium" "copper" "lead" "zinc" ...
```

Q26 : *Which field refers to the spatial object's geometry? What is its data type?* *Jump to A26* •

We would like to refer this dataset to the Earth's surface. At present its coordinate reference system (CRS) is unknown; we can see this with the `st_crs` function:

```
st_crs(meuse.sf)

Coordinate Reference System: NA
```

We know from the metadata that these coördinates are in the Dutch national triangulation ("Rijksdriehoek", or RDH) system. This has EPSG code 28992[15].

TASK 30 : Specify the correct CRS for this dataset. •

[15] See the EPSG database http://www.epsg-registry.org/

Again use the `st_crs` function, but this time assign to it:

```
st_crs(meuse.sf) <- 28992
print(st_crs(meuse.sf))

Coordinate Reference System:
  User input: EPSG:28992
  wkt:
PROJCRS["Amersfoort / RD New",
    BASEGEOGCRS["Amersfoort",
        DATUM["Amersfoort",
            ELLIPSOID["Bessel 1841",6377397.155,299.1528128,
                LENGTHUNIT["metre",1]]],
        PRIMEM["Greenwich",0,
            ANGLEUNIT["degree",0.0174532925199433]],
        ID["EPSG",4289]],
    CONVERSION["RD New",
        METHOD["Oblique Stereographic",
            ID["EPSG",9809]],
        PARAMETER["Latitude of natural origin",52.1561605555556,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8801]],
        PARAMETER["Longitude of natural origin",5.38763888888889,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8802]],
        PARAMETER["Scale factor at natural origin",0.9999079,
            SCALEUNIT["unity",1],
            ID["EPSG",8805]],
        PARAMETER["False easting",155000,
            LENGTHUNIT["metre",1],
            ID["EPSG",8806]],
        PARAMETER["False northing",463000,
            LENGTHUNIT["metre",1],
            ID["EPSG",8807]]],
    CS[Cartesian,2],
        AXIS["easting (X)",east,
            ORDER[1],
            LENGTHUNIT["metre",1]],
        AXIS["northing (Y)",north,
            ORDER[2],
            LENGTHUNIT["metre",1]],
    USAGE[
        SCOPE["Engineering survey, topographic mapping."],
        AREA["Netherlands - onshore, including Waddenzee, Dutch Wadden Islands and 12-mile of
        BBOX[50.75,3.2,53.7,7.22]],
    ID["EPSG",28992]]
```

TASK 31 : Load the Meuse river outline object and convert it to a spatial `sf` object, with the correct CRS. •

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. This must be converted to an `sf` object, and then can be plotted along with the sample points. The `st_linestring` function converts a matrix of coordinates into a curve (line) in a Simple Features format, and then the `st_sfc`'create a Simple Features class" function converts to an `sf` object. At the same time, the CRS must be specified; this is copied from the CRS already set for the points object.

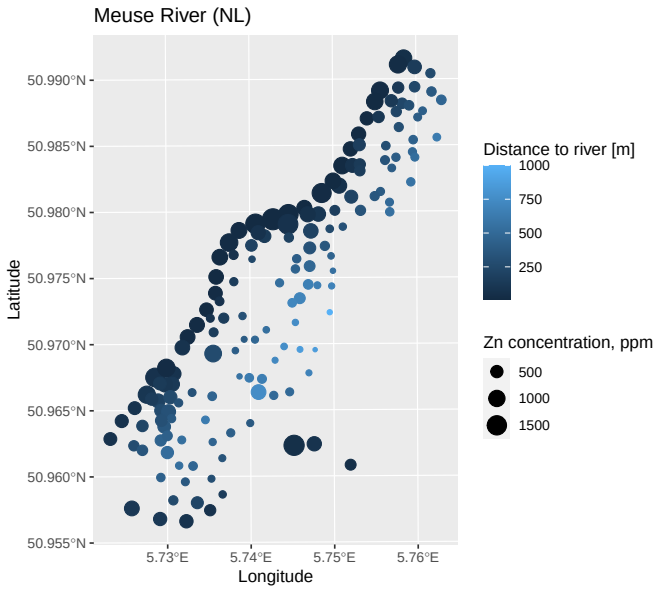
```
data(meuse.riv, package="sp")
class(meuse.riv)

[1] "matrix" "array"
```

```
meuse.riv.sf <- st_sfc(st_linestring(meuse.riv, dim="XY"), crs = st_crs(meuse.sf))
class(meuse.riv.sf)

[1] "sfc_LINESTRING" "sfc"

summary(meuse.riv.sf)

   LINESTRING    epsg:28992 +proj=ster...
            1             0             0
```

TASK 32 : Display a map of the sample locations, along with the line of the Meuse river. Show the distance to the river with one aesthetic, and the Zn concentration with another. •

This is our first use of `ggplot2` graphics; previous plots have used R base `graphics` package. The "Grammar of Graphics" builds plots from separate elements. Here we first specify the dataframe to be plotted, then the type of plot, here a map of an `sf` object, then the two aesthetic elements that show the value of variables in the dataframe, and finally some labels.

```
require(ggplot2)

Loading required package:  ggplot2

ggplot(data = meuse.sf) +
    geom_sf(mapping = aes(size=zinc, color=dist.m)) +
    labs(x = "Longitude", y = "Latitude", title = "Meuse River (NL)",
         color="Distance to river [m]", size = "Zn concentration, ppm")
```



Q27 : *How are the points distributed over the study area? For example, are they evenly-spaced (gridded)? Random? Denser in some areas?* *Jump to A27* •

We take this opportunity to also make the prediction grid covering the study area (`meuse.grid`) a Simple Features object. This will be used in §9, §11.2, and §13, below.

Task 33 : Load the 40 m x 40 m interpolation grid covering the study area and convert it to an `sf` object. •

As before, the `data` function loads a built-in dataset and the `st_as_sf`function converts a `data.frame` to an `sf` object, with a `geometry` field in the data frame. We also specify the CRS.

```
data(meuse.grid, package="sp")
class(meuse.grid)

[1] "data.frame"

names(meuse.grid)

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

meuse.grid.sf <- st_as_sf(meuse.grid, coords = c("x", "y"))
st_crs(meuse.grid.sf) <- st_crs(meuse.sf)
```

The grid is structured as a set of points.

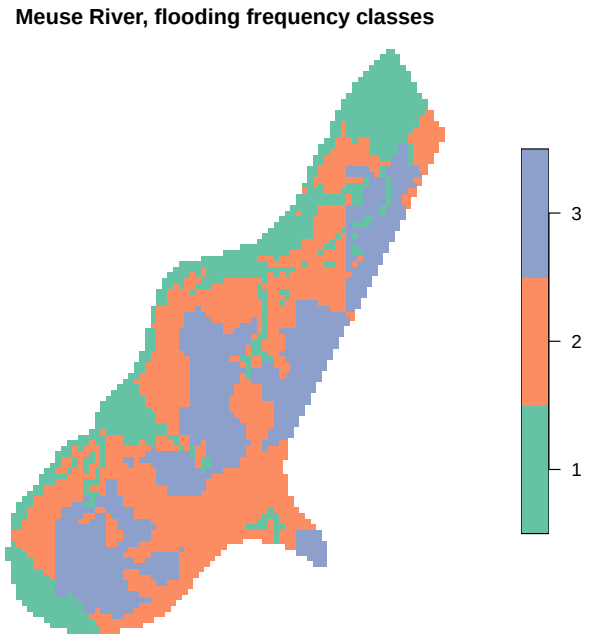
Task 34 : The grid has several attributes. List them and investigate their meaning (see the help text, `?meuse.grid`). •

```
summary(meuse.grid.sf)

     part.a           part.b            dist        soil     ffreq
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   1:1665   1: 779
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.1193   2:1084   2:1335
 Median :0.0000   Median :1.0000   Median :0.2715   3: 354   3: 989
 Mean   :0.3986   Mean   :0.6014   Mean   :0.2971
 3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.4402
 Max.   :1.0000   Max.   :1.0000   Max.   :0.9926
          geometry
 POINT        :3103
 epsg:28992   :   0
 +proj=ster...:   0
```

Task 35 : Display a map of the flood frequency classes. •

The default `plot` function displays the points; use the square character `pch = 15` to give the map a pixelated look.

```
plot(meuse.grid.sf["ffreq"], pch = 15,
     main = "Meuse River, flooding frequency classes")
```



Q28 : *What is the meaning of flood frequency class* 1*? What is its spatial distribution?* *Jump to A28* •

9 Data-driven feature-space modelling

Data-driven approaches to reach conclusions from data [3] do not assume any model structure and attempt to parameterize it from the data, as in model-based approaches as explained in §7, above. Instead, these approaches search for structures in the data. In this section we explain two approaches: **classification and regression trees** (regression: §9.1, classification: §9.2) and **random forests** (§9.3).

Hastie *et al.* [13, §9.2] give a thorough explanation of a tree-based regression method known as CART ("Classification and Regression Trees") [4], 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.* [17, §8.1]. A wide variety of R packages are available for data-driven approaches; these are listed and commented at the *CRAN Task View: Machine Learning & Statistical Learning*[16]

9.1 Regression trees

In this 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

[16] https://cran.r-project.org/web/views/MachineLearning.html

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.

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.[17] 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.

Task 36 : Build a regression tree to model $\log_{10}$Zn from the flooding frequency, normalized distance to river, elevation above m.a.s.l, and the soil type according to the 1:50 000 soil map of the Netherlands. These predictors are available in both the point data set (for model building) and the prediction grid (for mapping over the study area). •

Q29 : *What theory of the origin of the Zn is this model testing?* *Jump to A29* •

[17] The "classification" trees explained in §9.2 follow a similar logic, but are used to predict a categorical (classified) outcome.

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. This is an arbitrary choice, which we will re-examine by cross-validation.

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. Thus closely-correlated predictors may substitute for each other.

Here we use as possible predictors: (1) the flooding frequency `ffreq`, (2) the normalized distance to the river `dist`, (3) the elevation above m.a.s.l. `elev`, (4) the soil type according to the 1:50 000 soil map of the Netherlands `soil`.

> **Note:** We use the normalized distance, rather than the distance in meters, because the prediction grid includes normalized distances to each grid cell centre, not distance in meters.

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

TASK 37 : 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 1.513633e+01 2.556160
     2) dist>=0.160161 89 3.680112e+00 2.360022
       4) elev>=6.943 82 1.880255e+00 2.319222
         8) elev>=9.028 31 4.108683e-01 2.227815
          16) dist>=0.5896225 7 7.830826e-03 2.106901 *
          17) dist< 0.5896225 24 2.708457e-01 2.263082
            34) elev>=9.5415 11 9.304235e-02 2.191158 *
```

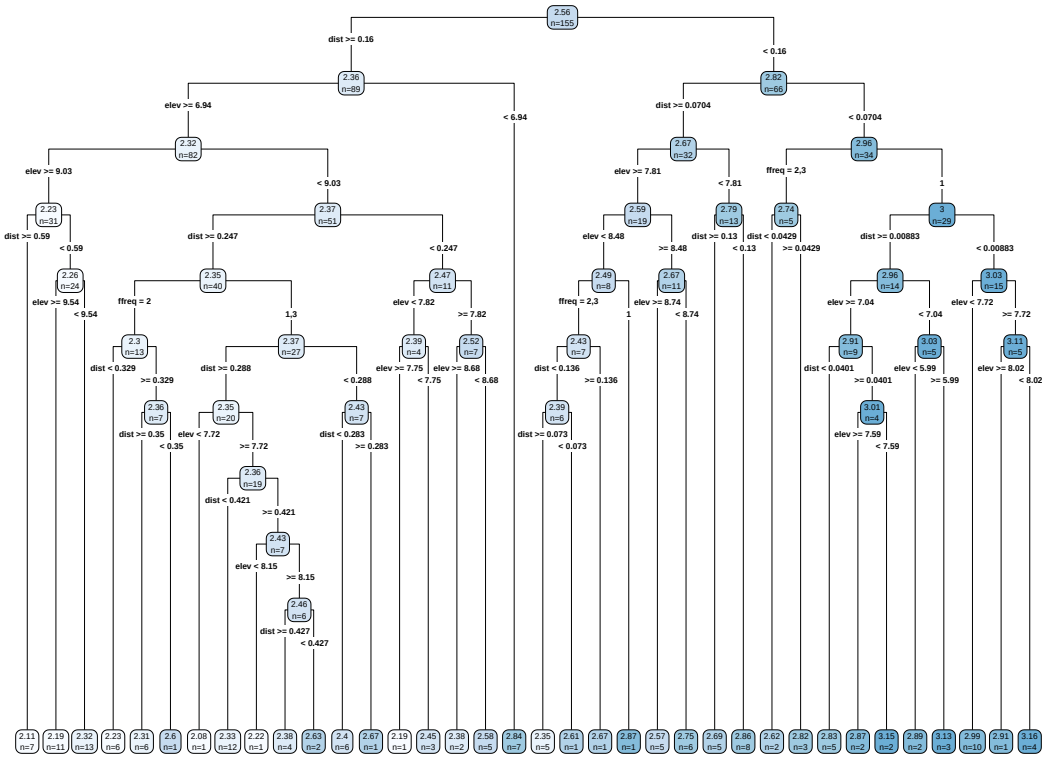
```
          35) elev< 9.5415 13 7.275213e-02 2.323940 *
       9) elev< 9.028 51 1.052935e+00 2.374783
        18) dist>=0.2471685 40 7.144941e-01 2.348192
          36) ffreq=2 13 1.489729e-01 2.298678
            72) dist< 0.3287205 6 2.046678e-02 2.232898 *
            73) dist>=0.3287205 7 8.029203e-02 2.355060
             146) dist>=0.350464 6 9.114778e-03 2.313893 *
             147) dist< 0.350464 1 0.000000e+00 2.602060 *
          37) ffreq=1,3 27 5.183037e-01 2.372032
            74) dist>=0.287949 20 3.677913e-01 2.350104
             148) elev< 7.7155 1 0.000000e+00 2.075547 *
             149) elev>=7.7155 19 2.884420e-01 2.364555
               298) dist< 0.4211215 12 4.965605e-02 2.326868 *
               299) dist>=0.4211215 7 1.925236e-01 2.429162
                 598) elev< 8.154 1 0.000000e+00 2.220108 *
                 599) elev>=8.154 6 1.415363e-01 2.464004
                  1198) dist>=0.426563 4 5.110716e-02 2.381562 *
                  1199) dist< 0.426563 2 8.869630e-03 2.628887 *
            75) dist< 0.287949 7 1.134198e-01 2.434683
             150) dist< 0.2825155 6 5.071462e-02 2.396044 *
             151) dist>=0.2825155 1 0.000000e+00 2.666518 *
        19) dist< 0.2471685 11 2.073078e-01 2.471478
          38) elev< 7.82 4 5.803321e-02 2.387391
            76) elev>=7.75 1 0.000000e+00 2.187521 *
            77) elev< 7.75 3 4.769157e-03 2.454014 *
          39) elev>=7.82 7 1.048300e-01 2.519529
            78) elev>=8.681 2 3.583297e-02 2.379365 *
            79) elev< 8.681 5 1.398883e-02 2.575594 *
     5) elev< 6.943 7 6.435650e-02 2.837964 *
   3) dist< 0.160161 66 3.415388e+00 2.820649
     6) dist>=0.0703509 32 1.198374e+00 2.673985
      12) elev>=7.81 19 6.626694e-01 2.593058
        24) elev< 8.48 8 3.329208e-01 2.488569
          48) ffreq=2,3 7 1.642503e-01 2.433687
            96) dist< 0.135693 6 9.742288e-02 2.393798
             192) dist>=0.07302095 5 4.374072e-02 2.351497 *
             193) dist< 0.07302095 1 0.000000e+00 2.605305 *
            97) dist>=0.135693 1 0.000000e+00 2.673021 *
          49) ffreq=1 1 0.000000e+00 2.872739 *
        25) elev>=8.48 11 1.788816e-01 2.669050
          50) elev>=8.7415 5 5.003672e-02 2.572624 *
          51) elev< 8.7415 6 4.361309e-02 2.749405 *
      13) elev< 7.81 13 2.294029e-01 2.792263
        26) dist>=0.130247 5 7.751971e-02 2.688173 *
        27) dist< 0.130247 8 6.385069e-02 2.857320 *
     7) dist< 0.0703509 34 8.808450e-01 2.958686
      14) ffreq=2,3 5 6.668477e-02 2.743572
        28) dist< 0.04291265 2 5.176234e-03 2.624905 *
        29) dist>=0.04291265 3 1.456883e-02 2.822684 *
      15) ffreq=1 29 5.428997e-01 2.995774
        30) dist>=0.0088282 14 2.988164e-01 2.956055
          60) elev>=7.0415 9 1.706471e-01 2.912502
           120) dist< 0.0400747 5 8.711746e-03 2.833014 *
           121) dist>=0.0400747 4 9.085509e-02 3.011861
             242) elev>=7.592 2 4.290153e-03 2.874330 *
             243) elev< 7.592 2 1.090607e-02 3.149392 *
          61) elev< 7.0415 5 8.036658e-02 3.034452
           122) elev< 5.99 2 5.676054e-05 2.888989 *
           123) elev>=5.99 3 9.777667e-03 3.131428 *
        31) dist< 0.0088282 15 2.013835e-01 3.032845
          62) elev< 7.721 10 6.895802e-02 2.993519 *
          63) elev>=7.721 5 8.602990e-02 3.111497
           126) elev>=8.02 1 0.000000e+00 2.909556 *
           127) elev< 8.02 4 3.505491e-02 3.161982 *
```

TASK 38 : 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)
```



Q30 : *How many terminal nodes, i.e., prediction groups, does this tree have? How many internal nodes, i.e., splits?* *Jump to A30* •

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] 36

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

[1] 35
```

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 residual sum of squares (RSS). Variables not used in the tree can be included here, if they could have been used a **surrogate** (replacement) for another variable, should that variable have a missing value. This information is stored in the `variable.importance` field of the fitted model.

Task 39 : Display the relative variable importance, as a percentage of the total reduction in RSS from the full tree. •

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

      variableImportance
dist           46.997606
elev           30.031328
soil           14.286767
ffreq           8.684299
```

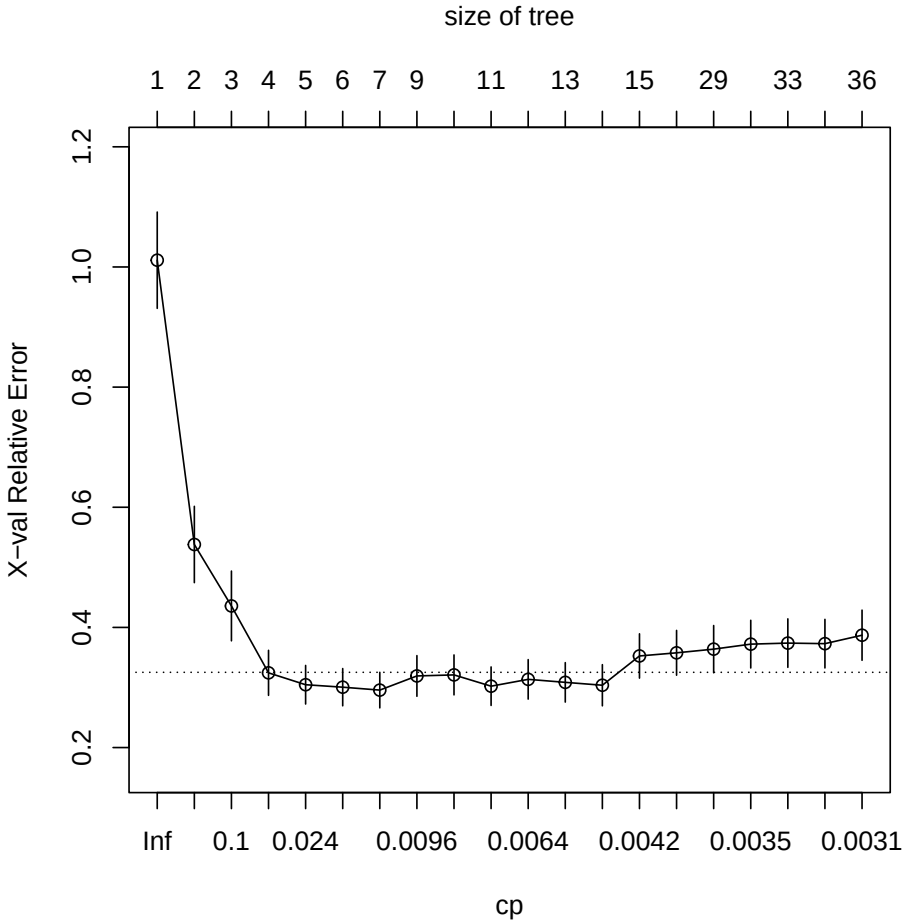
Q31 : *Which variables are most important?* *Jump to A31* •

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

Task 40 : Print and plot the cross-validation error rate vs. the complexity parameter and tree size. •

```
printcp(m.lzn.rp)


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

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

Root node error: 15.136/155 = 0.097654

n= 155

          CP nsplit rel error  xerror     xstd
1  0.5312272      0  1.000000 1.01125 0.080168
2  0.1146579      1  0.468773 0.53801 0.063548
3  0.0882756      2  0.354115 0.43571 0.058070
4  0.0275134      3  0.265839 0.32437 0.037491
5  0.0202362      4  0.238326 0.30458 0.032111
6  0.0179212      5  0.218090 0.30049 0.031017
7  0.0105553      6  0.200168 0.29564 0.029610
8  0.0087334      8  0.179058 0.31919 0.033851
9  0.0086635      9  0.170324 0.32092 0.033252
10 0.0069403     10  0.161661 0.30217 0.032055
11 0.0058160     11  0.154721 0.31352 0.032876
12 0.0056309     12  0.148905 0.30853 0.032855
13 0.0044150     13  0.143274 0.30372 0.034284
14 0.0040667     14  0.138859 0.35246 0.036891
15 0.0038071     19  0.118525 0.35769 0.037281
16 0.0035466     28  0.083869 0.36381 0.039318
17 0.0033631     29  0.080323 0.37215 0.039681
18 0.0032165     32  0.070233 0.37388 0.040371
19 0.0031011     34  0.063800 0.37295 0.040438
20 0.0030000     35  0.060699 0.38701 0.041786

plotcp(m.lzn.rp)
```



Note: Your output will look different. This is because the cross-validation is computed from a *random* starting point each time `rpart` is run. The resulting tree is the same (if the control parameters are the same) but the split of the dataset used to computed cross-validation is random. It is instructive to run this several times to see how variable are the cross-validation results. With small sample sets this variability can be high, showing the instability of regression trees, which will be mostly solved with Random Forests.

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.[18]

Q32 : *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 A32* •

A regression tree can be pruned back to any level of complexity. The aim

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

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.

TASK 41 : 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.[19]

```
head(cp.table <- m.lzn.rp[["cptable"]], 12)

            CP nsplit rel error    xerror       xstd
1  0.531227213      0 1.0000000 1.0112531 0.08016786
2  0.114657940      1 0.4687728 0.5380137 0.06354779
3  0.088275626      2 0.3541148 0.4357147 0.05807048
4  0.027513386      3 0.2658392 0.3243727 0.03749106
5  0.020236188      4 0.2383258 0.3045810 0.03211055
6  0.017921153      5 0.2180896 0.3004881 0.03101734
7  0.010555316      6 0.2001685 0.2956386 0.02961040
8  0.008733409      8 0.1790579 0.3191943 0.03385121
9  0.008663472      9 0.1703245 0.3209172 0.03325203
10 0.006940333     10 0.1616610 0.3021660 0.03205539
11 0.005815976     11 0.1547206 0.3135240 0.03287550
12 0.005630942     12 0.1489047 0.3085335 0.03285492

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

7
7

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

[1] 0.2956386

print(cp.table[cp.ix,])

        CP     nsplit  rel error     xerror       xstd
0.01055532 6.00000000 0.20016849 0.29563863 0.02961040

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

Q33 : *What is the minimum cross-validation error? At how many splits? What is the corresponding complexity parameter?* *Jump to A33* •

The minimum cross-validation error 0.2956 is found at complexity parameter 0.0106; 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] 0.325249

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

        CP     nsplit  rel error     xerror       xstd
```

[19] ?rpart

```
0.02751339 3.00000000 0.26583922 0.32437267 0.03749106

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.

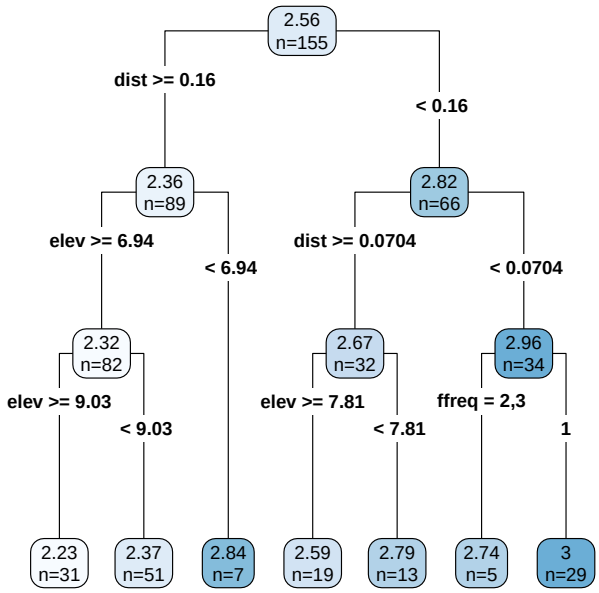
TASK 42 : Prune the tree back to complexity level identified by the minimum (not minimum plus sandard deviation – although you can try this for comparison). •

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

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

n= 155

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

 1) root 155 15.13633000 2.556160
   2) dist>=0.160161 89  3.68011200 2.360022
     4) elev>=6.943 82  1.88025500 2.319222
       8) elev>=9.028 31  0.41086830 2.227815 *
       9) elev< 9.028 51  1.05293500 2.374783 *
     5) elev< 6.943 7  0.06435650 2.837964 *
   3) dist< 0.160161 66  3.41538800 2.820649
     6) dist>=0.0703509 32  1.19837400 2.673985
      12) elev>=7.81 19  0.66266940 2.593058 *
      13) elev< 7.81 13  0.22940290 2.792263 *
     7) dist< 0.0703509 34  0.88084500 2.958686
      14) ffreq=2,3 5  0.06668477 2.743572 *
      15) ffreq=1 29  0.54289970 2.995774 *
```

TASK 43 : Plot the pruned regression tree. •

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



Q34 : *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 A34* •

Q35 : *Which predictor(s) was(were) used to build the pruned tree? What does this imply about the other(s)?* *Jump to A35* •

Q36 : *Interpret the structure of the tree in terms of geography.* *Jump to A36* •

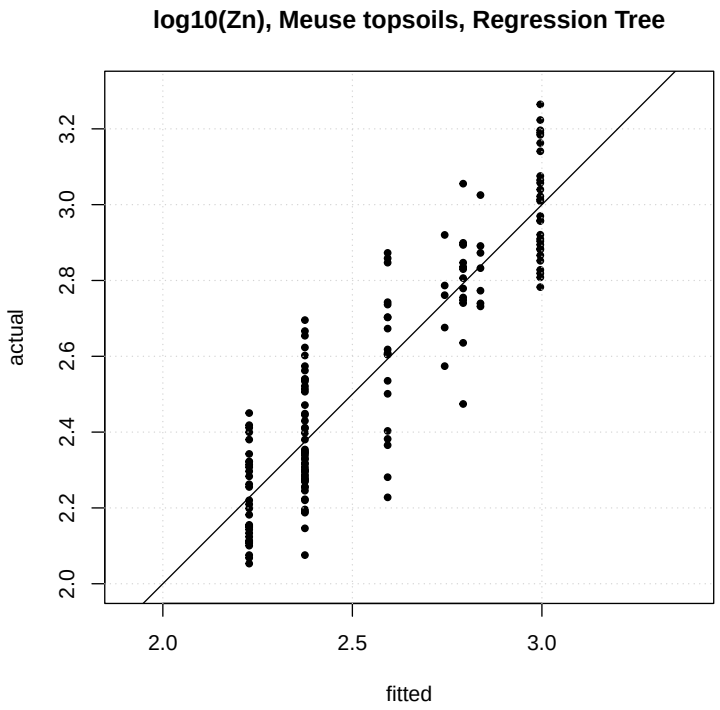
A fitted model can be used for prediction. There is a major difference with the linear model: the tree model only predicts the exact values at its leaves, whereas the linear model applies its formula to any predictor value and thus predicts any number of values. Here we do not have other points to predict,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

TASK 44 : 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.sf)
length(unique(p.rpp))

[1] 7

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

    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.36517 -0.09836 -0.01830  0.00000  0.09546  0.32070

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

[1] 0.1398113

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)
```



9.1.1 Predicting over the study area with the regression tree model

Now that we have a predictive model, we can use it to map the study area.

We have the `meuse.grid` prediction grid, convert to an `sf` object. This grid includes all the attributes we used in the tree, except elevation:

```
names(meuse.grid.sf)

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

Although elevation has been included in a different version of the Meuse

dataset[20] it's not in the dataset included in the `sp` package. So, we have to create it by interpolation from the known points.

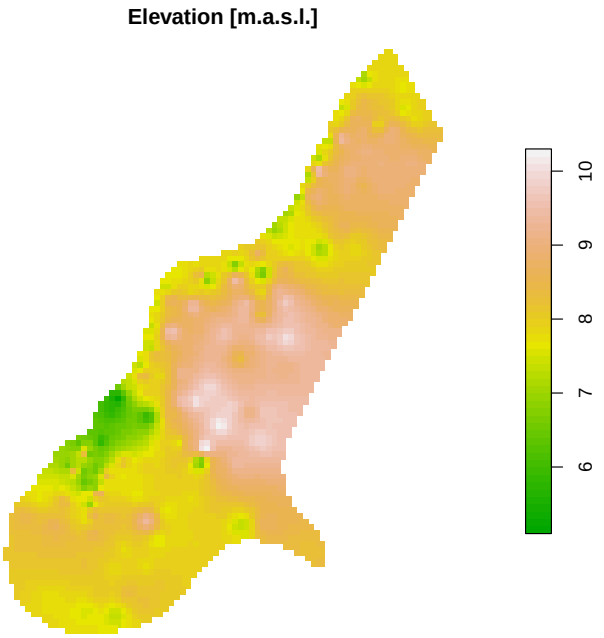
We thus need to "fill in the holes" between the observed points. We could do this by kriging, but that has not yet been introduced in this tutorial (see § 11). So we use a simple interpolation method that does not require a model of spatial dependence: inverse-distance weighted interpolation. This predicts at a location by a weighted linear combination of the values (here, elevation) at nearby known points, with the weights inversely proportional to some power of the distance to the point to be predicted.

TASK 45 : Predict the elevation over the prediction grid by inverse distance squared interpolation. •

The `idw` "inverse distance weighting" function is a special case of the `krige` function. Instead of a variogram model it uses the inverse distance power to be specified by the `idp` argument. We also must 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 <- gstat::idw(elev ~ 1, locations=meuse.sf,
           nmax=16, idp=2, newdata=meuse.grid.sf)

[inverse distance weighted interpolation]

meuse.grid.sf$elev <- tmp$var1.pred; rm(tmp)
summary(meuse.sf$elev)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  5.180   7.546   8.180   8.165   8.955  10.520

summary(meuse.grid.sf$elev)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  5.198   7.873   8.211   8.318   8.863  10.279

plot(meuse.grid.sf["elev"], pch=15,
     nbreaks = 64,
     pal = terrain.colors,
     main = "Elevation [m.a.s.l.]")
```

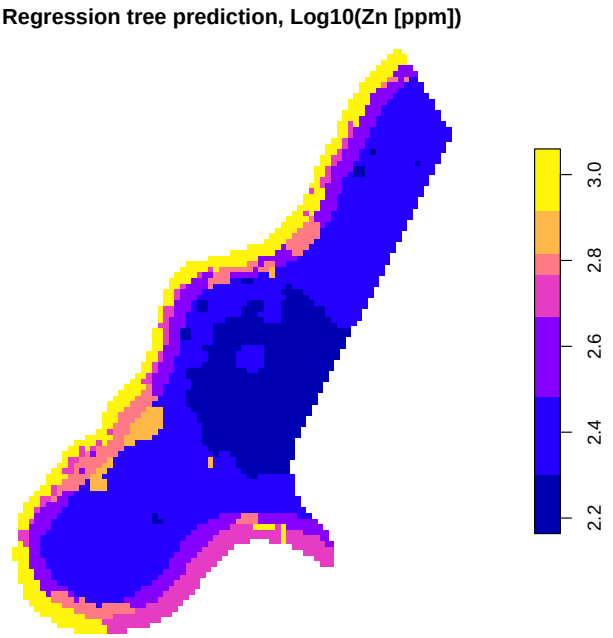
[20] http://spatial-analyst.net/book/meusegrids



Q37 : *Why are the extreme values of the point elevations not found in the grid elevations?* *Jump to A37* •

TASK 46 : Use the regression tree to map over the prediction grid. •

```
rt.grid <- predict(m.lzn.rpp, newdata = meuse.grid.sf)
meuse.grid.sf$rt.pred <- rt.grid; rm(rt.grid)
plot(meuse.grid.sf["rt.pred"], pch=15,
     nbreaks = 64,
     main = "Regression tree prediction, Log10(Zn [ppm])")
```



Q38 : *Comment on the success or otherwise of this modelling approach.*
Jump to A38 •

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.

Challenge: Build a regression tree with the four factors used in the above example, but also the coördinates. This will allow "boxes" of local relations of the other factors, but may simply isolate unusual points from their neighbours and not be useful in prediction – if so, this will be revealed in the cross-validation. You will need to give the coördinates explicitly in the model, so that the `data.frame` version of the dataset must be used, not the `sf` version.

9.2 * 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.* [13, §9.2] or James *et al.* [17, §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 equivalently increase in R^2) it uses the increase in classification accuracy. Outputs and interpretation are somewhat different from regression trees.

TASK 47 : Build a classification tree to model flooding frequency class based on the normalized distance to river, elevation above sea level, and soil type. To build (a possibly overfit) tree, use a complexity parameter of 1% minimum improvement. •

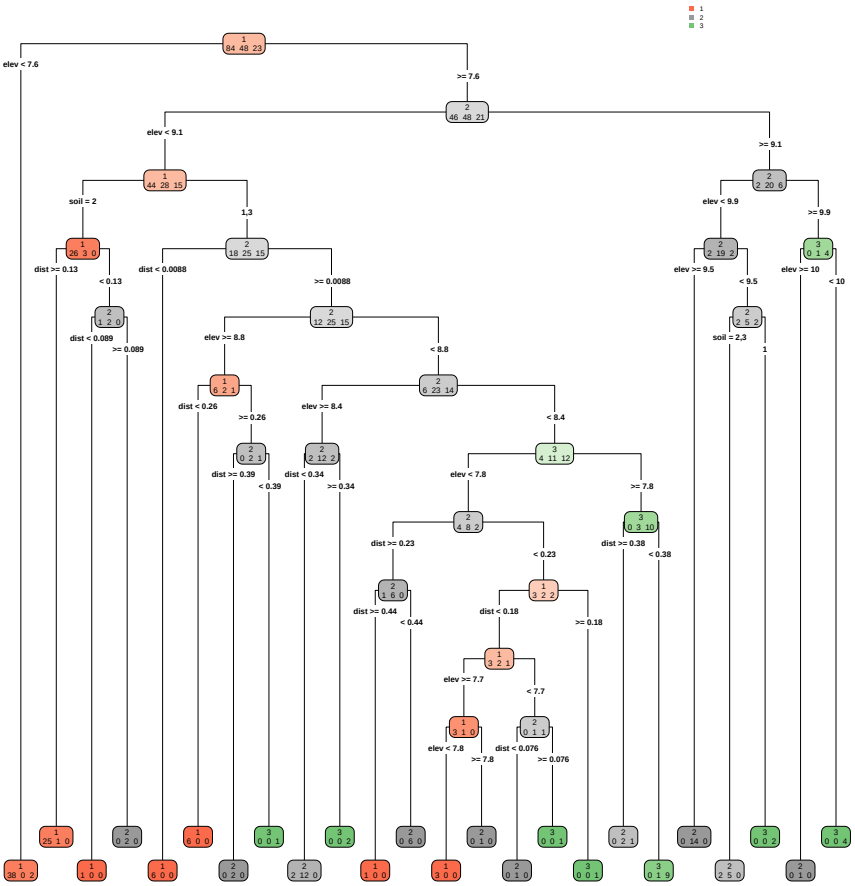
The model formula has the same form as for a continuous predictand. Since the response variable is categorical `rpart` automatically uses the `"class"` method instead of `"anova"` when determining the splits.

```
m.ff.rp <- rpart(ffreq ~ dist + elev + soil,
                 data=meuse.sf,
                 minsplit=2,
#                 method="class",
                 cp=0.01)
```

TASK 48 : Display the *unpruned* classification tree as a graph showing the number of observations at each node and leaf. •

This graph shows the majority class at each node in a different colour. The intensity of the colour (its chroma) is greater as the node purity increases.

```
rpart.plot(m.ff.rp, type=4, extra=1, tweak = 0.8)
```

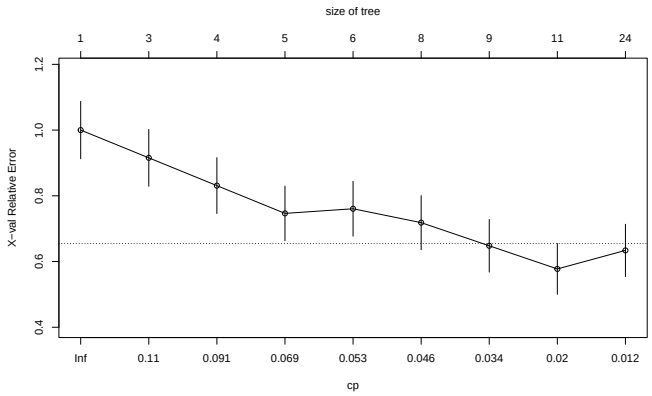


TASK 49 : Show the cross-validation error rate vs. complexity parameter, both in text and as a graph. Find the complexity parameter at the minimum cross-validation error rate. •

> **Note:** As in the regression trees, your output will look different, because the cross-validation is computed from a random starting point each time `rpart` is run.

```
printcp(m.ff.rp)


Classification tree:
rpart(formula = ffreq ~ dist + elev + soil, data = meuse.sf,
    minsplit = 2, cp = 0.01)

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

Root node error: 71/155 = 0.45806

n= 155

        CP nsplit rel error  xerror     xstd
1 0.126761      0   1.00000 1.00000 0.087366
2 0.098592      2   0.74648 0.91549 0.086527
3 0.084507      3   0.64789 0.83099 0.085141
4 0.056338      4   0.56338 0.74648 0.083179
```

```
5 0.049296      5   0.50704 0.76056 0.083547
6 0.042254      7   0.40845 0.71831 0.082391
7 0.028169      8   0.36620 0.64789 0.080107
8 0.014085     10   0.30986 0.57746 0.077343
9 0.010000     23   0.12676 0.63380 0.079594

plotcp(m.ff.rp)
cp.table.class <- m.ff.rp[["cptable"]]
# total variance explained is sum of the CP for all split
sum(cp.table.class[,"CP"])

[1] 0.51
```



The *root node error* shown in the summary is the proportion of observations that are *not* in the majority class, i.e., the class that would be predicted everywhere with a null model (no splits). This corresponds to a *relative error* $\equiv$ 1; the other errors are reductions from this.

We can compute this directly:

```
(n <- length(m.ff.rp$y))

[1] 155

(class.majority <- which.max(m.ff.rp$parms$prior))

1
1

(class.majority.proportion <-
     m.ff.rp$parms$prior[class.majority])

        1
0.5419355

(1 - (class.majority.proportion))

        1
0.4580645
```

Q39 : *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 A39* •

TASK 50 : Prune the tree to the complexity parameter corresponding to the minimum cross-validation error. •

```
cp.ix.class <- which.min(cp.table.class[,"xerror"])
(cp.min.class <- cp.table.class[cp.ix.class,"CP"])

[1] 0.01408451

(m.ff.rpp <- prune(m.ff.rp, cp=cp.min.class))

n= 155

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

  1) root 155 71 1 (0.54193548 0.30967742 0.14838710)
    2) elev< 7.558 40  2 1 (0.95000000 0.00000000 0.05000000) *
    3) elev>=7.558 115 67 2 (0.40000000 0.41739130 0.18260870)
      6) elev< 9.084 87 43 1 (0.50574713 0.32183908 0.17241379)
       12) soil=2 29  3 1 (0.89655172 0.10344828 0.00000000) *
       13) soil=1,3 58 33 2 (0.31034483 0.43103448 0.25862069)
         26) dist< 0.0088282 6  0 1 (1.00000000 0.00000000 0.00000000) *
         27) dist>=0.0088282 52 27 2 (0.23076923 0.48076923 0.28846154)
           54) elev>=8.812 9  3 1 (0.66666667 0.22222222 0.11111111)
            108) dist< 0.2580445 6  0 1 (1.00000000 0.00000000 0.00000000) *
            109) dist>=0.2580445 3  1 2 (0.00000000 0.66666667 0.33333333) *
           55) elev< 8.812 43 20 2 (0.13953488 0.53488372 0.32558140)
            110) elev>=8.446 16  4 2 (0.12500000 0.75000000 0.12500000)
              220) dist< 0.3423235 14  2 2 (0.14285714 0.85714286 0.00000000) *
              221) dist>=0.3423235 2  0 3 (0.00000000 0.00000000 1.00000000) *
            111) elev< 8.446 27 15 3 (0.14814815 0.40740741 0.44444444)
              222) elev< 7.821 14  6 2 (0.28571429 0.57142857 0.14285714) *
              223) elev>=7.821 13  3 3 (0.00000000 0.23076923 0.76923077) *
      7) elev>=9.084 28  8 2 (0.07142857 0.71428571 0.21428571)
       14) elev< 9.947 23  4 2 (0.08695652 0.82608696 0.08695652) *
       15) elev>=9.947 5  1 3 (0.00000000 0.20000000 0.80000000) *
```

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` < 7.56, for the lower-elevation points, 95% of the observations are class 1, 0% are class 2, and 5% in class 3[21]. For the higher-elevation points at this split 40% of the observations are class 1, 42% are class 2, and 18% in class 3.

Thus the lower-elevation points at this split are well-classified, whereas the others need more levels of the tree to classify them.

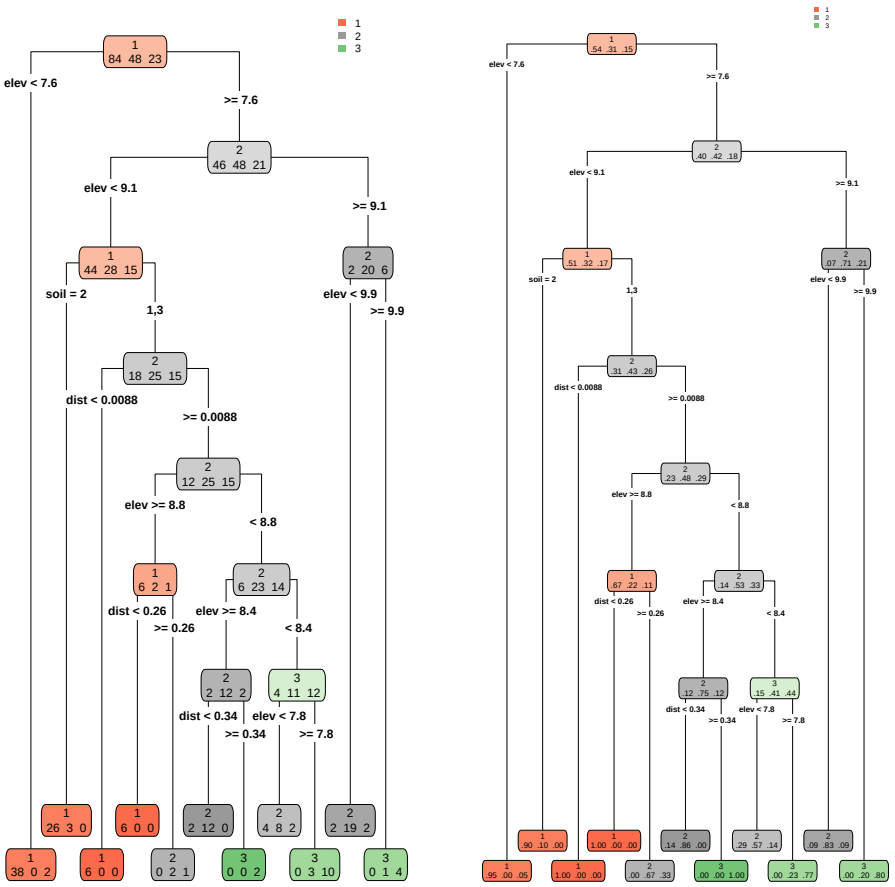
TASK 51 : Display the classification tree as a graph (1) showing the number of observations at each node and leaf; (2) showing the proportion of each class in each node and leaf. •

This graph shows the majority class at each node in a different colour. The intensity of the colour (its chroma) is greater as the node purity increases. The left-hand graph shows the counts, the right-hand graph the proportions.

```
par(mfrow=c(1,2))
rpart.plot(m.ff.rpp, type=4, extra=1, tweak = 0.8)
rpart.plot(m.ff.rpp, type=4, extra=4, tweak = 0.8)
```

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

```
par(mfrow=c(1,1))
```



Q40 : *How successful is this tree in modelling flooding frequency? Explain why.* *Jump to A40* •

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

9.3 Random forests for continuous predictands

In this section we discuss random forests for continuous predictands; this is an extension of the regression trees explained in §9.1.

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." [13, 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.* [13, §15] (advanced) or James *et al.* [17, §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 [9], Shalizi [27] or Hastie *et al.* [13, §8.2].

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

9.3.1 Building and evaluating a random forest model

TASK 52 : Build a random forest to model $\log_{10}$Zn from the flooding frequency, normalized distance to river, relative elevation, and soil type. •

These are chosen because they are available in the point data set and the prediction grid. Also, we have some theory about why they might be related to metal concentration.

Random forests are implemented with several R packages. We will use the `ranger` package: "A fast implementation of Random Forests, par-

ticularly suited for high dimensional data", even though in this small example we have a small dataset and few predictors. The `ranger` function computes the model; `print` applied to an object of class `ranger` displays the results.

The `ranger` function has many control parameters, which you should review[22] before any serious model building, and also consult the recommended textbooks for advice on the effect of various decisions about the parameters.

For example, the `importance` argument specifies whether and how the importance of each predictor is measured. Here we specify `"permutation"`, which is explained below.

Note that `ranger` requires a `data.frame` class for the data source (see `?ranger`), so we have to convert the `sf` object to a `data.frame` with the `as.data.frame` function.

```
library(ranger)
m.lzn.rf <- ranger(logZn ~ ffreq + dist+ elev + soil,
                   data = as.data.frame(meuse.sf),
                   importance = "permutation")
print(m.lzn.rf)

Ranger result

Call:
 ranger(logZn ~ ffreq + dist + elev + soil, data = as.data.frame(meuse.sf),      importance =

Type:                             Regression
Number of trees:                  500
Sample size:                      155
Number of independent variables:  4
Mtry:                             2
Target node size:                 5
Variable importance mode:         permutation
Splitrule:                        variance
OOB prediction error (MSE):       0.02263737
R squared (OOB):                  0.7696829

print(sqrt(m.lzn.rf$prediction.error))

[1] 0.1504572

print(m.lzn.rf$r.squared)

[1] 0.7696829
```

The summary shows how well the out-of-bag model fits match the actual data. First we have the *root mean-squared error* 0.15, in the same units as the target variable, i.e., $\log_{10}$(mg) kg^{-1}. Compared to the mean value of the dataset 2.556 $\log_{10}$(mg) kg^{-1}, this is quite a small relative error.

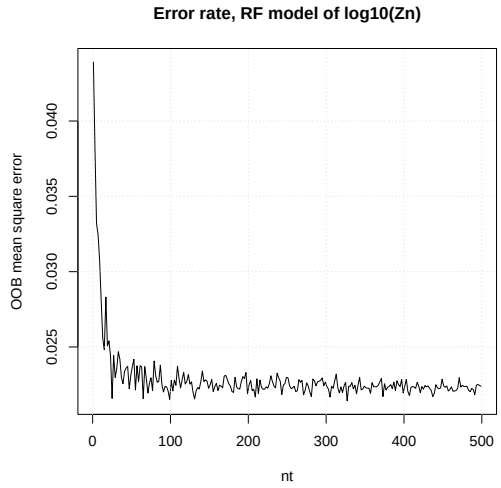
Second we have the overall goodness-of-fit of the OOB values, measured against a 1:1 line with the actual values. This is the proportion of variance explained, i.e., R^2. Here the value is 0.77

One question is whether the default number of trees (optional parameter `mtry`, default value 500, see `?ranger`) is sufficient. We can examine this

[22] `?ranger`

by plotting the the error rate, i.e., mean square error of the predictions of the out-of-bag observations, vs. the number of trees. To do this we recompute the model for a sequence of tree sizes and graph the results. Here the `plot` function is applied to a `ranger` object.

```
nt <- seq(1, 500, by = 2)
oob_mse <- vector("numeric", length(nt))
for(i in 1:length(nt)) {
  rf <- ranger(logZn ~ ffreq + dist+ elev + soil, meuse,
  num.trees = nt[i],
  write.forest = FALSE, importance = "none")
  oob_mse[i] <- rf$prediction.error
}
plot(x = nt, y = oob_mse, type = "l",
  main = "Error rate, RF model of log10(Zn)",
  ylab = "OOB mean square error")
grid()
```



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

While building the forest, if the optional `importance` argument is set to `"permutation"`, 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.

TASK 53 : Print and display the variable importances, as measured by the increase in mean-squared error, as well as their percentage of the OOB RMSE. •

We do this with the `importance` functions. This shows the increase in mean-square error of the predictions, if a single variable is permuted. Here we show its square root, i.e., the RMSE, which can be compared with the best-model RMSE.

```
print(sort(sqrt(importance(m.lzn.rf)), decreasing = TRUE))

      dist       elev      ffreq       soil
0.27454213 0.17669211 0.09515244 0.08512463

print(round(sort(100*sqrt(importance(m.lzn.rf))/sqrt(rf$prediction.error),
                 decreasing = TRUE)))

 dist  elev ffreq  soil
  183   118    64    57
```

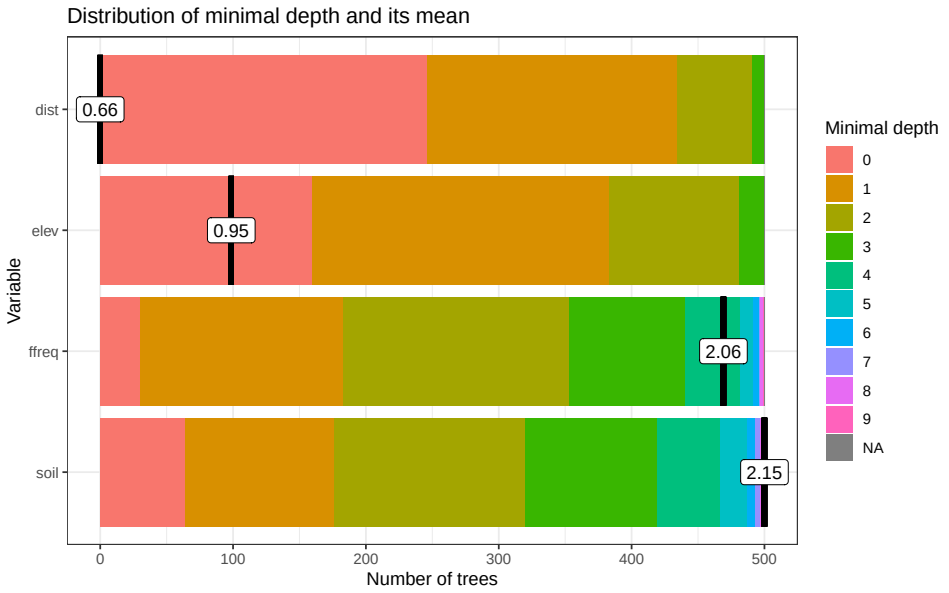
Q42 : *Which variables are most important? Compare with the single pruned regression tree.* *Jump to A42* •

9.3.2 * A deeper investigation of variable importance

Another way to look at variable importance is by the depth in the tree at which each predictor is used. This is found with the `min_depth_frame` function of the `randomForestExplainer` package.[23].

```
require(randomForestExplainer)

Loading required package:  randomForestExplainer

Registered S3 method overwritten by 'GGally':
method from
+.gg   ggplot2

min_depth_frame <- min_depth_distribution(m.lzn.rf)
str(min_depth_frame)  # has results for all the trees

'data.frame': 1999 obs. of  3 variables:
 $ tree         : int  1 1 1 1 2 2 2 2 3 3 ...
 $ variable     : chr  "dist" "elev" "ffreq" "soil" ...
 $ minimal_depth: num  2 1 0 1 0 1 3 1 2 1 ...

plot_min_depth_distribution(min_depth_frame)
```

[23] This package has a tutorial, see https://cran.rstudio.com/web/packages/randomForestExplainer/vignettes/randomForestExplainer.html



This shows the minimum depth (i.e., closest to root = 0) at which a predictor is used in the set of trees, along with the average depth.

Q43 : *Which predictor is used most as the root? At the second level?* *Jump to A43* •

We can see this information as a table with some more details:

```
importance.frame <- measure_importance(m.lzn.rf)
print(importance.frame)

  variable mean_min_depth no_of_nodes permutation no_of_trees
1     dist       0.658000        8573 0.075373380         500
2     elev       0.952000        8873 0.031220103         500
3    ffreq       2.056000        2450 0.009053987         500
4     soil       2.147996        1712 0.007246203         499
  times_a_root p_value
1          246       0
2          160       0
3           30       1
4           64       1
```

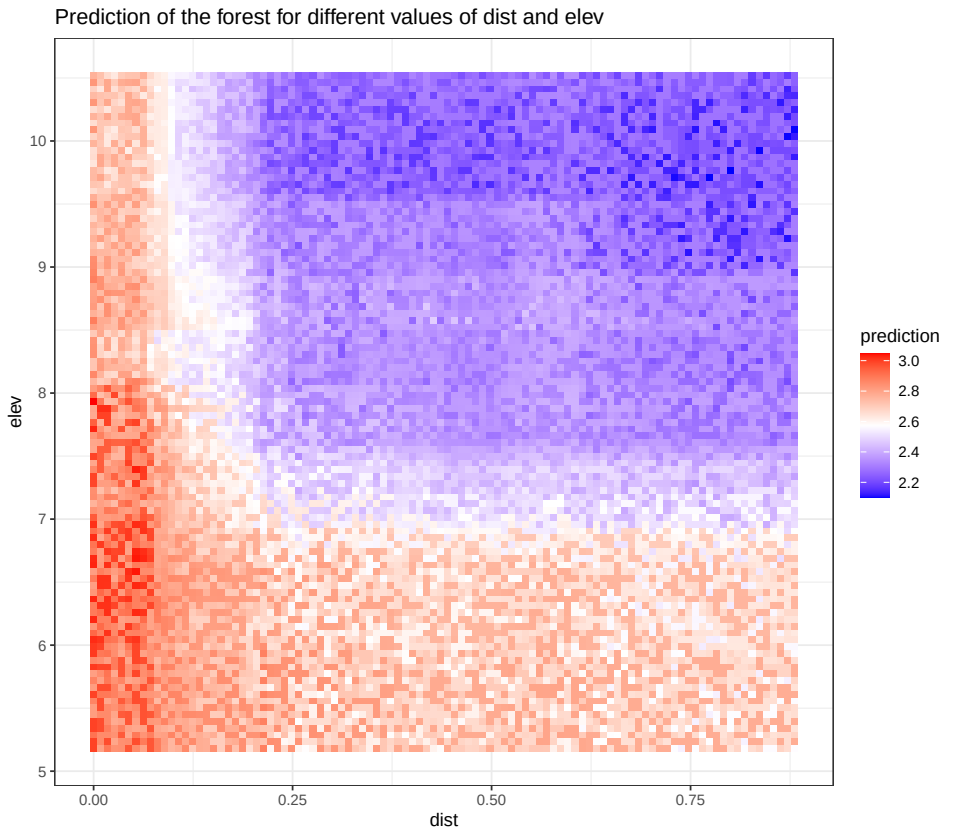
The p-value measures whether the observed number of successes (number of nodes in which X_j was used for splitting) exceeds the theoretical number of successes if they were random, following a binomial distribution.

TASK 54 : Plot the predictions resulting from the interactions between distance and elevation, as a 2.5D plot. •

We use the function `plot_predict_interaction` function of the `randomForestExpla` package to plot the prediction of our forest on a grid of values for the components of each interaction.

Here we see the predicted log(Zn) content for all combinations of distance from river and elevation:

```
plot_predict_interaction(m.lzn.rf, meuse, "dist", "elev")
```



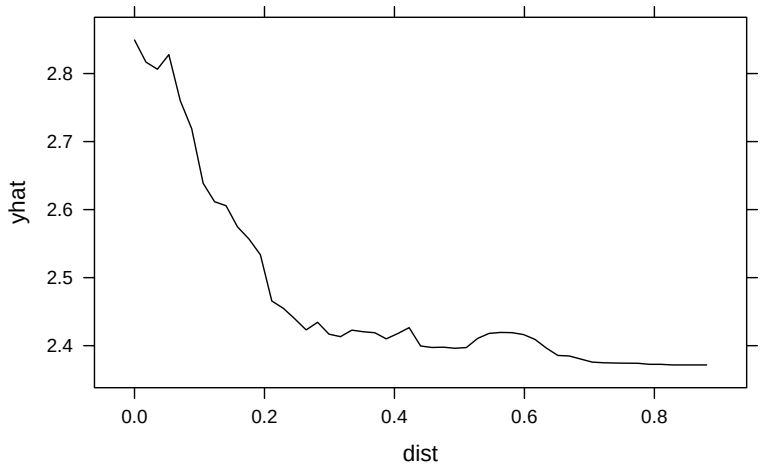
Q44 : *What is the overall pattern of the interaction?* *Jump to A44* •

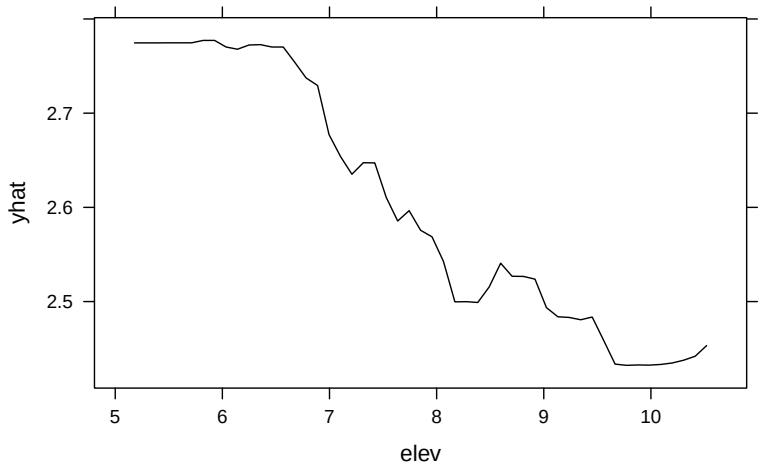
Another useful diagnostic is the *Partial dependence plot*. This shows the effect of one predictor on the predictand, holding all the others **constant** at their **median** value. This is implemented by the `partial` function of the `pdp` "Partial Dependence Plots" package.

We show these for the two most important predictors.

TASK 55 : Display the partial dependence plots for distance and elevation. •

```
require(pdp)

Loading required package:  pdp

partial.dist <- pdp::partial(m.lzn.rf, pred.var = "dist", pred.data = meuse.sf)
plotPartial(partial.dist)
partial.elev <- pdp::partial(m.lzn.rf, pred.var = "elev", pred.data = meuse.sf)
plotPartial(partial.elev)
```

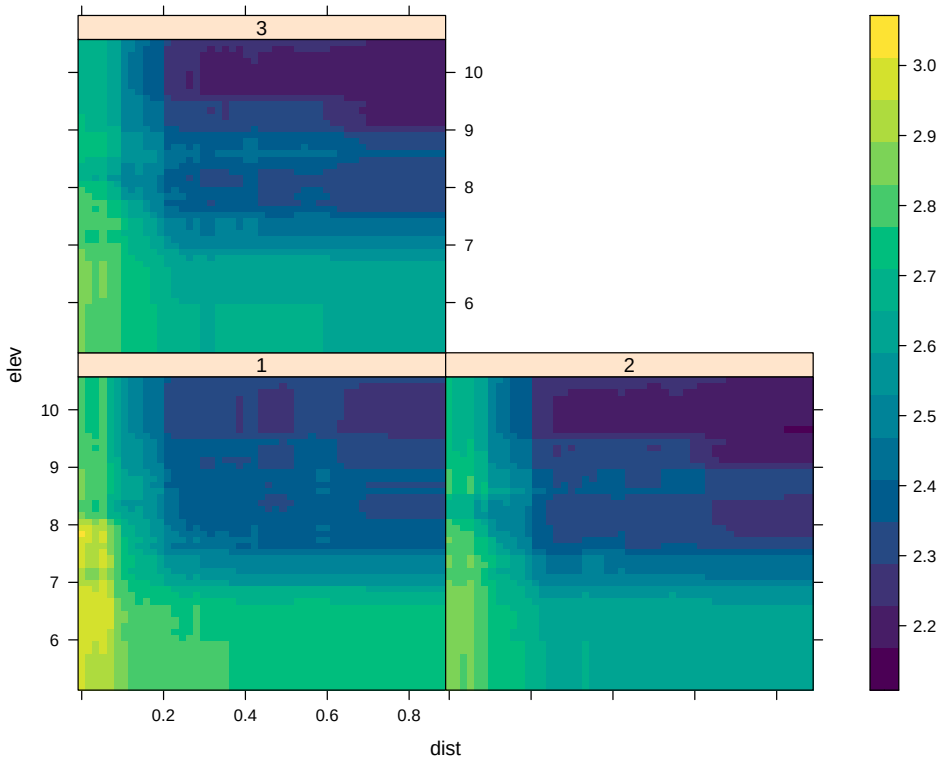




Q45 : *What is the effect of the distance to river on the RF prediction of log_{10}Zn, when the other variables are held at their mean values. Is the effect monotonic (i.e., consistently increasing or decreasing)? Why or why not?* *Jump to A45* •

The joint dependence can also be plotted. With this package we can see the interaction separately for each flood frequency class.

```
partial.dist.elev <- pdp::partial(m.lzn.rf, pred.var = c("dist", "elev", "ffreq"),
                                  pred.data = meuse.sf)
plotPartial(partial.dist.elev)
```



Q46 : *What is the pattern of the joint dependence of distance and elevation as the flood frequency decreases?* *Jump to A46* •

9.3.3 Prediction with the random forest

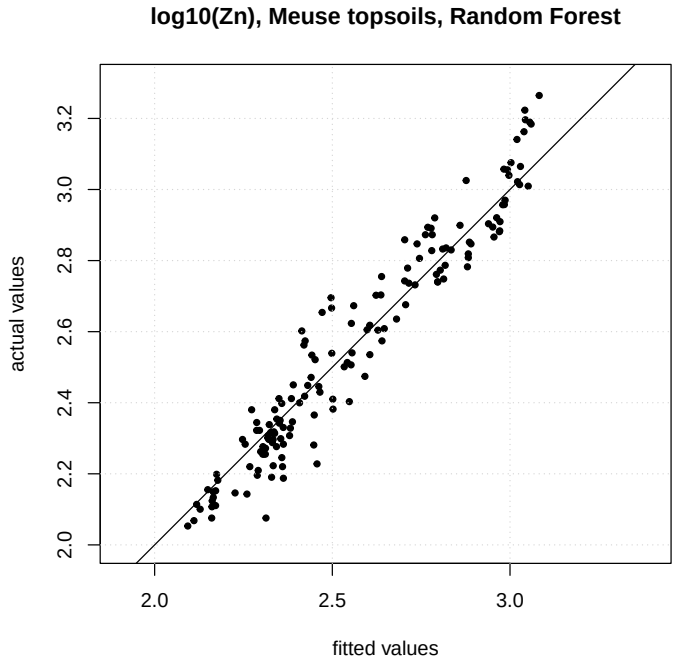
Of course we want to use our model for prediction, here at the known points:

TASK 56 : 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 `ranger` 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 argument `data`. In this case we use the known values in the `meuse.sf` object.

```
p.rf <- predict(m.lzn.rf, data=as.data.frame(meuse.sf))
str(p.rf)

List of 5
 $ predictions              : num [1:155] 3.05 2.98 2.75 2.5 2.47 ...
 $ num.trees                : num 500
 $ num.independent.variables: num 4
 $ num.samples              : int 155
 $ treetype                 : chr "Regression"
 - attr(*, "class")= chr "ranger.prediction"
```

```
length(unique(p.rf$predictions))

[1] 155

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

    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.23785 -0.05276 -0.01752 -0.00188  0.04547  0.19904

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

[1] 0.08313264

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

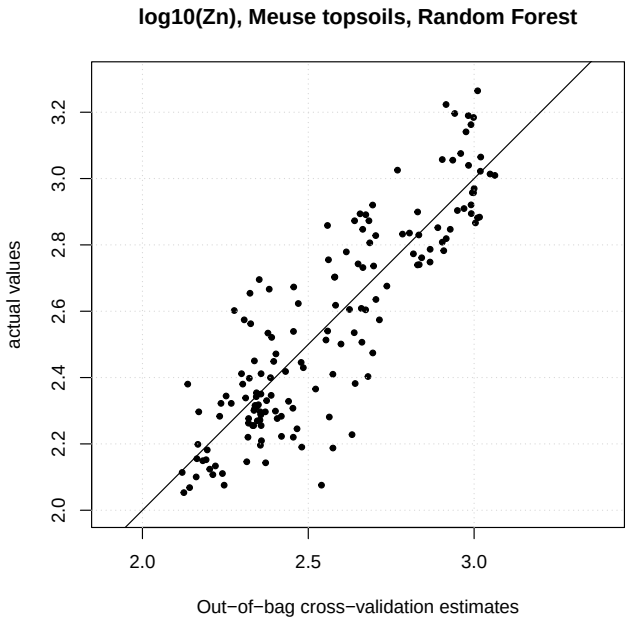


Q47 : *Compare this 1:1 plot (actual vs. fitted) to that produced by the regression tree. Which would you prefer for prediction at unknown points? Why?* *Jump to A47* •

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. These are automatically computed when `ranger` builds the random forest. These are in field `predictions` of the fitted model.

TASK 57 : 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. •

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

     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-0.464324 -0.094993 -0.032961 -0.004212  0.092266  0.343349

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

[1] 0.1504572

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



Q48 : *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 A48* •

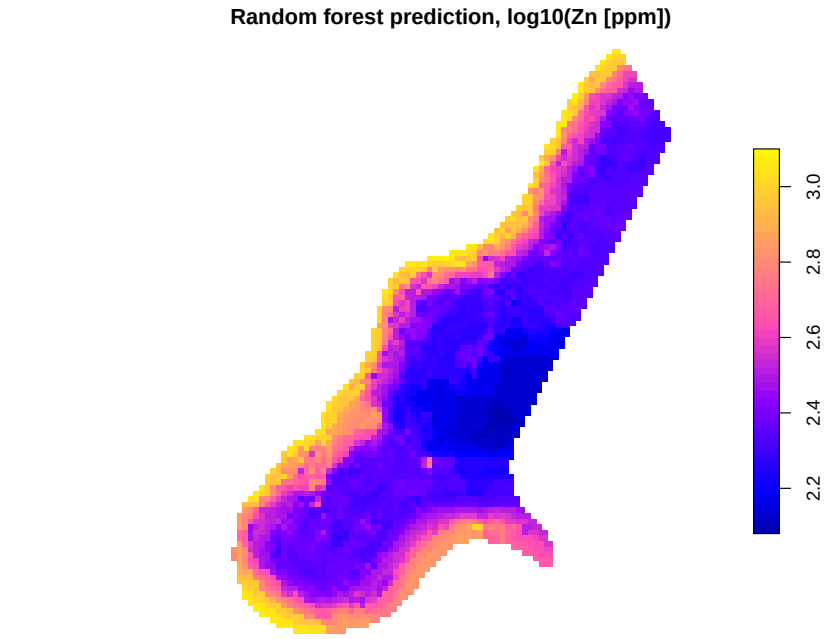
9.3.4 Predicting over the study area with the random forest model

Now that we have a predictive model, we can use it to map the study area.

TASK 58 : Use the random forest model to map over the prediction grid. •

```
rf.grid <- predict(m.lzn.rf, data=as.data.frame(meuse.grid.sf))
meuse.grid.sf$rf.pred <- rf.grid$predictions; rm(rf.grid)
```

```
plot(meuse.grid.sf["rf.pred"], pch = 15,
     nbreaks = 64,
     main="Random forest prediction, log10(Zn [ppm])")
```



Q49 : *Compare this map to that from the regression tree. What are the major differences?* *Jump to A49* •

Challenge: Build another random forest and compare its predictions and variable importances to the first one. How much difference do you expect? Was the difference as you expected? Can you explain why this is much more stable than the regression tree?

9.4 * 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 §9.2. The output is somewhat different that for continuous predictands.

TASK 59 : Build a random forest model of flooding frequency class based on normalized distance to river and elevation, using the default number of trees. •

The variable importance in this case is measured by class impurity, i.e., the Gini index. So we specify `"impurity"` as the option for the `importance` argument to `ranger`. This is how often a *randomly chosen* training observation would be *incorrectly* assigned if it were *randomly labeled* according to the *frequency distribution* of classes in the training dataset.

```
m.ff.rf <- ranger(ffreq ~ dist + elev + soil,
                  data = as.data.frame(meuse.sf),
                  importance = "impurity",
                  write.forest = FALSE)
print(m.ff.rf)

Ranger result

Call:
 ranger(ffreq ~ dist + elev + soil, data = as.data.frame(meuse.sf),      importance = "impuri

Type:                             Classification
Number of trees:                  500
Sample size:                      155
Number of independent variables:  3
Mtry:                             1
Target node size:                 1
Variable importance mode:         impurity
Splitrule:                        gini
OOB prediction error:             36.13 %

str(m.ff.rf)

List of 14
 $ predictions              : Factor w/ 3 levels "1","2","3": 1 1 2 1 1 1 1 2 2 1 ...
 $ num.trees                : num 500
 $ num.independent.variables: num 3
 $ mtry                     : num 1
 $ min.node.size            : num 1
 $ variable.importance      : Named num [1:3] 18.5 28.5 11.6
  ..- attr(*, "names")= chr [1:3] "dist" "elev" "soil"
 $ prediction.error         : num 0.361
 $ confusion.matrix         : 'table' int [1:3, 1:3] 71 15 6 12 28 17 1 5 0
  ..- attr(*, "dimnames")=List of 2
  .. ..$ true     : chr [1:3] "1" "2" "3"
  .. ..$ predicted: chr [1:3] "1" "2" "3"
 $ splitrule                : chr "gini"
 $ treetype                 : chr "Classification"
 $ call                     : language ranger(ffreq ~ dist + elev + soil, data = as.data.fram
 $ importance.mode          : chr "impurity"
 $ num.samples              : int 155
 $ replace                  : logi TRUE
 - attr(*, "class")= chr "ranger"

table(m.ff.rf$predictions)


 1  2  3
92 57  6

print(m.ff.rf$prediction.error)

[1] 0.3612903

print(sort(importance(m.ff.rf, type=1), decreasing = TRUE))

    elev     dist     soil
28.45325 18.53250 11.57283
```

TASK 60 : 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 <- m.ff.rf$predictions
table(meuse$ffreq) # observed


 1  2  3
84 48 23

table(p.rf)        # predicted

p.rf
 1  2  3
92 57  6

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

         Observed
Predicted  1  2  3
        1 71 15  6
        2 12 28 17
        3  1  5  0
```

> **Note:** The confusion matrix is automatically stored in the fitted model and can be retrieved as the `confusion.matrix` field.

We follow Lark [18] 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, 0 of the total 6 observations predicted to be in flood frequency class 3 were actually in that class. This gives a map unit purity of 0%.

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

 1  2  3
71 28  0

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

```
 1  2  3
92 57  6

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

        1         2         3
0.7717391 0.4912281 0.0000000
```

Q50 : *(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 A50*

•

Challenge: There are two other possible predictors that we have not used in the machine learning models: the coördinates of the points. The coördinates of each prediction point (grid cell) are also known. What might be value of including them as predictors? What might be the differences in the resulting maps? You can build models including these predictors and see the results. Note you will need to add the coördinates to the data frame; they are now only in the special `geometry` field.

10 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.

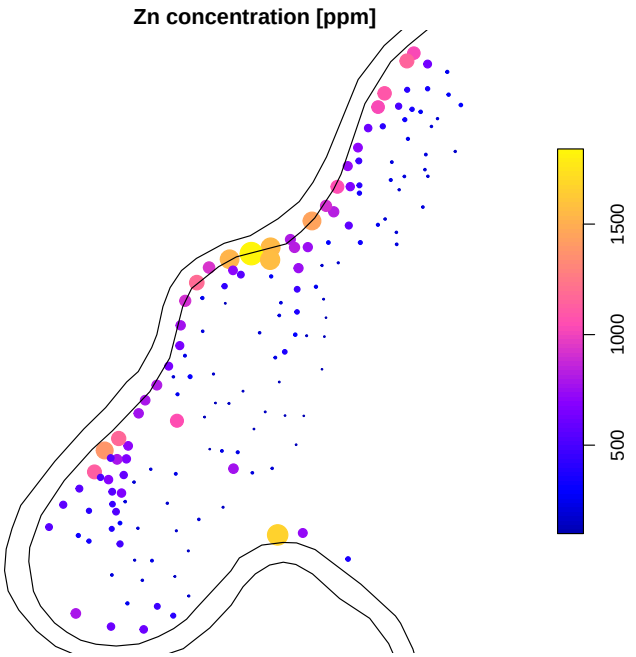
10.1 Assessing spatial dependence with the empirical variogram

---

Task 61 : 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.sf["zinc"],
     reset = FALSE,
     nbreaks = 64, pch = 20,
     cex=4*meuse.sf$zinc/max(meuse.sf$zinc),
     main = "Zn concentration [ppm]")
plot(meuse.riv.sf, add = TRUE)
```



Q51 : *Do nearby observations seem to more similar to each other than would be expected at random?* *Jump to A51* •

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 62 : 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.sf$logZn)
n*(n-1)/2

[1] 11935
```

Q52 : *How many unique* pairs *of points are there in this dataset?* *Jump to A52* •

TASK 63 : 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(st_coordinates(meuse.sf))

[1] 155   2

st_coordinates(meuse.sf)[1,]

     X      Y
181072 333611

st_coordinates(meuse.sf)[2,]

     X      Y
181025 333558

(sep <- dist(st_coordinates(meuse.sf)[1:2,]))

         1
2 70.83784

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

[1] 0.001144082
```

Q53 : *What is the separation and semivariance for this point-pair?* *Jump to A53* •

10.1.1 * The variogram cloud

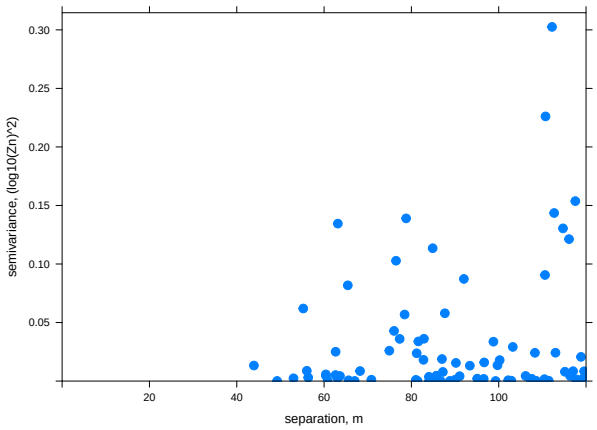
We can see the individual semivariances with the **variogram cloud**. Since there are so many point-pairs, this is difficult to interpret. However, we can show just the closest pairs of points.

TASK 64 : Display a variogram cloud for point-pairs separated by less than 72 m, for the $\log_{10}$Zn concentration. •

The `variogram` function, with the optional argument `cloud` set to `TRUE`, computes the variogram cloud. The optional `cutoff` argument sets the maximum separation for which semivariances should be computed.

```
head(vc <- variogram(logZn ~ 1, meuse.sf, cutoff=120, cloud=TRUE))

       dist        gamma dir.hor dir.ver   id left right
1  70.83784 1.144082e-03       0       0 var1    2     1
2 118.84864 2.065953e-02       0       0 var1    3     1
```

```
3 108.57716 1.796278e-04       0       0 var1    6     5
4  67.00746 9.815006e-05       0       0 var1   11    10
5  93.39165 1.313838e-02       0       0 var1   14     9
6  85.44589 3.675269e-04       0       0 var1   15     9

class(vc)

[1] "variogramCloud" "data.frame"

print(plot(vc, pch = 20, cex=2, xlim=c(0,120),
     ylab="semivariance, (log10(Zn)^2)", xlab="separation, m"))
```



Q54 : *Do these close-by points in general have low semi-variance, i.e., do they have similar values? Are there exceptions? Which pair of close-by points has the largest semi-variance?* *Jump to A54* •

This gives us a chance to investigate the `order` "ordering permutation" and `which.max` "index in vector of the maximum element" functions.[24].

```
order(vc$gamma, decreasing=TRUE)

 [1] 60 54 55 46 50 69 47 53 45 52 68 70 41 43 74 56 11 48 62 75 61 34
[23] 23 10 51 33 35  2  9 65  7 72 64 63 40  5 42 76 77 44 29 17 57 18
[45] 31 37 39 78 32 67 71 73 27 12 38 16 30 15 28 26  1 58 20 13 21 49
[67]  6  8 22  3 25  4 19 14 79 36 59 24 66

order(vc$gamma, decreasing=TRUE)[1]

[1] 60

which.max(vc$gamma)

[1] 60

(unusual.pair <- vc[which.max(vc$gamma),])

       dist     gamma dir.hor dir.ver   id left right
60 112.2185 0.3025447       0       0 var1  125    59

meuse[138, "logZn"]; meuse[76, "logZn"]

[1] 2.313867
[1] 2.832509
```

[24] Naturally, there is also a `which.min` function

```
(gamma.76.138 <- 0.5 * (meuse[138, "logZn"] - meuse[76, "logZn"])^2)

[1] 0.1344946
```

10.1.2 The empirical variogram

! → 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* ...".

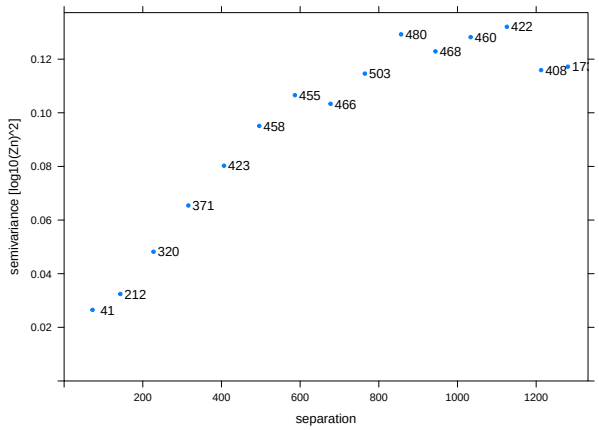
TASK 65 : Plot the experimental variogram of the log-Zn concentrations, i.e. the average semi-variances of the point-pairs versus average distance (also known as the "lag"), with a bin width of 90 m, to a maximum lag distance ("cutoff") of 1300 m[25]. •

The `variogram` function computes the experimental variogram. The optional `cutoff` argument sets the maximum separation for which semivariances should be computed, and the optional `width` argument sets the bin width.

```
(v <- variogram(logZn ~ 1, meuse.sf, cutoff=1300, width=90))

    np       dist      gamma dir.hor dir.ver   id
1   41   72.24836 0.02649954       0       0 var1
```

[25] By default, the `variogram` function uses a cutoff equal to 1/3 of the maximum distance across the diagonal of the bounding box (see `st_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.

```
2  212  142.88031 0.03242411       0       0 var1
3  320  227.32202 0.04818895       0       0 var1
4  371  315.85549 0.06543093       0       0 var1
5  423  406.44801 0.08025949       0       0 var1
6  458  496.09401 0.09509850       0       0 var1
7  455  586.78634 0.10656591       0       0 var1
8  466  677.39566 0.10333481       0       0 var1
9  503  764.55712 0.11461332       0       0 var1
10 480  856.69422 0.12924402       0       0 var1
11 468  944.02864 0.12290106       0       0 var1
12 460 1033.62277 0.12820318       0       0 var1
13 422 1125.63214 0.13206510       0       0 var1
14 408 1212.62350 0.11591294       0       0 var1
15 173 1280.65364 0.11719960       0       0 var1

print(plot(v, plot.numbers = TRUE, pch = 20,
           xlab = "separation", ylab = "semivariance [log10(Zn)^2]"))
```



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.

Q55 : *What is the **average separation** and **average semivariance** in the first bin? How many **point-pairs** are averaged for this?* *Jump to A55* •

Q56 : *What evidence does this plot give that closer points are more similar, i.e., what is the evidence for **local spatial dependence**?* *Jump to A56* •

range of spatial dependence

Q57 : *At what separation between point-pairs is there no more spatial dependence? (This is called the **range** of spatial dependence)* *Jump to A57* •

Q58 : *What is the semivariance at zero separation? (This is called the **nugget**). Why is it not zero?* *Jump to A58* •

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.

---

variogram sill

Q59 : *What is the semivariance at the range and beyond? (This is called the **total sill**)* *Jump to A59* •

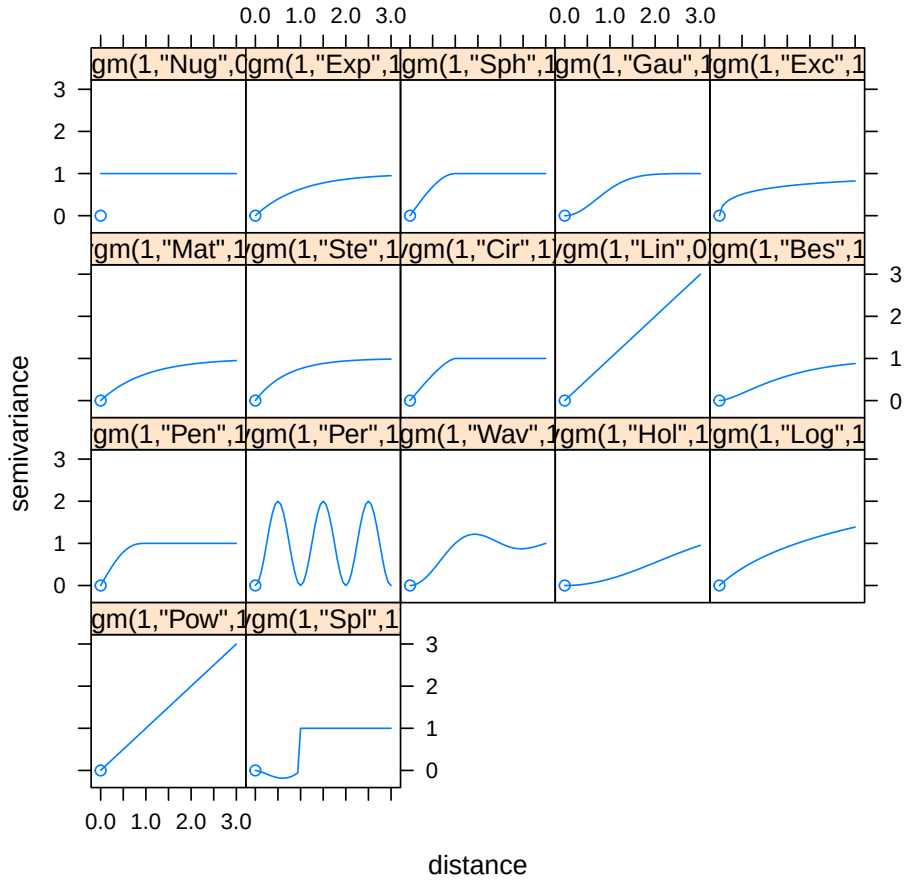
10.2 Building a variogram model

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 66 : 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.

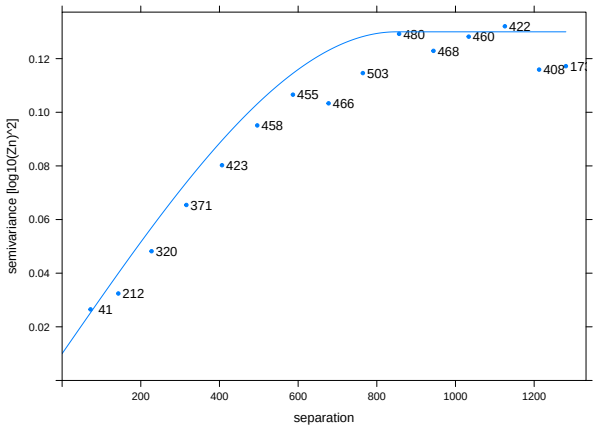
In addition, the whole variogram is raised by a **nugget** variance.

> **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 67 : 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:

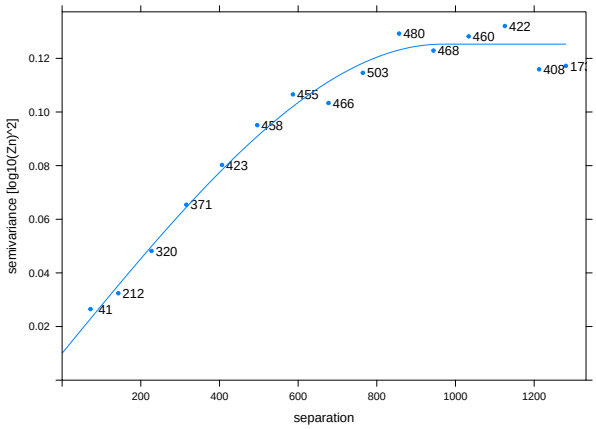
```
vm <- vgm(psill=0.12,model="Sph",range=850,nugget=0.01)
print(plot(v, pl=T, model=vm, pch=20,
           xlab = "separation", ylab = "semivariance [log10(Zn)^2]"))
```



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 0.01004123   0.0000
2   Sph 0.11525698 967.2634

print(plot(v, pl = T, model = vmf, pch = 20,
```

```
      xlab = "separation", ylab = "semivariance [log10(Zn)^2]"))
```



Q60 : *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 A60*

•

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

Challenge: Compute an empirical variogram of $\log_{10}$Zn with shorter or longer range, and different bin sizes. Fit a spherical variogram model to these empirical variograms. How much *should* the models differ from each other and from the single model we built in this section? Why? How much *do* the models differ? Why?

11 Mapping by Ordinary Kriging 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**.

11.1 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:

$$\mathbf{A}\lambda = \mathbf{b} \quad (12) \qquad \mathbf{A} = \begin{bmatrix} \gamma(\mathbf{x}_1,\mathbf{x}_1) & \gamma(\mathbf{x}_1,\mathbf{x}_2) & \cdots & \gamma(\mathbf{x}_1,\mathbf{x}_N) & 1 \\ \gamma(\mathbf{x}_2,\mathbf{x}_1) & \gamma(\mathbf{x}_2,\mathbf{x}_2) & \cdots & \gamma(\mathbf{x}_2,\mathbf{x}_N) & 1 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \gamma(\mathbf{x}_N,\mathbf{x}_1) & \gamma(\mathbf{x}_N,\mathbf{x}_2) & \cdots & \gamma(\mathbf{x}_N,\mathbf{x}_N) & 1 \\ 1 & 1 & \cdots & 1 & 0 \end{bmatrix}$$

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

This system shows that the semivariance γ must be known for the prediction point $\mathbf{x}_0$ and all observation points $\mathbf{x}_i, i = 1 \ldots N$ (this is the $\mathbf{b}$ vector) and also between all pairs $\mathbf{x}_i, \mathbf{x}_j$ 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 the LaGrange multiplier ψ, which depends on covariance structure of the sample points:

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

This expression (Eqn. 15) is what is minimized by the selection of kriging weights λ by solution of Eqn. 13.

! → 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.

11.2 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. The predictions are at the *centre points* of each grid cell, on the same **geostatistical support**, i.e., physical sample size, as the original observations. In this case (see §A) the samples were taken on a support of 15x15 m, so in the kriging interpolations which follow the predictions are on the same support. This is taken as the value throughout the grid cell, and in particular the kriging prediction variance refers to the average on that support. To have the same prediction support as observation support, we would create a grid with the observation resolution, in this case 15x15 m.

For this sample dataset, a regular grid has been prepared, named `meuse.grid`; this was converted to a spatial object in §10, above.

> **Note:** It is possible to obtain an average value and its kriging prediction variance on any block size, using the `block` optional argument of the `krige` function. However in this case the change of support is complicated by the fact that the original observations are on a block, not on a small "point" support whose physical dimensions can be ignored in block kriging. See Webster & Oliver [28] for the mathematics of block kriging and change of support.

Task 68 : Predict the attribute value at all grid points using Ordinary Kriging. •

The `krige` function 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.sf,
             newdata = meuse.grid.sf,
             model = vmf)

[using ordinary kriging]
```

As with the empirical variogram plot (§10) 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`.

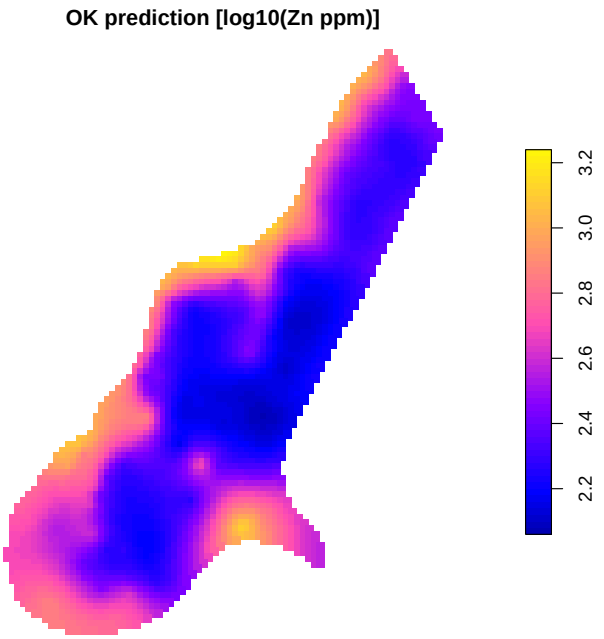
Task 69 : Display the structure of the kriging prediction object. •

```
str(k40)

Classes 'sf' and 'data.frame': 3103 obs. of  3 variables:
 $ var1.pred: num  2.83 2.88 2.83 2.78 2.94 ...
 $ var1.var : num  0.0596 0.0471 0.0509 0.055 0.0335 ...
 $ geometry :sfc_POINT of length 3103; first list element:  'XY' num  181180 333740
 - attr(*, "sf_column")= chr "geometry"
 - attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA
  ..- attr(*, "names")= chr [1:2] "var1.pred" "var1.var"
```

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 70 : Display the map of predicted values, usng a blue-pink-yellow colour ramp (smooth transition among colours showing different values). •

```
plot(k40["var1.pred"], pch=15,
     nbreaks = 64,
     main="OK prediction [log10(Zn ppm)]")
```



The `plot` "spatial plot" method of the `sf` package displays spatial objects.

The default `bpy.colors` "blue-pink-yellow colour scheme" colour ramp from the `sp` package also displays well if printed in gray scale.

```
head(sp::bpy.colors(64))

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

sp::bpy.colors(64)[32]

[1] "#C225DAFF"
```

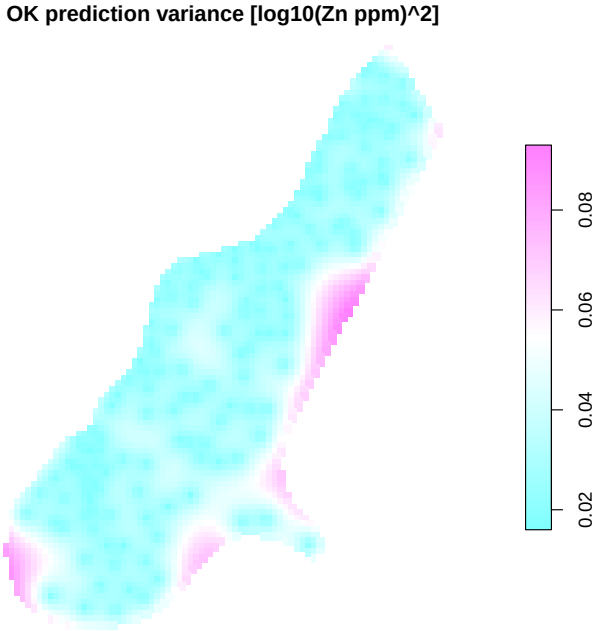
Each colour is made of four numbers, representing Red, Green, Blue and alpha (transparency) on $[0 \ldots 255]$, represented as hexidecimal characters, so FF is $(16 \cdot 16) - 1 = 255$. So for example the 32^{nd} colour, which is used for the midpoint of the scale, is C225DAFF; its red component is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

Q61 : *Describe the kriged map with respect to its (1) smoothness, (2) values at observed data points.* *Jump to A61* •

TASK 71 : 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.

```
plot(k40["var1.var"], pch=15,
     nbreaks = 64,
     pal = cm.colors,
     main="OK prediction variance [log10(Zn ppm)^2]")
```



Q62 : *Describe the variances map. Where is the prediction variance lowest? Does this depend on the data value?* *Jump to A62* •

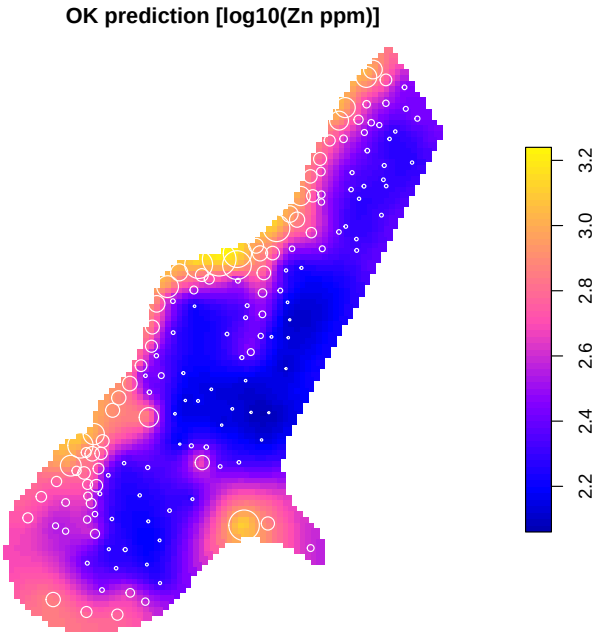
It may be helpful to visualize the predictions and their variances in relation to the observations.

To do this, we specify that the first plot can be over-printed with other elements, using the optional `reset` argument to `plot`. We then add a layer showing the observation points.

For the kriging prediction, we show the post-plot: value proportional to

circle size[26].

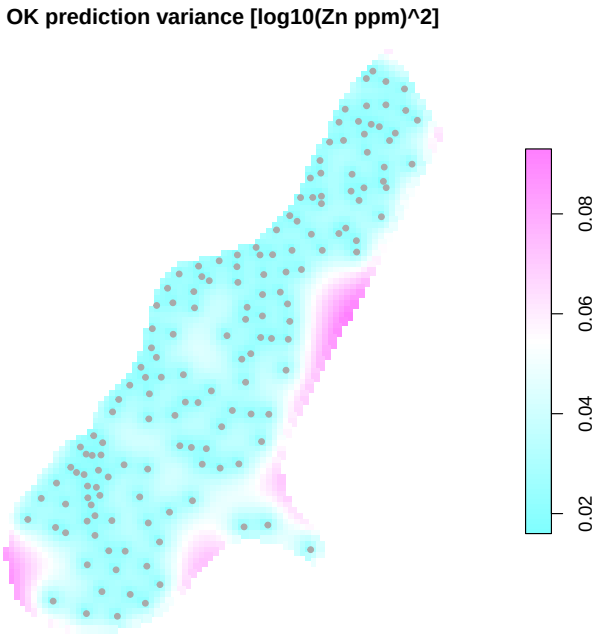
```
plot(k40["var1.pred"], pch=15,
     nbreaks = 64,
     reset = FALSE,  # allow further elements to be plotted
     main="OK prediction [log10(Zn ppm)]")
plot(meuse.sf["logZn"], add = TRUE, pch = 21,
     col = "white",
     cex=4*meuse.sf$zinc/max(meuse.sf$zinc))
```



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

```
plot(k40["var1.var"], pch=15,
     nbreaks = 64,
     reset = FALSE,  # allow further elements to be plotted
     pal = cm.colors,
     main="OK prediction variance [log10(Zn ppm)^2]")
plot(meuse.sf["logZn"], add = TRUE, pch = 20,
     col = "darkgray")
```

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



Challenge: Repeat the process of §10 (local spatial dependence) and §11 (OK) for one of the other metals (Cu, Pb, or Cd). What differences, if any, do you expect in the fitted variogram model? Why? Do you expect a similar spatial pattern to that for Zn? Why or why not? Compare the fitted variogram models and the OK predictions and their variances. Comment on the similarities and differences, and try to explain the reasons for these.

12 * 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[27], 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 72 : 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.sf$zn.i <- (meuse.sf$zinc < 150)
summary(meuse.sf$zn.i)

   Mode   FALSE    TRUE
logical     140      15
```

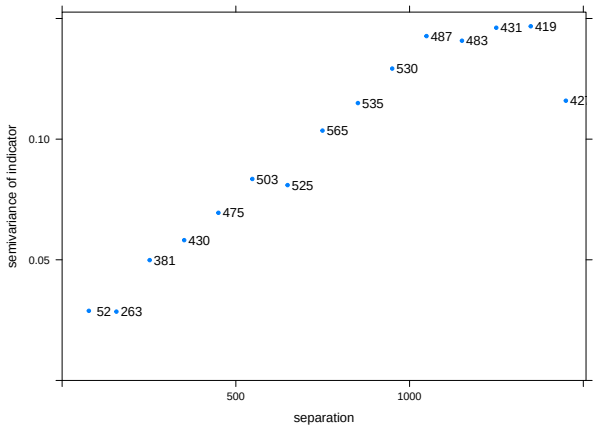
Q63 : *How many observations are above the threshold?* *Jump to A63* •

TASK 73 : Make an empirical **indicator variogram** and model it. •

The empirical variogram:

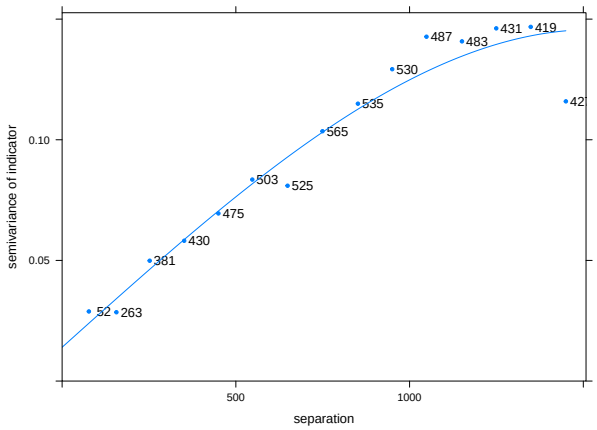
```
vi <- variogram(zn.i ~ 1, location=meuse.sf, cutoff=1500)
print(plot(vi, pl=T, pch = 20,
           xlab = "separation", ylab = "semivariance of indicator"))
```

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



With the fitted model:

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

  model      psill    range
1   Nug 0.01400961    0.000
2   Sph 0.13165108 1527.442

print(plot(vi, pl = T, model = vimf, pch = 20,
           xlab = "separation", ylab = "semivariance of indicator"))
```



Q64 : *What is the range of the indicator? Does this match that for the original variable?* *Jump to A64* •

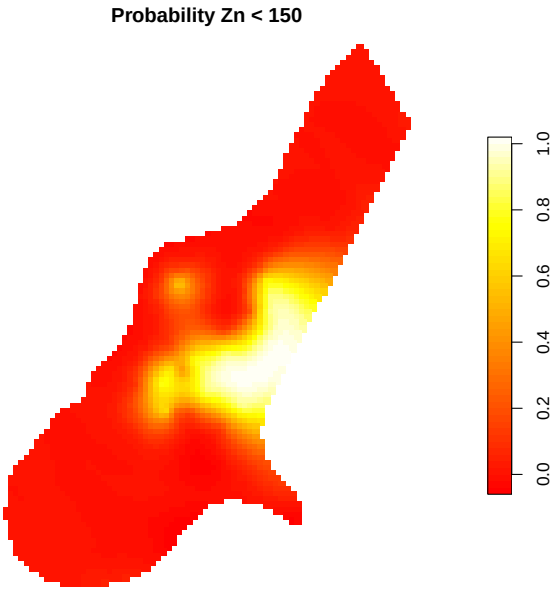
Q65 : *What is the sill? What are its units?* *Jump to A65* •

TASK 74 : 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.sf,
               newdata = meuse.grid.sf, model = vimf)
```

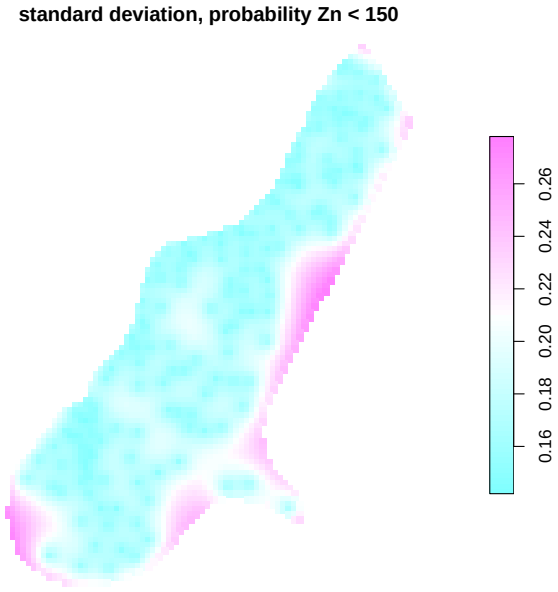
```
[using ordinary kriging]

plot(k40.i["var1.pred"], pch = 15,
     nbreaks = 64,
     pal = heat.colors,
     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`.

The kriging prediction variance is difficult to interpret: what is the variance of a probability? At any rate we can see where the prediction is more reliable.

```
k40.i$var1.sd <- sqrt(k40.i$var1.var)
plot(k40.i["var1.sd"], pch = 15,
     nbreaks = 64,
     pal = cm.colors,
     main="standard deviation, probability Zn < 150")
```



The decision-maker can use the probability map directly at any risk level.

TASK 75 : Show the "safe" areas, as defined as 80% probability of being below the critical level. •

```
k40.i$safe150.80pct <- ifelse((k40.i$var1.pred <= 0.8), TRUE, FALSE)
summary(k40.i$safe150.80pct)

   Mode   FALSE    TRUE
logical     241    2862

plot(k40.i["safe150.80pct"], pch = 15, pal = c("darkgreen", "red"),
             main="(p >= 0.8) below critical level (Zn < 150)")
```

Q66 : *What parts of the study area are 'safe' for agricultural land use?*
Jump to A66 •

13 Mixed predictors

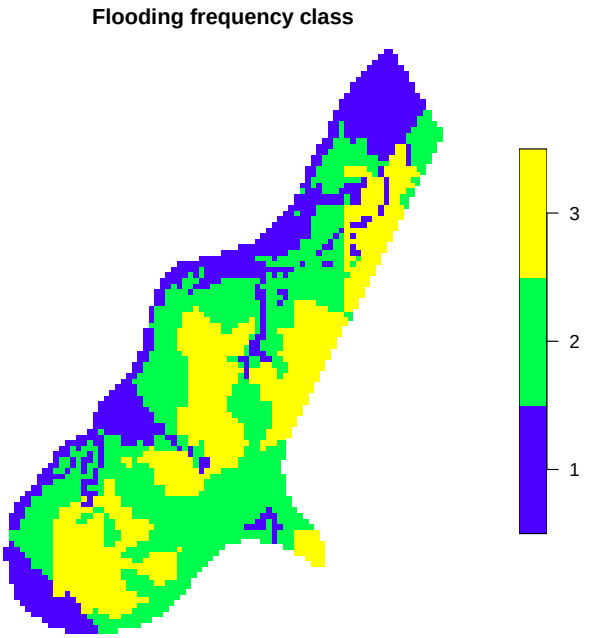
In §7 we saw that the feature-space attribute "flooding frequency" explained about 25% of the variation in $\log_{10}$Zn concentration. Yet, we ignored this information when predicting by Ordinary Kriging (§11.2). In this section we examine a method to combine the two.

13.1 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).

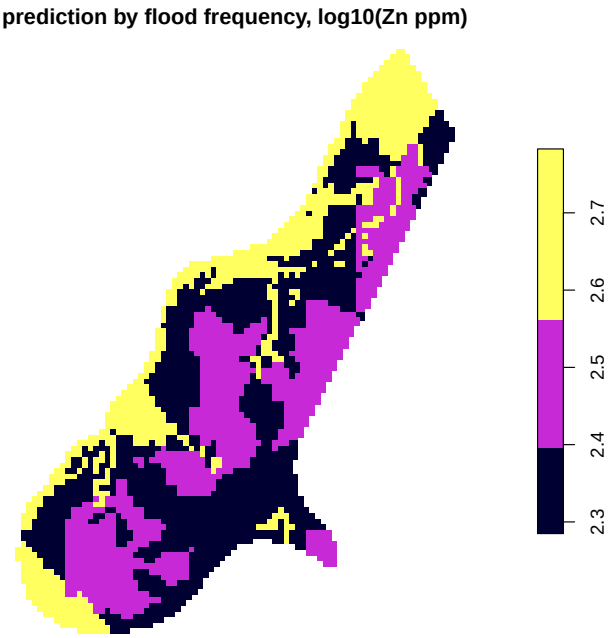
```
summary(meuse.grid.sf$ffreq)

   1    2    3
 779 1335  989

plot(meuse.grid.sf["ffreq"],
     pal = topo.colors(3), pch = 15,
     main="Flooding frequency class")
```



TASK 76 : Predict the metal concentration by flood-frequency class. •

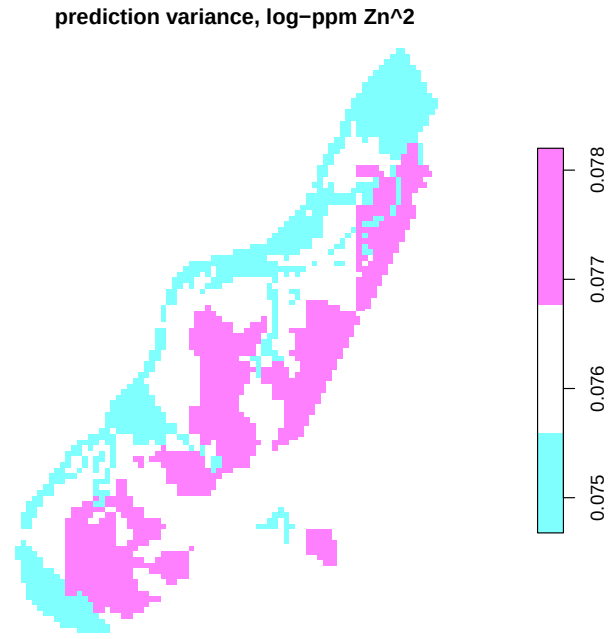
The `krige` method with the `model` argument set to `NULL` predicts without a model of spatial dependence, i.e., just from the feature-space model (here, metal predicted by flood-frequency class). The `krige` method computes the OLS regression exactly as does the `lm` function, and then uses that regression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

```
k.ffreq <- krige(logZn ~ ffreq, locations=meuse.sf,
                 newdata=meuse.grid.sf, model=NULL)

[ordinary or weighted least squares prediction]

plot(k.ffreq["var1.pred"],
     pal = sp::bpy.colors, pch = 15,
     main="prediction by flood frequency, log10(Zn ppm)")
```



And like any linear model, this also has a prediction variance. As in §11.2 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`:

```
plot(k.ffreq["var1.var"],
     pal = cm.colors, pch = 15,
     main="prediction variance, log-ppm Zn^2")
```



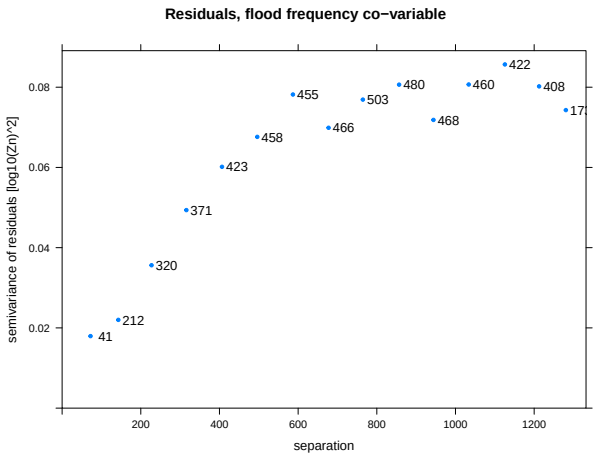
Q67 : *Explain the spatial pattern of this prediction and its variance.* *Jump to A67* •

13.2 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 77 : Compute and display the empirical residual variogram of $\log_{10}$Zn, after accounting for flooding frequency. •

```
(vr <- variogram(logZn ~ ffreq, location=meuse.sf,
                 cutoff=1300, width=90))

    np       dist      gamma dir.hor dir.ver   id
1   41   72.24836 0.01794898       0       0 var1
2  212  142.88031 0.02199982       0       0 var1
3  320  227.32202 0.03561628       0       0 var1
4  371  315.85549 0.04936334       0       0 var1
5  423  406.44801 0.06015197       0       0 var1
6  458  496.09401 0.06761754       0       0 var1
7  455  586.78634 0.07817727       0       0 var1
8  466  677.39566 0.06988742       0       0 var1
9  503  764.55712 0.07690598       0       0 var1
10 480  856.69422 0.08063470       0       0 var1
11 468  944.02864 0.07184257       0       0 var1
12 460 1033.62277 0.08067768       0       0 var1
13 422 1125.63214 0.08567598       0       0 var1
14 408 1212.62350 0.08020714       0       0 var1
15 173 1280.65364 0.07429483       0       0 var1

print(plot(vr, plot.numbers=T, pch = 20,
           xlab = "separation", ylab = "semivariance of residuals [log10(Zn)^2]",
           main="Residuals, flood frequency co-variable"))
```



Q68 : *How does this empirical variogram compare to the original (non-residual) empirical variogram computed in § 10.1.2?* *Jump to A68*

•

Clearly, accounting for the flood frequency has **reduced the total variability** (as expected from the results of the linear modelling), but it has also **shortened the range of spatial dependence**. Some of the appar-

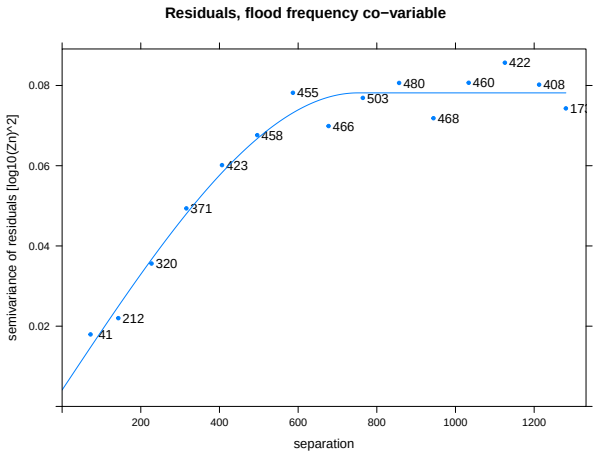
ent range in the original variogram was due to the spatial extent of the flooding classes; this has now been removed.

TASK 78 : 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=800, nugget=0.01)))

  model       psill    range
1   Nug 0.004094969   0.0000
2   Sph 0.074061465 752.1326

print(vmf)

  model      psill    range
1   Nug 0.01004123   0.0000
2   Sph 0.11525698 967.2634
```

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, pch = 20, model=vrmf,
           xlab = "separation", ylab = "semivariance of residuals [log10(Zn)^2]",
           main="Residuals, flood frequency co-variable"))
```



Q69 : *How does this variogram model compare to the original (non-residual) variogram model?* *Jump to A69*

•

The residual variogram 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.

13.3 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 79 : 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.

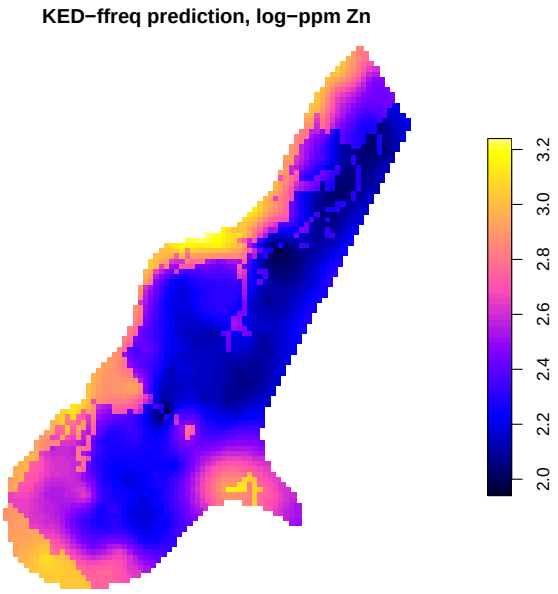
! → In KED, the formula for kriging *must* match that for the residual variogram.

```
kr40 <- krige(logZn ~ ffreq, locations=meuse.sf,
              newdata=meuse.grid.sf, 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 by `gstat`; we prefer to call it KED.

TASK 80 : Display the map of KED predicted values. •

```
plot(kr40["var1.pred"],
     nbreaks = 64, pch = 15,
     pal = sp::bpy.colors,
     main="KED-ffreq prediction, log-ppm Zn")
```



Q70 : *How does this KED map compare to the OK map? Where is the effect of flood frequency class reflected in the prediction?* *Jump to A70*

•

13.3.1 Displaying several maps on the same scale

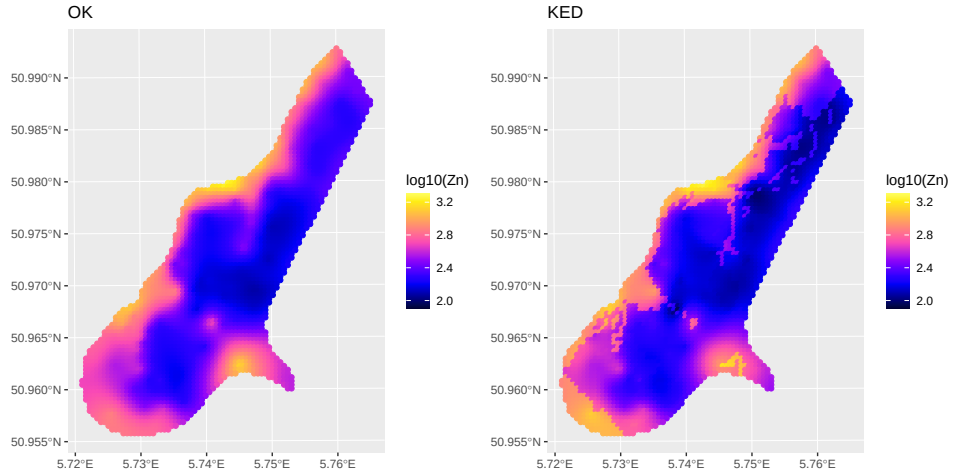
To get a better idea of the differences in the predictions, we'd like to show the two maps side-by-side.

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 break-points for the colour ramp.

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

[1] 3.237329

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

[1] 1.956585

(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
```

Now plot them with `ggplot2` graphics. We specify a `scale_color_gradientn` colour ramp with a defined number of colours and specific breakpoints, so the two plots have the same visualization for the same predicted value.

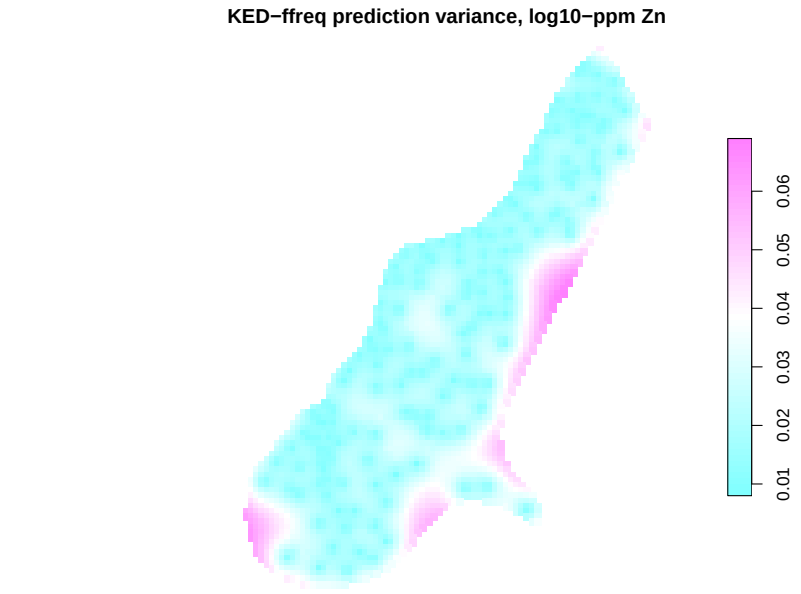
```
g.ok <- ggplot() +
    geom_sf(data = k40, aes(col = var1.pred)) +
    labs(title = "OK", col = "log10(Zn)") +
    scale_color_gradientn(limits = range(ramp),
                          colors = sp::bpy.colors(length(ramp)))
g.ked <-ggplot() +
    geom_sf(data = kr40, aes(col = var1.pred))  +
    labs(title = "KED", col = "log10(Zn)") +
    scale_color_gradientn(limits = range(ramp),
                          colors = sp::bpy.colors(length(ramp)))
gridExtra::grid.arrange(g.ok, g.ked, nrow=1)
```



13.3.2 KED Prediction variances

TASK 81 : Display the map of the KED prediction variances. •

```
plot(kr40["var1.var"],
     nbreaks = 64, pch = 15,
     pal = cm.colors,
     main="KED-ffreq prediction variance, log10-ppm Zn")
```



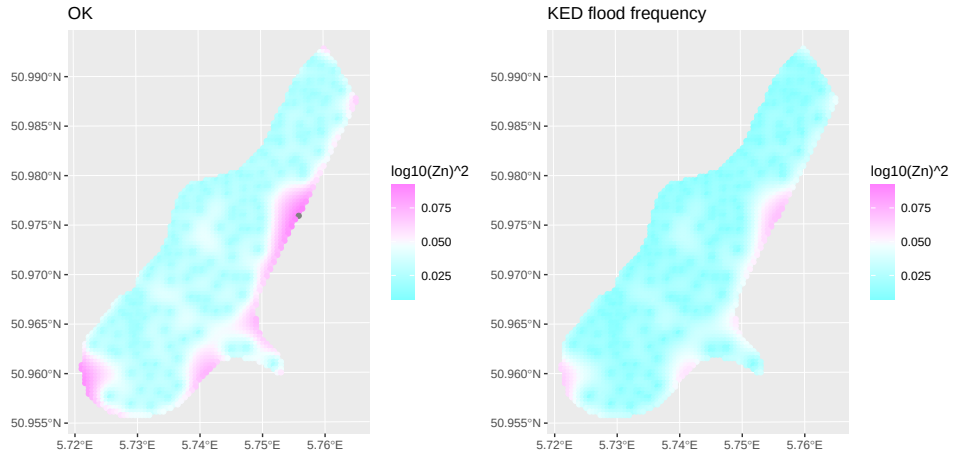
TASK 82 : Compare these prediction variances to those for OK, both numerically and graphically. •

```
summary(kr40$var1.var)

    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.008075 0.016284 0.020451 0.023638 0.028071 0.068614

summary(k40$var1.var)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.01662 0.02596 0.03050 0.03471 0.03943 0.09231
```

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 0.057
[12] 0.062 0.067 0.072 0.077 0.082 0.087 0.092
```

```
g.ok <- ggplot() +
    geom_sf(data = k40, aes(col = var1.var)) +
    labs(title = "OK", col = "log10(Zn)^2") +
    scale_color_gradientn(limits = range(ramp),
                          colors = cm.colors(length(ramp)))
g.ked <-ggplot() +
    geom_sf(data = kr40, aes(col = var1.var))  +
    labs(title = "KED flood frequency", col = "log10(Zn)^2") +
    scale_color_gradientn(limits = range(ramp),
                          colors = cm.colors(length(ramp)))
gridExtra::grid.arrange(g.ok, g.ked, nrow=1)
```



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[28]. This helps especially at locations far from observation points.

13.4 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 83 : 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.sf)

[1] "part.a"   "part.b"   "dist"     "soil"     "ffreq"    "geometry"
[7] "elev"     "rt.pred"  "rf.pred"

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

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

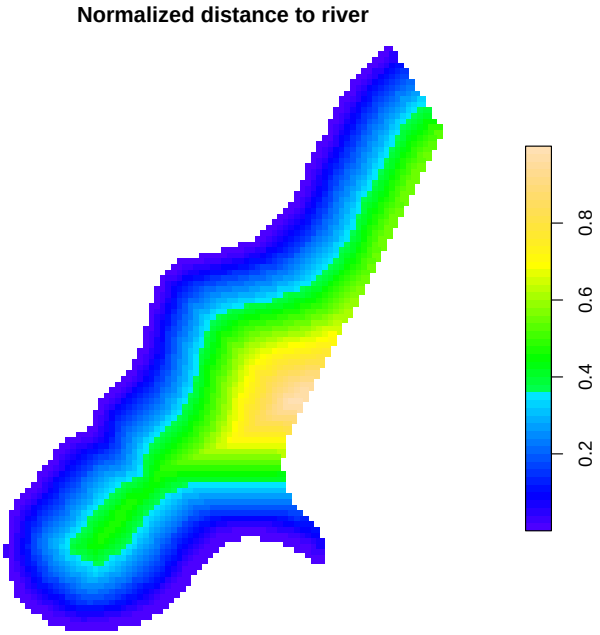
Q71 : *What covariables are available?* *Jump to A71* •

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.

[28] 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 84 : Display the normalized distance to river. •

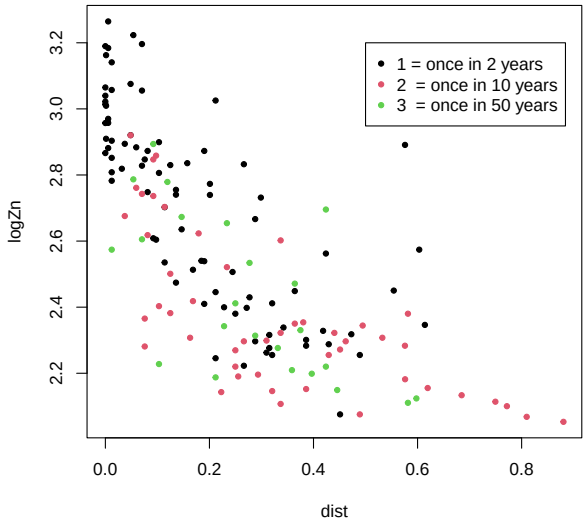
```
plot(meuse.grid.sf["dist"],
     pal = topo.colors, pch = 15, nbreaks = 64,
     main = "Normalized distance to river")
```



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 85 : Display a feature-space scatterplot of metal concentration vs. distance from river, with the observations coloured by flood frequency class. •

```
plot(logZn ~ dist, data=meuse.sf,
     col=meuse.sf$ffreq, pch = 20)
legend(x=0.5, y=3.2,
       legend=c("1 = once in 2 years",
                "2  = once in 10 years","3  = once in 50 years"),
       pch=20, col=1:3)
```



Q72 : *Does there appear to be a linear relation between distance and metal concentration? How strong does it appear to be?* *Jump to A72* •

TASK 86 : 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 §13.1 for the classified predictor (flood frequency class).

```
m.lzn.dist <- lm(logZn ~ dist, data=meuse.sf)
summary(m.lzn.dist)


Call:
lm(formula = logZn ~ dist, data = meuse.sf)

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)  2.83759    0.02680  105.87   <2e-16 ***
dist        -1.17256    0.08631  -13.59   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2118 on 153 degrees of freedom
Multiple R-squared:  0.5468,Adjusted R-squared:  0.5438
F-statistic: 184.6 on 1 and 153 DF,  p-value: < 2.2e-16
```

Q73 : *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 A73* •

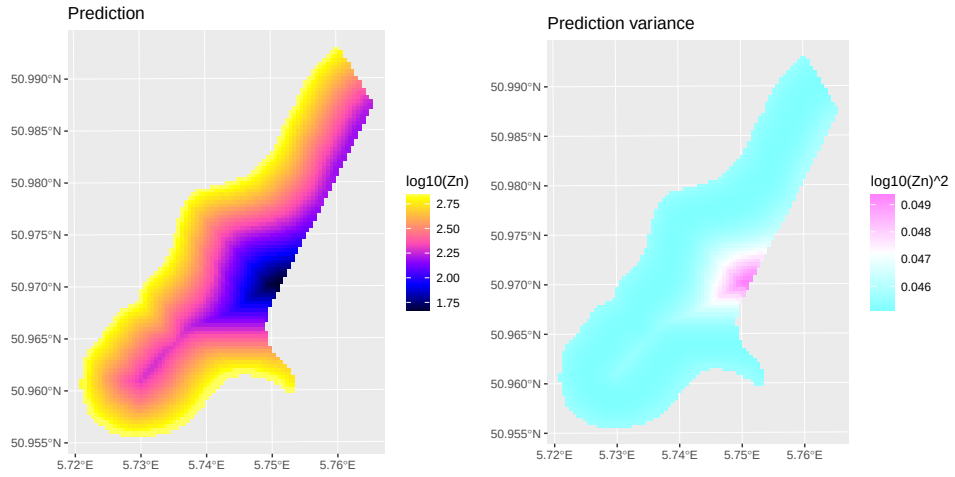
TASK 87 : Predict the metal concentration over the study area, from the distance to river. •

As in §13.1 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.sf,
                newdata=meuse.grid.sf, model=NULL)

[ordinary or weighted least squares prediction]
```

TASK 88 : Display the interpolated distance to river and its prediction variance across the study area. •

```
g.idw <- ggplot() +
    geom_sf(data = k.dist, aes(col = var1.pred), pch = 15) +
    labs(title = "Prediction", col = "log10(Zn)") +
    scale_color_gradientn(colors = sp::bpy.colors(64))
g.idw.v <-ggplot() +
    geom_sf(data = k.dist, aes(col = var1.var), pch = 15)  +
    labs(title = "Prediction variance", col = "log10(Zn)^2") +
    scale_color_gradientn(colors = cm.colors(64))
gridExtra::grid.arrange(g.idw, g.idw.v, nrow=1)
```



Q74 : *Explain the spatial pattern of this prediction and its variance.* *Jump to A74* •

TASK 89 : 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 vari-

ance, 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.sf)
m.lzn.ff.dist.i <- lm(logZn ~ ffreq * dist, data=meuse.sf)
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  5.3452
2    151  5.7919 -2   -0.4467  6.2261   0.00253 **
3    153  6.8605 -2   -1.0686 14.8945 1.268e-06 ***
4    152 11.3057  1   -4.4452
---
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.

Q75 : *Do the two-predictor models give significantly lower residual sum-of-squares?* *Jump to A75* •

Q76 : *Does the interaction model give significantly lower residual sum-of-squares than the additive model?* *Jump to A76*
•

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 90 : 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 -37.36411
m.lzn.ff         4  42.06207
m.lzn.ff.dist    5 -59.60963
m.lzn.ff.dist.i  7 -68.05030
```

Q77 : *Which model has the lowest AIC? Based on this and the ANOVA, which model gives the best feature-space prediction of metal concentra-*

tion? What does this imply about the process? *Jump to A77*
•

TASK 91 : Display the model summary for the best model. •

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


Call:
lm(formula = logZn ~ ffreq * dist, data = meuse.sf)

Residuals:
     Min       1Q   Median       3Q      Max
-0.40992 -0.13492 -0.00252  0.10804  0.72421

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.93981    0.03048  96.435  < 2e-16 ***
ffreq2      -0.33285    0.05729  -5.810 3.64e-08 ***
ffreq3      -0.24230    0.08486  -2.855 0.004913 **
dist        -1.34236    0.12531 -10.712  < 2e-16 ***
ffreq2:dist  0.61585    0.17471   3.525 0.000563 ***
ffreq3:dist  0.35963    0.27724   1.297 0.196571
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1894 on 149 degrees of freedom
Multiple R-squared:  0.6469,Adjusted R-squared:  0.635
F-statistic: 54.59 on 5 and 149 DF,  p-value: < 2.2e-16
```

Q78 : *How much of the variability in metal concentration is explained by this model?* *Jump to A78* •

13.4.1 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.

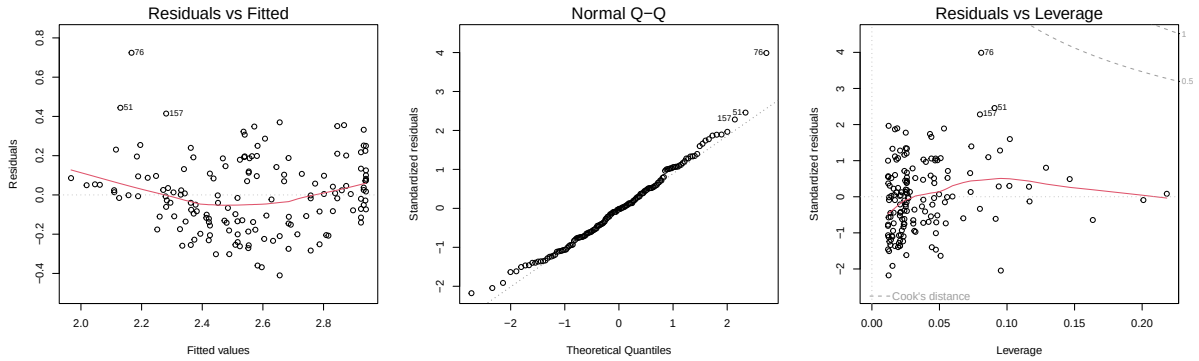
As explained in §7.2.2 linear model must satisfy several assumptions [7, 11], among which are:

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 92 : 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))
```



Q79 : *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 A79* •

Q80 : *Looking at the "Normal Q-Q" plot, do the residuals appear to be normally distributed?* *Jump to A80* •

Q81 : *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 A81*
•

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.

> **Note:** Recall from §7.2.2: 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.

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.sf) == "76"))

[1] 69

meuse.sf[ix,]

Simple feature collection with 1 feature and 15 fields
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: 179852 ymin: 330801 xmax: 179852 ymax: 330801
Projected CRS: Amersfoort / RD New
   cadmium copper lead zinc elev     dist  om ffreq soil lime landuse
76     3.4     55  325  778 6.32 0.575877 6.9     1    1    0      Bw
   dist.m   logZn    logCu              geometry  zn.i
76    750 2.89098 1.740363 POINT (179852 330801) FALSE
```

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

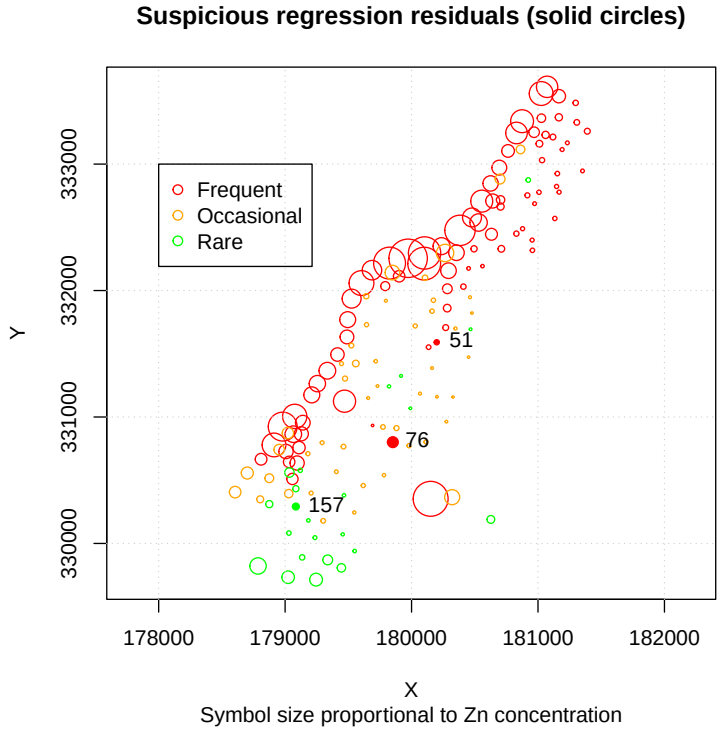
TASK 93 : Plot the suspicious regression residuals in geographic space. •

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.

```
# which row numbers correspond to the observations witH large residuals?
(bad.pt <- which(row.names(meuse.sf) %in% c("76","51","157")))

[1]  50  69 152

# where are they?
st_coordinates(meuse.sf)[bad.pt, ]

         X      Y
50  180199 331591
69  179852 330801
152 179085 330292

# make a logical vector of all rows, whether they have large residuals
#  or not
is.row.bad <- (row(meuse.sf)[,1] %in% bad.pt)
```

```
colours.ffreq = c("red","orange","green")
plot(st_coordinates(meuse.sf), asp=1,
     col=colours.ffreq[meuse.sf$ffreq],
     # select print character, large residual or not?
     # 20 = filled circle; 1 = open circle
     pch=ifelse(is.row.bad, 20, 1),
     # symbol size proportional to Zn concentration
     cex=4*meuse.sf$zinc/max(meuse.sf$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(st_coordinates(meuse.sf)[bad.pt,],
    c("51","76","157"), 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, is.row.bad)
```

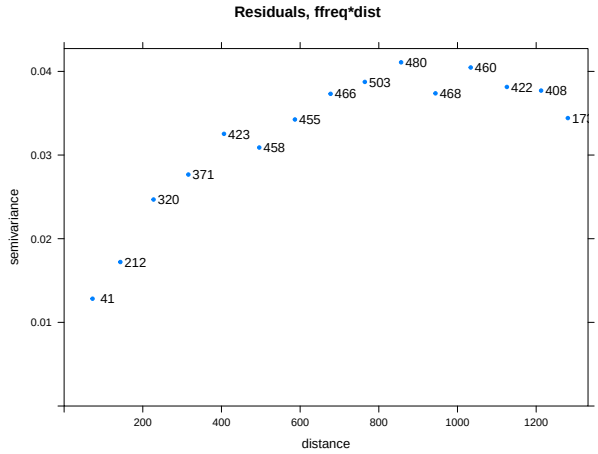
13.4.2 Spatial structure of the the residuals

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 94 : 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. •

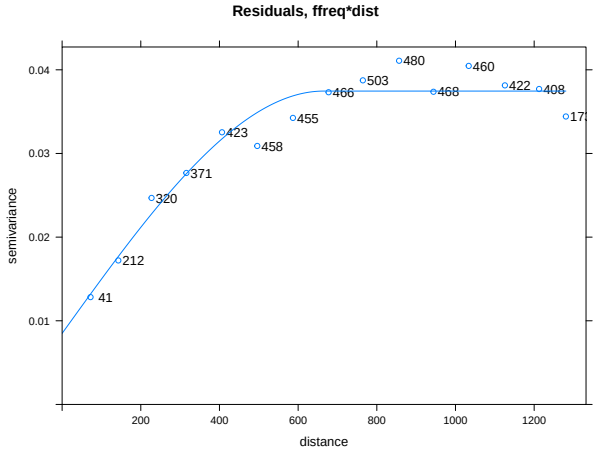
```
(vr2 <- variogram(logZn ~ ffreq*dist, location=meuse.sf,
                  cutoff=1300, width=90))

    np       dist      gamma dir.hor dir.ver   id
```

```
1   41   72.24836 0.01282962       0       0 var1
2  212  142.88031 0.01721119       0       0 var1
3  320  227.32202 0.02468288       0       0 var1
4  371  315.85549 0.02766886       0       0 var1
5  423  406.44801 0.03253304       0       0 var1
6  458  496.09401 0.03089852       0       0 var1
7  455  586.78634 0.03425093       0       0 var1
8  466  677.39566 0.03731832       0       0 var1
9  503  764.55712 0.03874194       0       0 var1
10 480  856.69422 0.04108453       0       0 var1
11 468  944.02864 0.03737466       0       0 var1
12 460 1033.62277 0.04046701       0       0 var1
13 422 1125.63214 0.03813938       0       0 var1
14 408 1212.62350 0.03770635       0       0 var1
15 173 1280.65364 0.03442058       0       0 var1

print(plot(vr2, plot.numbers=T, pch = 20,
           main="Residuals, ffreq*dist"))
```




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

  model       psill    range
1   Nug 0.008492801   0.0000
2   Sph 0.028964991 664.7822

print(vrmf)

  model       psill    range
1   Nug 0.004094969   0.0000
2   Sph 0.074061465 752.1326

print(vmf)

  model      psill    range
1   Nug 0.01004123   0.0000
2   Sph 0.11525698 967.2634
```

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



Q82 : *What happens to the values of the ranges and partial (structural) sills as the model includes more predictors?* *Jump to A82* •

13.4.3 KED prediction

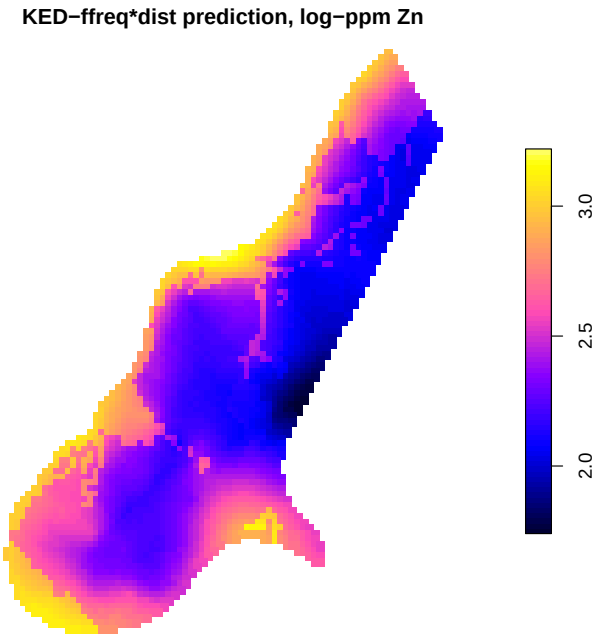
TASK 95 : Predict by KED using the best feature-space model as covariables. •

```
kr240 <- krige(logZn ~ ffreq*dist, locations=meuse.sf,
               newdata=meuse.grid.sf, 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 96 : Display the map of predicted values. •

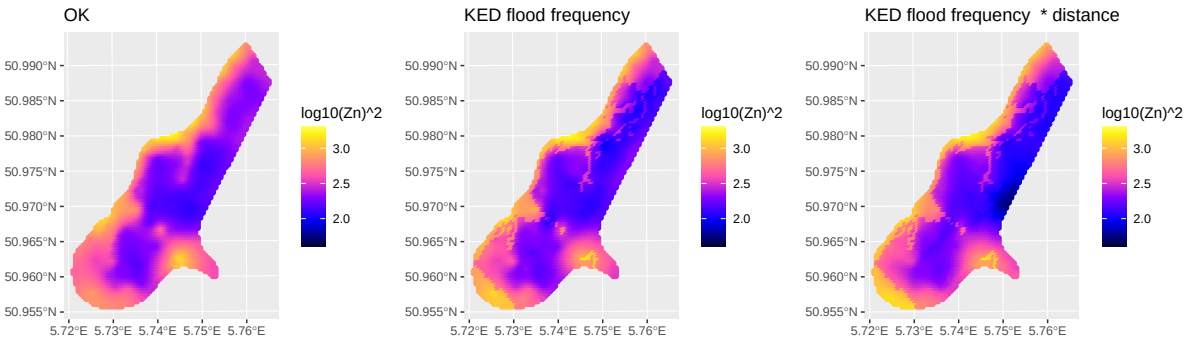
```
plot(kr240["var1.pred"], pch = 15, nbreaks = 64,
             pal = sp::bpy.colors,
             main="KED-ffreq*dist prediction, log-ppm Zn")
```



TASK 97 : 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)
g.ok <- ggplot() +
    geom_sf(data = k40, aes(col = var1.pred)) +
    labs(title = "OK", col = "log10(Zn)^2") +
    scale_color_gradientn(limits = range(ramp),
                          colors = sp::bpy.colors(length(ramp)))
g.ked <-ggplot() +
    geom_sf(data = kr40, aes(col = var1.pred))  +
    labs(title = "KED flood frequency", col = "log10(Zn)^2") +
    scale_color_gradientn(limits = range(ramp),
                          colors = sp::bpy.colors(length(ramp)))
g.ked.2 <-ggplot() +
    geom_sf(data = kr240, aes(col = var1.pred))  +
    labs(title = "KED flood frequency  * distance", col = "log10(Zn)^2") +
    scale_color_gradientn(limits = range(ramp),
                          colors = sp::bpy.colors(length(ramp)))
gridExtra::grid.arrange(g.ok, g.ked, g.ked.2, nrow=1)
```



Q83 : *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 A83* •

13.4.4 KED prediction variances

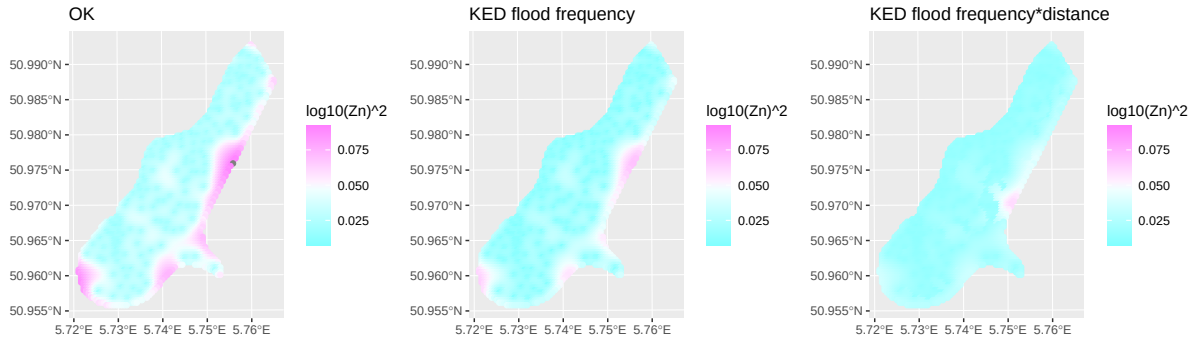
TASK 98 : Compare these prediction variances to those for OK, both numerically and graphically. •

```
summary(kr240$var1.var)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.01213 0.01602 0.01817 0.02050 0.02309 0.05985

summary(kr40$var1.var)

    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.008075 0.016284 0.020451 0.023638 0.028071 0.068614
```

```
summary(k40$var1.var)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.01662 0.02596 0.03050 0.03471 0.03943 0.09231
```

```
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)
```

```
g.ok <- ggplot() +
    geom_sf(data = k40, aes(col = var1.var)) +
    labs(title = "OK", col = "log10(Zn)^2") +
    scale_color_gradientn(limits = range(ramp),
                          colors = cm.colors(length(ramp)))
g.ked <-ggplot() +
    geom_sf(data = kr40, aes(col = var1.var))  +
    labs(title = "KED flood frequency", col = "log10(Zn)^2") +
    scale_color_gradientn(limits = range(ramp),
                          colors = cm.colors(length(ramp)))
g.ked.2 <-ggplot() +
    geom_sf(data = kr240, aes(col = var1.var))  +
    labs(title = "KED flood frequency*distance", col = "log10(Zn)^2") +
    scale_color_gradientn(limits = range(ramp),
                          colors = cm.colors(length(ramp)))
gridExtra::grid.arrange(g.ok, g.ked, g.ked.2, nrow=1)
```



Q84 : *What is the effect of adding the distance to river as a predictor on the spatial pattern of the prediction variances?* *Jump to A84* •

Challenge: Repeat the process this section for one of the other metals (Cu, Pb, or Cd). What differences, if any, do you expect in the linear models and in the fitted residual variogram model? Why? Do you expect a similar spatial pattern to that for Zn? Why or why not? Compare the fitted variogram models and the KED predictions and their variances. Comment on the similarities and differences, and try to explain the reasons for these.

14 * Generalized Least Squares

The mixed prediction by Kriging with External Drift (KED) of §13.3, although convenient, is not mathematically-correct. This is because the linear model residuals are based on the Ordinary Least Squares (OLS) solution of the linear (feature-space) model, but we've seen in §13.2 that these residuals are spatially-correlated. Thus the linear model parameters are not optimal, and so neither is the mixed predictor.

To solve the problem of spatially-correlated OLS residuals, we turn to generalized least squares (GLS), which solves both the feature-space and residual spatial correlation in one step.

14.1 * GLS – theory

The key difference between OLS and GLS is that in the linear model fit by OLS, the residuals ε are assumed to be *independently* and *identically* distributed with the same variance σ^2:

$$\mathbf{y} = \mathbf{X}\beta + \varepsilon,\ \varepsilon \sim \mathcal{N}(0, \sigma^2\mathbf{I}) \tag{17}$$

Whereas, now the residuals are themselves considered to be a random variable η that has a covariance structure:

$$\mathbf{y} = \mathbf{X}\beta + \eta,\ \eta \sim \mathcal{N}(0, \mathbf{V}) \tag{18}$$

where $\mathbf{V}$ is a positive-definite variance-covariance matrix of the model residuals. The covariances in this matrix (off-diagonals) are typically based on the distance between observations, using some model of spatial correlation.

Lark & Cullis [19, Appendix] point out that the error vectors can now not be assumed to be spherically distributed in feature space around the 0 expected value, but rather that error vectors in some directions are longer than in others. So, the measure of distance (the vector norm) is now a so-called "generalized" distance[29], taking into account the covariance between error vectors:

$$S = (\mathbf{y} - \mathbf{X}\beta)^T\mathbf{V}^{-1}(\mathbf{y} - \mathbf{X}\beta) \tag{19}$$

The OLS equivalent is simpler:

$$S = (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) \tag{20}$$

Comparing these equations, we see that the GLS formulation of Equation 19 includes the variance-covariance matrix of the residuals $\mathbf{V} = \sigma^2 C$, where σ^2 is the variance of the residuals and $\mathbf{C}$ is the correlation matrix. This reduces to the OLS formulation of Equation 20 if there is no covariance, i.e., $\mathbf{V} = \mathbf{I}$.

[29] This is closely related to the Mahalanobis distance

Expanding Equation 19, taking the partial derivative with respect to the parameters, setting equal to zero and solving we obtain:

$$\begin{aligned} \frac{\partial}{\partial \beta} S &= -2\mathbf{X}^T\mathbf{V}^{-1}\mathbf{y} + 2\mathbf{X}^T\mathbf{V}^{-1}\mathbf{X}\beta \\ 0 &= -\mathbf{X}^T\mathbf{V}^{-1}\mathbf{y} + \mathbf{X}^T\mathbf{V}^{-1}\mathbf{X}\beta \\ \hat{\beta}_{\text{GLS}} &= (\mathbf{X}^T\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{V}^{-1}\mathbf{y} \end{aligned} \tag{21}$$

This reduces to the OLS estimate $\hat{\beta}_{\text{OLS}}$ if there is no covariance, i.e., $\mathbf{V} = \mathbf{I}$.

In the case of spatial correlation, we ensure positive-definiteness (i.e., always a real-valued solution) by using an **authorized covariance function** C and assuming that the entries are completely determined by the vector **distance** between points $\mathbf{x}_i - \mathbf{x}_j$:

$$\mathbf{C}_{i,j} = C(\mathbf{x}_i - \mathbf{x}_j) \tag{22}$$

In this formulation C has a three-parameter vector θ, as does the corresponding variogram model: the range a, the total sill σ^2, and the proportion of total sill due to pure error, not spatial correlation s[30].

fixed vs. random effects

In modelling terminology, the coefficients β are called **fixed** effects, because their effect on the response variable is fixed once the parameters are known. By contrast the covariance parameters η are called **random** effects, because their effect on the response variable is stochastic, depending on a random variable with these parameters.

mixed models

Models with the form of Equation 18 are called **mixed** models: some effects are fixed (here, the relation between the predictand $\log_{10}$Zn and the predictors distance to river and flood frequency) and others are random (here, the error variances) but follow a known structure; these models have many applications and are extensively discussed in Pinheiro & Bates [23]. Here the random effect η represents both the spatial structure of the residuals from the fixed-effects model, and the unexplainable (short-range) noise. This latter corresponds to the noise σ^2 of the linear model of Equation 17.

To solve Equation 21 we first need to compute $\mathbf{V}$, i.e., estimate the variance parameters $\theta = [\sigma^2, s, a]$, use these to compute $\mathbf{C}$ with equation 22 and from this $\mathbf{V}$, after which we can use equation 21 to estimate the fixed effects β. But θ is estimated from the residuals of the fixed-effects regression, which has not yet been computed. How can this "chicken-and-egg"[31] computation be solved?

REML

The answer is to use **residual** (sometimes called "restricted") **maximum likelihood** (REML) to maximize the likelihood of the *random* effects θ independently of the *fixed* effects β.

[30] In variogram terms, this is the nugget variance c_0 as a proportion of the total sill $(c_0 + c_1)$.

[31] from the question "which came first, the chicken or the egg?"

Lark & Cullis [19, Eq. 12] show that the likelihood of the parameters in Equation 17 can be expanded to include the spatial dependence implicit in the variance-covariance matrix $\mathbf{V}$, rather than a single residual variance σ^2. The log-likelihood is then:

$$\ell(\beta, \theta|\mathbf{y}) = c - \frac{1}{2}\log|\mathbf{V}| - \frac{1}{2}(\mathbf{y} - \mathbf{X}\beta)^T\mathbf{V}^{-1}(\mathbf{y} - \mathbf{X}\beta) \tag{23}$$

where c is a constant (and so does not vary with the parameters) and $\mathbf{V}$ is built from the variance parameters θ and the distances between the observations. By assuming **second-order stationarity**[32], the structure can be summarized by the covariance parameters $\theta = [\sigma^2, s, a]$, i.e., the total sill, nugget proportion, and range.

However, maximizing this likelihood for the random-effects covariance parameters θ also requires maximizing in terms of the fixed-effects regression parameters β, which in this context are called *nuisance parameters* since at this point we don't care about their values; we will compute them after determining the covariance structure.

Both the covariance and the nuisance parameters β must be estimated, it seems at the same time ("chicken and egg" problem) but in fact the technique of REML can be used to first estimate θ without having to know the nuisance parameters. Then we can use these to compute $\mathbf{C}$ with equation 22 and from this $\mathbf{V}$, after which we can use equation 21 to estimate the fixed effects β.

The maximum likelihood estimate of θ is thus called "restricted", because it only estimates the covariance parameters (random effects). Conceptually, REML estimation of the covariance parameters θ is ML estimation of both these and the nuisance parameters β, with the later integrated out [23, §2.2.5]:

$$\ell(\theta|\mathbf{y}) = \int \ell(\beta, \theta|\mathbf{y})\, d\beta \tag{24}$$

Pinheiro & Bates [23, §2.2.5] show how this is achieved, given a likelihood function, by a change of variable to a statistic sufficient for β.

14.2 GLS – practice

The computations are performed with the `gls` function of the `nlme` 'Nonlinear mixed effects models' package [1].

Task 99 : Set up and solve a GLS model, using the covariance structure estimated from the variogram of the OLS residuals from the best mixed linear model of §13.4. •

Setting up a GLS model

The linear model formulation is the same as for `lm`. However:

[32] that is, the covariance structure is the same over the entire field, and only depends on the distance between pairs of points

- It has an additional argument `correlation`, which specifies the correlation structure.
- This is built with various correlation models; we use `corSpher` for spherical spatial correlation, which is what we fit for the OLS residuals in §13.4.2.
  - The `form` names the spatial dimensions, here 2D with the two coördinates `x` and `y`.
  - The `value` argument to initialize the search for the correlation structure parameters; this a list of the initial values of the **range** and the **proportional nugget**, i.e., the proportion of the total sill represented by the nugget.

These initial parameter are:

```
vrm2f[2,"range"]

[1] 664.7822

(prop.nugget <- vrm2f[1,"psill"]/sum(vrm2f[,"psill"]))

[1] 0.2267299
```

> **Note:** For a list of the predefined model forms see `?corClasses`. Users can also define their own `corStruct` classes.

Unfortunately `nlme` does not understand `sp` structures, so the coördinates `x` and `y` must be brought back into a dataframe for use in the `form` argument; the `as.data.frame` method changes the class..

```
library(nlme)
m.gls <- gls(model=logZn ~ ffreq * dist,
             data=meuse,
             correlation=corSpher(
                 form=~x + y,
                 nugget=TRUE,
                 value=c(vrm2f[2,"range"], prop.nugget),
                 ))
```

The `gls` function is not guaranteed to find a valid correlation structure. First, there may be no spatial correlation of the residuals. Second, we may have specified an inappropriate model form. Third, if the starting values are not close to good fits, the optimization method may not find the correct fit. Therefore it is crucial to check the results of the model fitting to see if they are reasonable.

Task 100 : Display the model summary. •

```
summary(m.gls)

Generalized least squares fit by REML
  Model: logZn ~ ffreq * dist
  Data: meuse
        AIC       BIC   logLik
  -112.3879 -85.35235 65.19394

Correlation Structure: Spherical spatial correlation
 Formula: ~x + y
```

```
 Parameter estimate(s):
      range      nugget
1163.141208    0.109821

Coefficients:
                Value  Std.Error  t-value p-value
(Intercept)  2.9805919 0.08740310 34.10167  0.0000
ffreq2      -0.2575186 0.04446084 -5.79203  0.0000
ffreq3      -0.2525222 0.10315548 -2.44798  0.0155
dist        -1.0373138 0.21693848 -4.78160  0.0000
ffreq2:dist  0.0175028 0.15400959  0.11365  0.9097
ffreq3:dist -0.0130419 0.29324057 -0.04447  0.9646

 Correlation:
            (Intr) ffreq2 ffreq3 dist   ffrq2:
ffreq2      -0.118
ffreq3      -0.291  0.221
dist        -0.480  0.187  0.240
ffreq2:dist  0.109 -0.743 -0.240 -0.354
ffreq3:dist  0.249 -0.222 -0.880 -0.308  0.427

Standardized residuals:
       Min         Q1        Med         Q3        Max
-2.0317652 -0.8598677 -0.1926644  0.3522743  2.0018849

Residual standard error: 0.2536374
Degrees of freedom: 155 total; 149 residual
```

This shows the fitted coefficients and their standard errors, as in the summary for `lm`. It also shows the residuals, their standard error, and the correlation parameters.

TASK 101 : Compare the GLS estimated coefficients with the OLS estimates: •

```
coef(m.gls); coef(m.lzn.ff.dist.i)

(Intercept)      ffreq2      ffreq3        dist ffreq2:dist
 2.98059194 -0.25751856 -0.25252222 -1.03731381  0.01750280
ffreq3:dist
-0.01304186
(Intercept)      ffreq2      ffreq3        dist ffreq2:dist
  2.9398053  -0.3328491  -0.2422956  -1.3423623   0.6158492
ffreq3:dist
  0.3596295

# percent change
round(100*(coefficients(m.gls)
           - coefficients(m.lzn.ff.dist.i))/
      coefficients(m.lzn.ff.dist.i),2)

(Intercept)      ffreq2      ffreq3        dist ffreq2:dist
       1.39      -22.63        4.22      -22.72      -97.16
ffreq3:dist
    -103.63
```

Q85 : *Are the coefficients different when fit by OLS and GLS? In this model, which change the most?* *Jump to A85* •

TASK 102 : Display the confidence intervals for the coefficients. •

The `intervals` function of the `nlme` package gives approximate confidence intervals of the GLS fit.

```
intervals(m.gls, level=0.95)$coef

                lower        est.       upper
(Intercept)  2.8078823  2.98059194  3.15330162
ffreq2      -0.3453738 -0.25751856 -0.16966336
ffreq3      -0.4563588 -0.25252222 -0.04868565
dist        -1.4659871 -1.03731381 -0.60864052
ffreq2:dist -0.2868222  0.01750280  0.32182777
ffreq3:dist -0.5924891 -0.01304186  0.56640537
attr(,"label")
[1] "Coefficients:"
```

These seem quite wide, indicating that the model is perhaps not sufficiently specified to capture all the reasons for variation in $\log_{10}$Zn over this area.

TASK 103 : Display the correlation structure fit by `gls`. Compare with the correlation structure estimated from the OLS residuals. •

```
round(intervals(m.gls, level=0.95)$corStruct,4)

           lower      est.    upper
range  1002.5849 1163.1412 1350.588
nugget    0.0293    0.1098    0.335
attr(,"label")
[1] "Correlation structure:"

print(paste("range:", round(vrm2f[2,"range"], 4), "m" ))

[1] "range: 664.7822 m"

print(paste("proportional nugget", round(prop.nugget, 4)))

[1] "proportional nugget 0.2267"
```

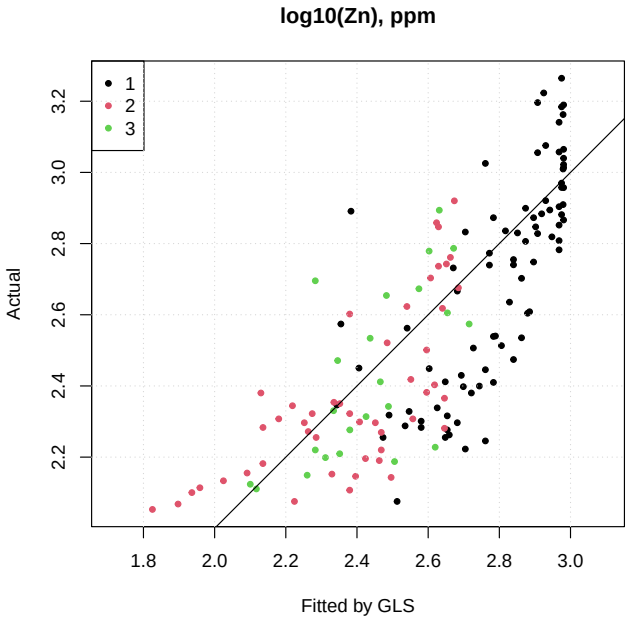
The confidence interval of the range parameter is not too wide; however the proportional nugget confidence limit ranges from near 0 to over 1/3 of the total sill.

Q86 : *Does the effective range of the exponential model fit by GLS match that fit from the exponential model fit to the OLS-derived empirical variogram?* *Jump to A86*

•

TASK 104 : Plot the actual vs. model fits on a 1:1 scatterplot. •

```
plot(meuse$logZn ~ predict(m.gls),
     col=meuse$ffreq, pch=20, asp=1,
     xlab="Fitted by GLS",
     ylab="Actual",
     main="log10(Zn), ppm")
legend("topleft", levels(meuse$ffreq), pch=20, col=1:4)
grid()
abline(0,1)
```

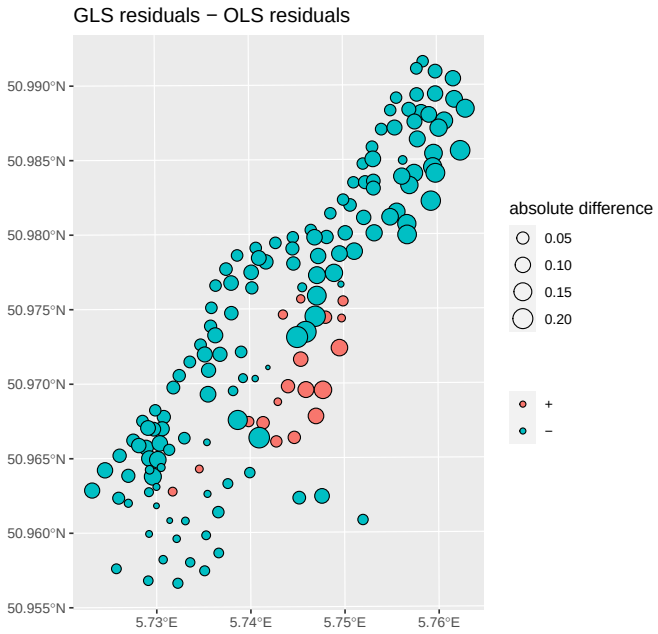


The fit clusters well around the 1:1 line (good accuracy) but is diffuse (low precision). However, the model under-predicts the highest values (> 3) and over-predicts the lowest values (< 2.2).

TASK 105 : Compute the difference between the GLS and OLS residuals, add them to the spatial points, and display as a bubble plot. •

A "bubble plot" shows (1) the absolute value of the variable by its size, and (2) its sign by its colour.

```
meuse.sf$diff.gls.ols.resid <- residuals(m.gls) - residuals(m.lzn.ff.dist.i)
summary(meuse.sf$diff.gls.ols.resid)

    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.22811 -0.09874 -0.05558 -0.05949 -0.02137  0.14210

ggplot(data = meuse.sf) +
    geom_sf(aes(size = abs(diff.gls.ols.resid),
                fill = ifelse(diff.gls.ols.resid > 0, "green", "red")),
            pch = 21) +
    labs(title = "GLS residuals - OLS residuals",
         size = "absolute difference", fill = "") +
    scale_fill_discrete(labels=c('+', '-'))
```



Q87 : *Describe the geographic pattern of the discrepencies between the GLS and OLS residuals. Explain the pattern in terms of how GLS solves for spatial correlation of residuals.* *Jump to A87* •

14.3 GLS prediction

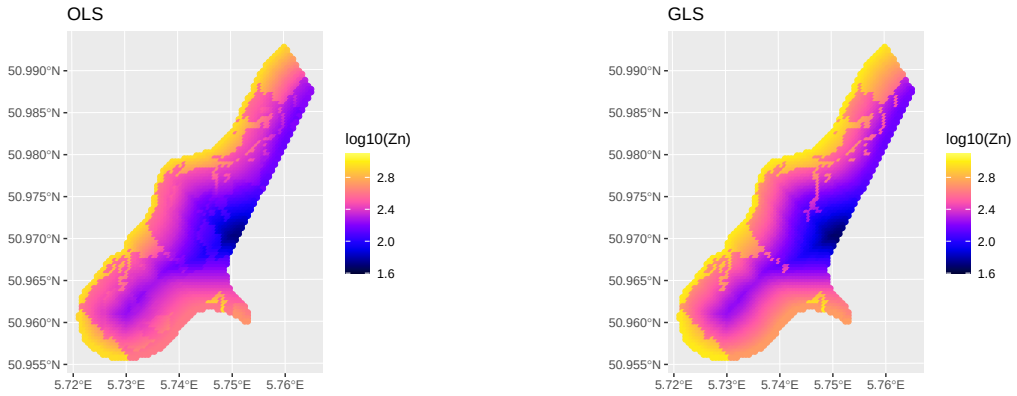
TASK 106 : Predict over the Meuse grid with the OLS and GLS models; display them side-by-side, along with their differences. •

```
meuse.grid.sf$ols.pred <- predict(m.lzn.ff.dist.i, newdata=meuse.grid.sf)
meuse.grid.sf$gls.pred <- predict(m.gls, newdata=meuse.grid.sf)
summary(meuse.grid.sf$ols.pred)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.722   2.282   2.436   2.446   2.580   2.940

summary(meuse.grid.sf$gls.pred)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.685   2.283   2.483   2.481   2.674   2.981

meuse.grid.sf$diff.ols.gls.pred <- meuse.grid.sf$ols.pred - meuse.grid.sf$gls.pred
summary(meuse.grid.sf$diff.ols.gls.pred)

      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
-0.2562961 -0.0699342 -0.0348473 -0.0357450  0.0002291  0.1420990
```

The GLS predictions are on average a bit larger.

```
zmax <- round(max(meuse.grid.sf$ols.pred,
                  meuse.grid.sf$gls.pred), 1) + 0.1
zmin <- round(min(meuse.grid.sf$ols.pred,
                  meuse.grid.sf$gls.pred), 1) - 0.1
ramp <- seq(from=zmin, to=zmax, by=.1)
```

```
g.ols <- ggplot() +
    geom_sf(data = meuse.grid.sf, aes(col = ols.pred)) +
    labs(title = "OLS", col = "log10(Zn)") +
    scale_color_gradientn(limits = range(ramp),
                          colors = sp::bpy.colors(length(ramp)))
g.gls <-ggplot() +
    geom_sf(data = meuse.grid.sf, aes(col = gls.pred))  +
    labs(title = "GLS", col = "log10(Zn)") +
    scale_color_gradientn(limits = range(ramp),
                          colors = sp::bpy.colors(length(ramp)))
gridExtra::grid.arrange(g.ols, g.gls, nrow=1)
```



Q88 : *Where are the largest discrepencies between the predictions?* *Jump to A88* •

14.4 GLS-RK

Now that we have a better fit to the linear model, we can krige the residuals to obtain a final regression kriging prediction. Note that we have a variogram structure as fit by `gls`, but we need to convert it into a form used by `krige`.

TASK 107 : Build a variogram model in `gstat` format from the correlation structure fit by `gls`. •

The correlation structure has a range and proportional nugget, but no total sill; this is the variance of the residuals of the GLS model.

We also compare this model with residual variogram from OLS.

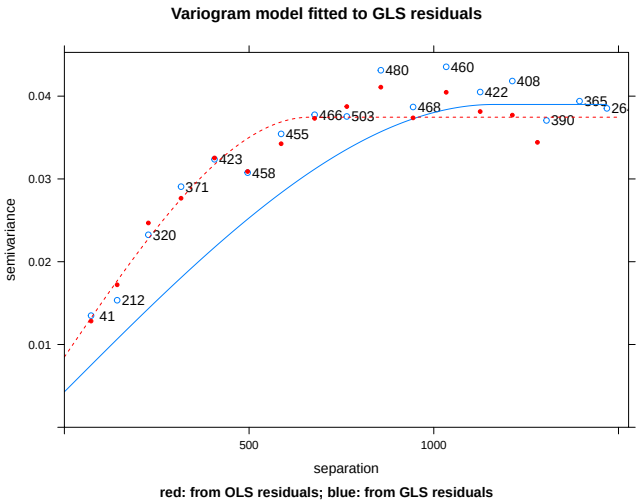
```
meuse.sf$gls.resid <- residuals(m.gls)
(p.nugget <- intervals(m.gls)$corStruct["nugget","est."])

[1] 0.109821

(t.sill <- var(meuse.sf$gls.resid))

[1] 0.03899568

(nugget <- t.sill * p.nugget)

[1] 0.004282543
```

```
(vmf.r.gls <- vgm(psill=t.sill-nugget,
                  model="Sph",
                  range=intervals(m.gls)$corStruct["range","est."],
                  nugget=nugget))

  model       psill    range
1   Nug 0.004282543    0.000
2   Sph 0.034713133 1163.141

print(vrm2f)

  model       psill    range
1   Nug 0.008492801   0.0000
2   Sph 0.028964991 664.7822
```

TASK 108 : Display the empirical variogram of the GLS residuals, with the variogram model derived from the correlation structure fit by `gls`. Also show the empirical variogram and fitted variogram model from the OLS residuals. •

> **Note:** This is some tricky code, using the concept of "panels" and functions from the `lattice` graphics package. This could be re-done in `ggplot2`, when time allows.

```
v.r.gls <- variogram(gls.resid ~ 1,
                     loc=meuse.sf, cutoff=1500, width=90)
# panel function to also show variogram and fitted model from OLS residuals
mypanel <- function(x, y, ...) {
    vgm.panel.xyplot(x, y, plot.numbers=TRUE, ...)
    panel.pointPairs(vr2$dist, vr2$gamma, col="red", pch=20)
    lattice::panel.lines(variogramLine(vrm2f, maxdist=1500), lty=2, col='red')
    }
plot(v.r.gls, pl=T, model=vmf.r.gls,
     main="Variogram model fitted to GLS residuals",
     xlab = "separation",
     sub = "red: from OLS residuals; blue: from GLS residuals",
     panel = mypanel)
```



Q89 : *Describe the differences between the variogram fits.* *Jump to A89* •

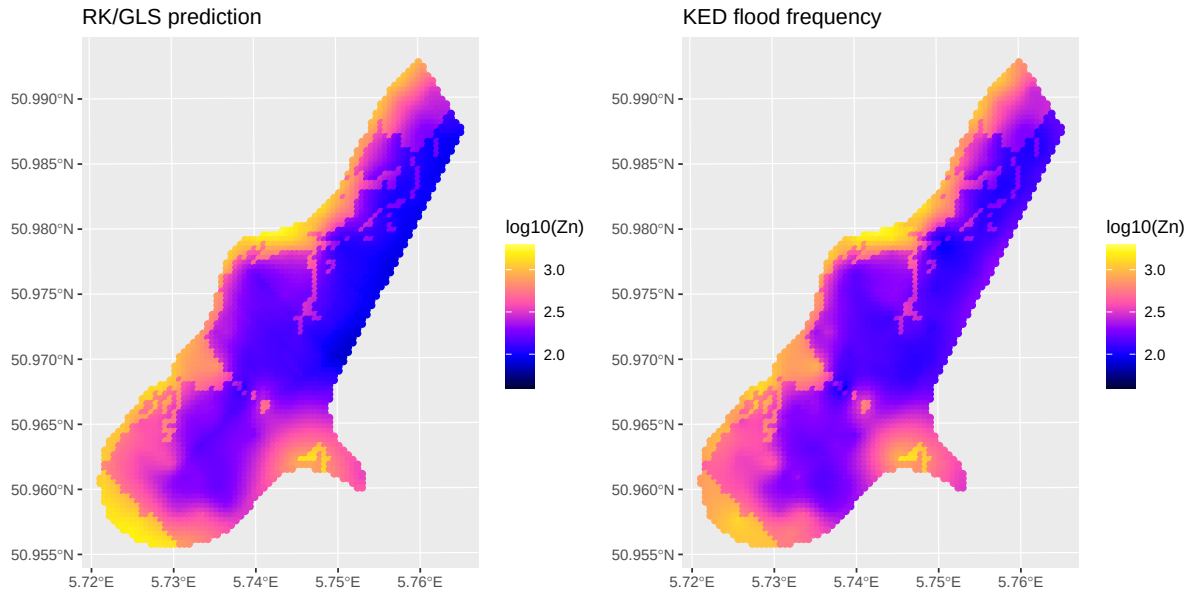
TASK 109 : Krige the GLS residuals onto the prediction grid, add them to the GLS predictions, and display the resulting RK map, along with the KED map and their difference. •

Note that we use Ordinary Kriging, not Simple Kriging, because we can not assume the mean residual from a GLS model is zero; in fact, we know it is biased.

```
k.gls.r <- krige(gls.resid ~ 1, loc=meuse.sf,
                 newdata=meuse.grid.sf, model=vmf.r.gls)

[using ordinary kriging]

summary(k.gls.r)

   var1.pred           var1.var                  geometry
 Min.   :-0.42626   Min.   :0.006397   POINT        :3103
 1st Qu.:-0.19085   1st Qu.:0.008765   epsg:28992   :   0
 Median :-0.04808   Median :0.009904   +proj=ster...:   0
 Mean   :-0.04812   Mean   :0.010982
 3rd Qu.: 0.09199   3rd Qu.:0.012181
 Max.   : 0.30647   Max.   :0.025632

k.gls.r$rk.gls.pred <-
    meuse.grid.sf$gls.pred + k.gls.r$var1.pred
k.gls.r$diff.rk.gls.ked <-
    k.gls.r$rk.gls.pred - kr240$var1.pred
summary(k.gls.r$diff.rk.gls.ked)

      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
-1.266e-01 -1.208e-02 -2.224e-05  4.115e-03  1.412e-02  1.570e-01

zmax <- round(max(k.gls.r$rk.gls.pred,
                  kr240$var1.pred), 1) + 0.1
zmin <- round(min(k.gls.r$rk.gls.pred,
                  kr240$var1.pred), 1) - 0.1
ramp <- seq(from=zmin, to=zmax, by=.1)
```

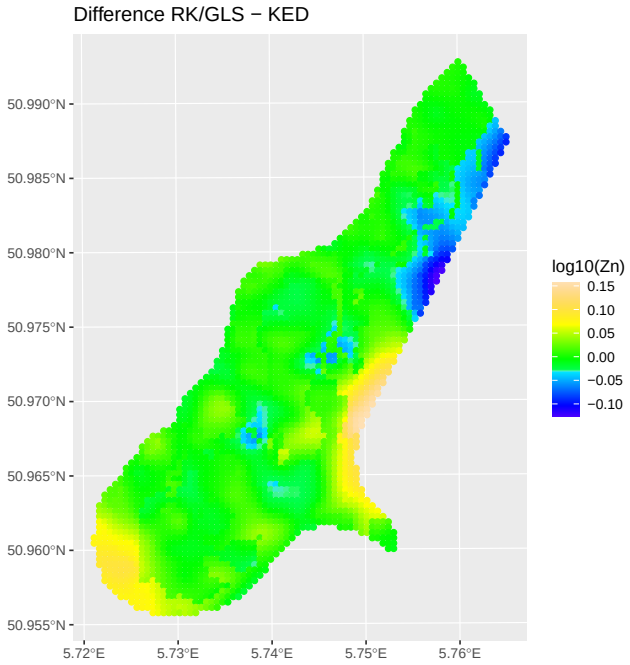
The two predictions:

```
g.glsrk <- ggplot() +
    geom_sf(data = k.gls.r, aes(col = rk.gls.pred)) +
    labs(title = "RK/GLS prediction", col = "log10(Zn)") +
    scale_color_gradientn(limits = range(ramp),
                          colors = sp::bpy.colors(length(ramp)))
g.ked <-ggplot() +
    geom_sf(data = kr40, aes(col = var1.pred))  +
    labs(title = "KED flood frequency", col = "log10(Zn)") +
    scale_color_gradientn(limits = range(ramp),
                          colors = sp::bpy.colors(length(ramp)))
gridExtra::grid.arrange(g.glsrk, g.ked, nrow=1)
```



Their differences:

```
g.glsrk.ked <- ggplot() +
    geom_sf(data = k.gls.r, aes(col = diff.rk.gls.ked))  +
    labs(title = "Difference RK/GLS - KED", col = "log10(Zn)") +
    scale_color_gradientn(colors = topo.colors(64))
plot(g.glsrk.ked)
```



Q90 : *How large are the differences betwen the RK/GLS and KED predictions?* *Jump to A90*

•

15 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**.[33] 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 §11.2 (OK) and §13.3.2 (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 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;

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

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.* [6] 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.

15.1 Independent evaluation set

In some studies there is one sampling campaign, and then the whole dataset is randomly *split* into calibration and evaluation ("validation") sets. But we used all the observations to build our model and predict; we do not have an independent evaluation set. If we split it, re-fit the model with only the calibration set, and predict at the evaluation 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**.

15.2 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 110 : Perform LOOCV for the OK and KED predictions. •

The `krige.cv` method performs this.

```
kcv.ok <- krige.cv(logZn ~ 1, locations=meuse.sf, model=vmf)
kcv.rk <- krige.cv(logZn ~ ffreq, locations=meuse.sf, model=vrmf)
kcv.rk2 <- krige.cv(logZn ~ ffreq*dist, locations=meuse.sf, 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 111 : Summarize the results of the OK cross-validation. •

```
class(kcv.ok)

[1] "sf"         "data.frame"

summary(kcv.ok)

   var1.pred        var1.var          observed
 Min.   :2.105   Min.   :0.02207   Min.   :2.053
 1st Qu.:2.331   1st Qu.:0.02947   1st Qu.:2.297
 Median :2.550   Median :0.03316   Median :2.513
 Mean   :2.556   Mean   :0.03511   Mean   :2.556
 3rd Qu.:2.748   3rd Qu.:0.03785   3rd Qu.:2.829
 Max.   :3.154   Max.   :0.10233   Max.   :3.265
    residual             zscore               fold
 Min.   :-0.428470   Min.   :-2.313324   Min.   :  1.0
 1st Qu.:-0.096429   1st Qu.:-0.501074   1st Qu.: 39.5
 Median :-0.004522   Median :-0.024152   Median : 78.0
 Mean   :-0.000147   Mean   :-0.000147   Mean   : 78.0
 3rd Qu.: 0.085278   3rd Qu.: 0.454976   3rd Qu.:116.5
 Max.   : 0.641388   Max.   : 3.220399   Max.   :155.0
          geometry
 POINT        :155
 epsg:28992   :  0
 +proj=ster...:  0
```

The spatial object 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 112 : 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.
-0.428470 -0.096429 -0.004522 -0.000147  0.085278  0.641388

summary(kcv.rk$residual)
```

```
      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
-0.5612339 -0.0834470 -0.0136438  0.0009373  0.0898658  0.4768219

summary(kcv.rk2$residual)

     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-0.561260 -0.084644  0.001424  0.001413  0.078414  0.590621
```

Q91 : *Is any prediction **biased**? (Compare the mean to zero). Which has the narrower overall and inter-quartile range?* *Jump to A91* •

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

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

[1] 0.1725589

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

[1] 0.1410212

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

[1] 0.1448551
```

Q92 : *Which prediction is, on average, more precise?* *Jump to A92* •

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. This can be visualized by a **bubble plot**: the symbol size is proportional to the absolute value, and the sign is shown by a colour.

Task 113 : Display bubble plots of the OK and KED (flood frequency and distance interaction) cross-validations. •

Again we harmonize the legend scales, here with the maximum absolute residual.

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

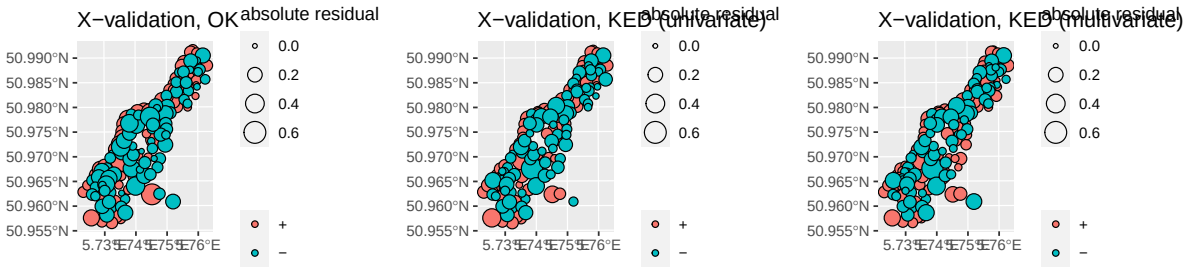
We also ensure identically-formatted graphs by writing a function for this:

```
cv.bubble.plot <- function(cv.result, kriging.type) {
    g <- ggplot(data = cv.result) +
        geom_sf(aes_(size = abs(cv.result$residual),
                     fill = ifelse(cv.result$residual > 0, "green", "red")),
                pch = 21) +
        labs(title = paste("X-validation,", kriging.type),
             size = "absolute residual", fill = "") +
        scale_fill_discrete(labels=c('+', '-')) +
        scale_size_continuous(limits = c(0, zmax))
```

```
    return(g)
}
```

Now apply the function to all three objects and display the plots side-by-side:

```
g.cv.ok <- cv.bubble.plot(kcv.ok, "OK")

Warning:  'aes_()' was deprecated in ggplot2 3.0.0.
i Please use tidy evaluation ideoms with 'aes()'

g.cv.rk <- cv.bubble.plot(kcv.rk, "KED (univariate)")
g.cv.rk2 <- cv.bubble.plot(kcv.rk2, "KED (multivariate)")
gridExtra::grid.arrange(g.cv.ok, g.cv.rk, g.cv.rk2, nrow=1)
```



Q93 : *Is there any pattern to the positive and negative residuals? Is there a difference between the OK and KED patterns?* *Jump to A93* •

It's always good practice to remove temporary variables:

```
rm(zmax, g.cv.ok, g.cv.rk, g.cv.rk2, cv.bubble.plot)
```

Challenge: Repeat the cross-validation exercise for the other heavy metal (Cu, Pb, Cd) that you used for the challenges at the end of §11 and §13. Are your results similar to those for $\log_{10}$Zn?

16 * 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:

$$y = \beta_0 + \sum_k \beta_k x_k \tag{25}$$

we allow *functions* f_k of these:

$$y = \beta_0 + \sum_k f_k(x_k) \tag{26}$$

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.* [13, §9.1] give a thorough explanation of GAM; a simplified explanation of the same material is given in James *et al.* [17, §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.

Q94 : *What theory of the origin of the Zn is this model testing?* *Jump to A94* •

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 114 : Load the `mgcv` package into the workspace. •

```
library(mgcv)

This is mgcv 1.8-41.  For overview type 'help("mgcv-package")'.
```

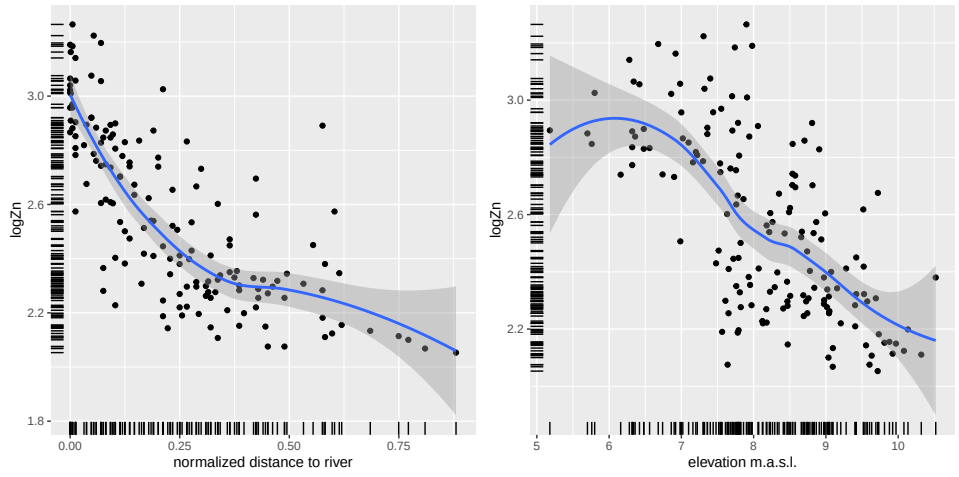
TASK 115 : Display a scatterplot of the two predictors against the $\log_{10}$Zn, with an empirical smoother provided by the `ggplot2` graphics package. •

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.

We also use the `geom_rug` function to show the actual values of each variable in the plot margins.

```
library(ggplot2)
p1 <- ggplot(data = meuse, aes(dist, logZn)) +
    geom_point() +
    geom_smooth(method="loess") +
    geom_rug()  +
    labs(x = "normalized distance to river")
p2 <- ggplot(data = meuse, aes(elev, logZn)) +
    geom_point() +
    geom_smooth(method="loess")  +
    geom_rug() +
    labs(x = "elevation m.a.s.l.")
require(gridExtra)

Loading required package:  gridExtra

grid.arrange(p1, p2, ncol=2)

`geom_smooth()` using formula = 'y ~ x'

`geom_smooth()` using formula = 'y ~ x'
```



Q95 : *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 A95*

•

TASK 116 : 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)  2.55616    0.01479   172.8   <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) 3.561  4.423 65.91  <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.
-0.495862 -0.113870 -0.007694  0.000000  0.090849  0.613842

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)  2.55616    0.01866     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) 3.798   4.76 26.98  <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.
-0.59112 -0.14982 -0.01936  0.00000  0.13746  0.66824
```

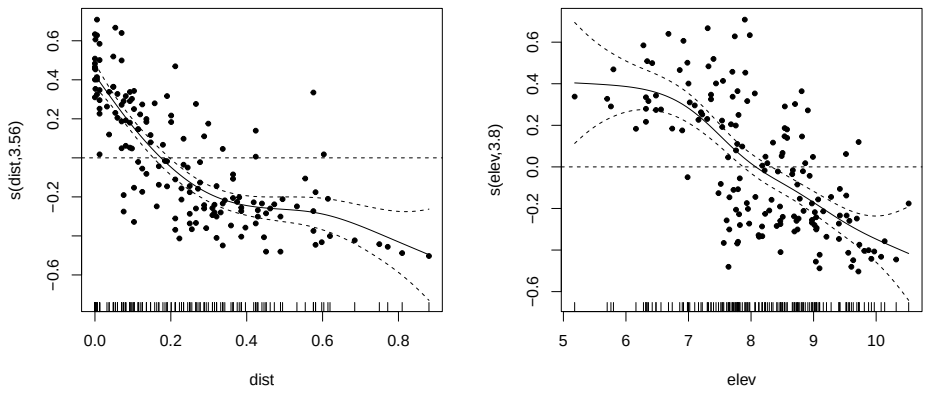
The model summary gives the adjusted R^2, i.e., proportion of variance explained, and a closely-related statistic, the deviance explained.

Q96 : *Do both predictors provide useful information about the metal concentration? Which of the two predictors is better?* *Jump to A96* •

TASK 117 : 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))
```



Q97 : *Describe the fitted relations. Do these fit the hypothesis of the origin of the metals?* *Jump to A97* •

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

```
ggplot(data = meuse, aes(dist, elev)) +
    geom_point() +
    geom_smooth() +
    geom_rug()
```

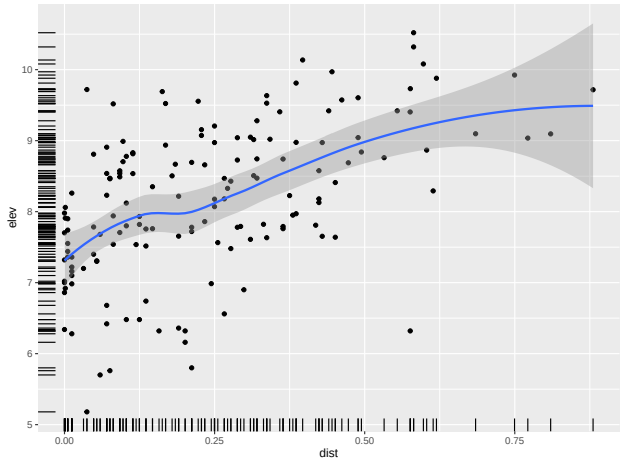
```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

```
cor(meuse$elev, meuse$dist, method='pearson')
```

```
[1] 0.5301023
```

```
cor(meuse$elev, meuse$dist, method='spearman')
```

```
[1] 0.5463665
```



Q98 : *How are elevation and distance related? Is it a strong relation?* *Jump to A98* •

TASK 119 : 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)  2.55616    0.01207   211.7   <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) 3.915  4.839 41.68  <2e-16 ***
s(elev) 6.141  7.310 10.66  <2e-16 ***
---
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)  2.55586    0.01598   159.9   <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)      3.768  4.685 36.866  <2e-16 ***
s(elev)      4.298  5.469 15.117  <2e-16 ***
ti(dist,elev) 4.716  5.201  3.123  0.0107 *
---
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.
-0.41600 -0.08573 -0.01559  0.00000  0.09119  0.37517

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

    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.41519 -0.07618 -0.01417  0.00000  0.08543  0.34944
```

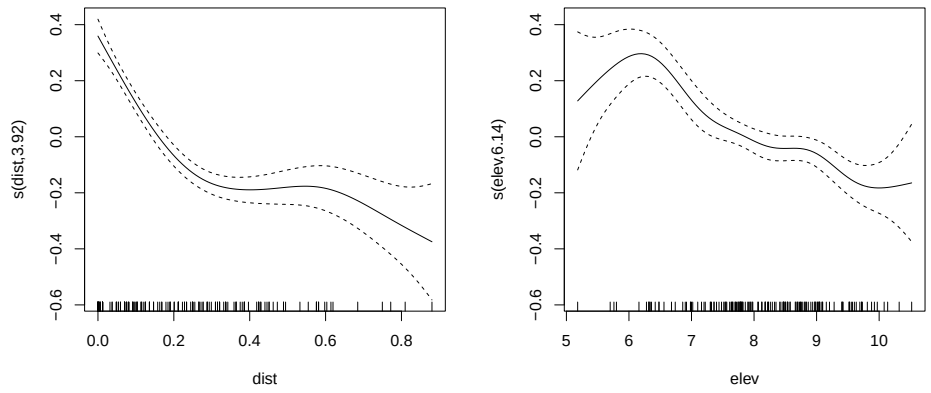
Q99 : *Are these models better than either of the two single-predictor models? Is the interaction model better than the additive model? (Hint: look at the approximate significance of the interaction term.)* *Jump to A99* •

TASK 120 : 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.

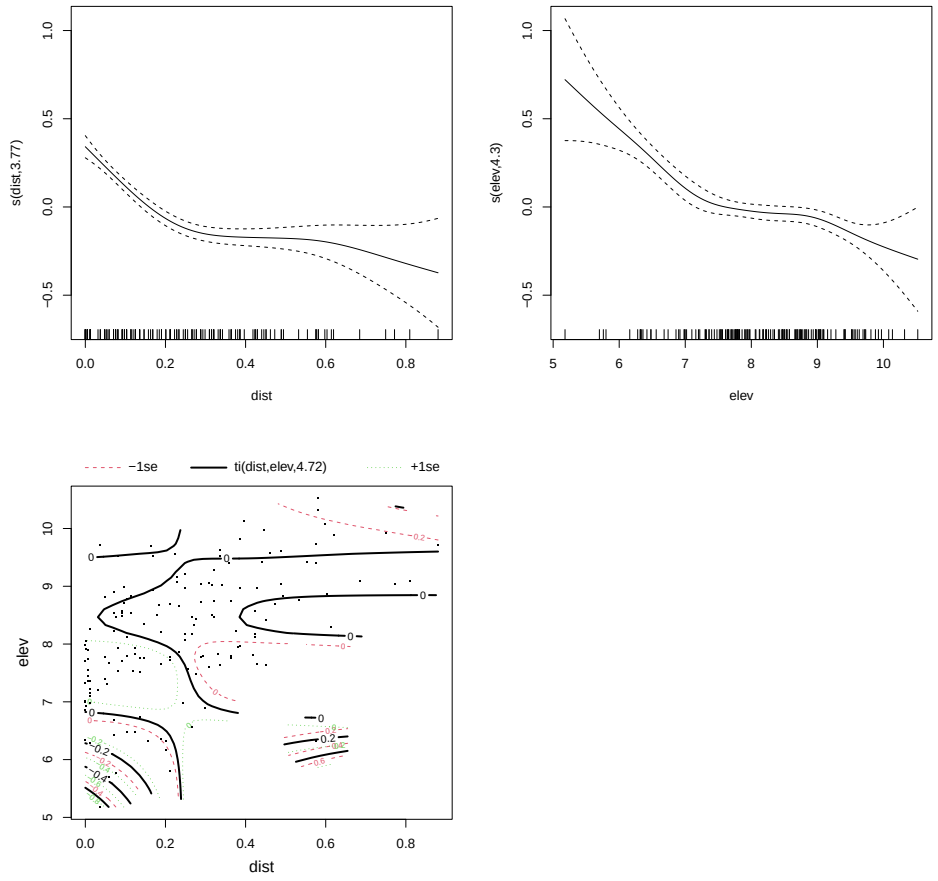
First the additive model:

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



Then the interaction model:

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



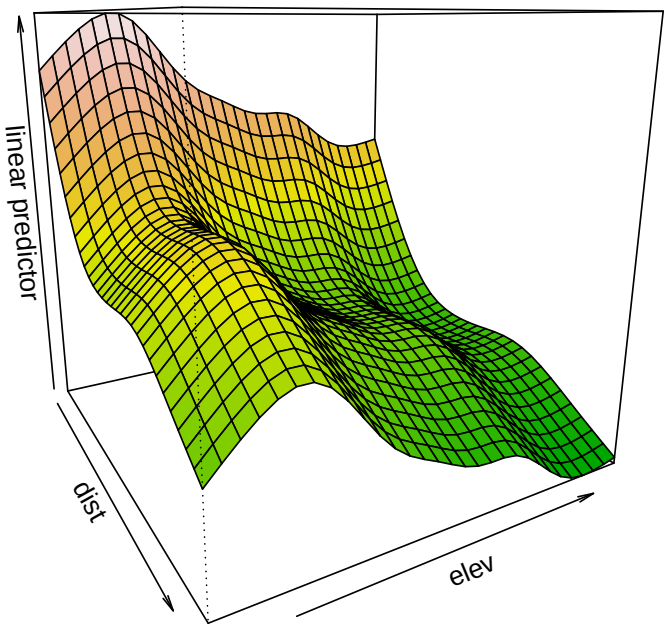
Q100 : *Compare the additive model to the two single-factor models.*

How do the smooth fits differ? Where does the interaction tensor most modify the additive relation? *Jump to A100* •

TASK 121 : Display a 3D plot of the response surface to distance and elevation. •

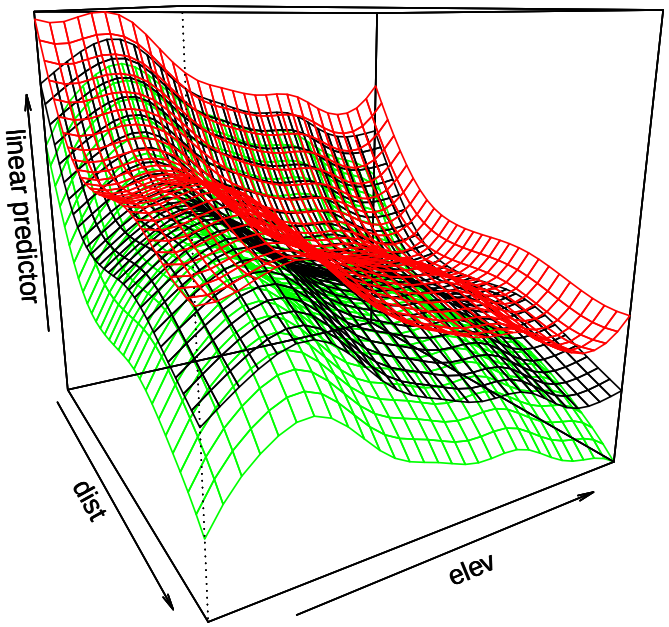
A very nice plot to show this is `vis.gam`.

```
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. For this, we need the prediction grid to include all the predictors we used to build the GAM, i.e., normalized distance and elevation. Normalized distance was already in the sample data `meuse.grid`, but elevation was not. We added that in §9.1.1, above.

TASK 122 : 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.sf, se.fit=TRUE)
names(tmp)

[1] "fit"    "se.fit"

meuse.grid.sf$k.g.i <- tmp$fit
meuse.grid.sf$k.g.i.se <- tmp$se.fit

g.gam <- ggplot() +
    geom_sf(data = meuse.grid.sf, aes(col = k.g.i)) +
    labs(title = "GAM prediction", col = "log10(Zn)") +
    scale_color_gradientn(colors = bpy.colors(64))

Error in bpy.colors(64):  could not find function "bpy.colors"
```

```
g.gam.se <- ggplot() +
    geom_sf(data = meuse.grid.sf, aes(col = k.g.i.se)) +
    labs(title = "GAM prediction standard error", col = "log10(Zn)") +
    scale_color_gradientn(colors = cm.colors(64))
grid.arrange(g.gam, g.gam.se, ncol=2)
```

```
Error in arrangeGrob(...):  object 'g.gam' not found
```

Q101 : *Comment on the suitability of this GAM.* *Jump to A101* •

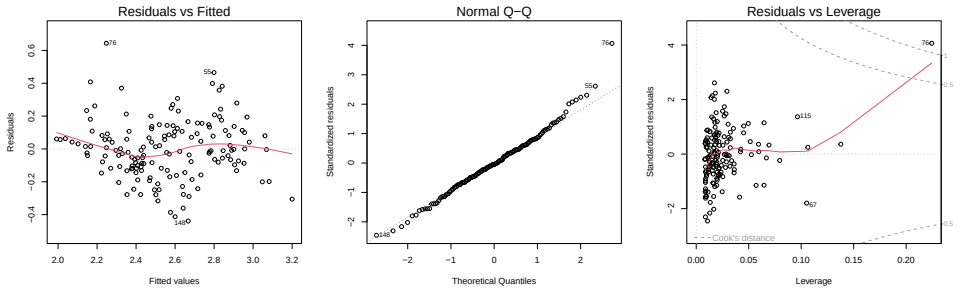
We can compare this result with a linear model using the same formula.

Task 123 : 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)  4.13897    0.17885  23.142  < 2e-16 ***
dist        -3.02561    0.62826  -4.816 3.53e-06 ***
elev        -0.16886    0.02267  -7.448 6.76e-12 ***
dist:elev    0.25228    0.07220   3.494 0.000624 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1797 on 151 degrees of freedom
Multiple R-squared:  0.6779,Adjusted R-squared:  0.6715
F-statistic: 105.9 on 3 and 151 DF,  p-value: < 2.2e-16
```

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



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 124 : Plot the predictions and their standard errors. •

```
tmp <- predict.lm(object=m.dist.elev.i,
                  newdata=meuse.grid.sf, se.fit=TRUE)
meuse.grid.sf$k.i <- tmp$fit
meuse.grid.sf$k.i.se <- tmp$se.fit

g.lm <- ggplot() +
    geom_sf(data = meuse.grid.sf, aes(col = k.i)) +
    labs(title = "OLS prediction", col = "log10(Zn)") +
    scale_color_gradientn(colors = bpy.colors(64))
```

```
Error in bpy.colors(64):  could not find function "bpy.colors"
```

```
g.lm.se <- ggplot() +
    geom_sf(data = meuse.grid.sf, aes(col = k.i.se)) +
    labs(title = "OLS prediction standard error", col = "log10(Zn)") +
    scale_color_gradientn(colors = cm.colors(64))
gridExtra::grid.arrange(g.lm, g.lm.se, ncol=2)
```

```
Error in arrangeGrob(...):  object 'g.lm' not found
```

TASK 125 : 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")]
```

```
   zinc elev     dist
76  778 6.32 0.575877
```

```
log10(meuse[ix,"zinc"])
```

```
[1] 2.89098
```

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

```
      76
2.247543
```

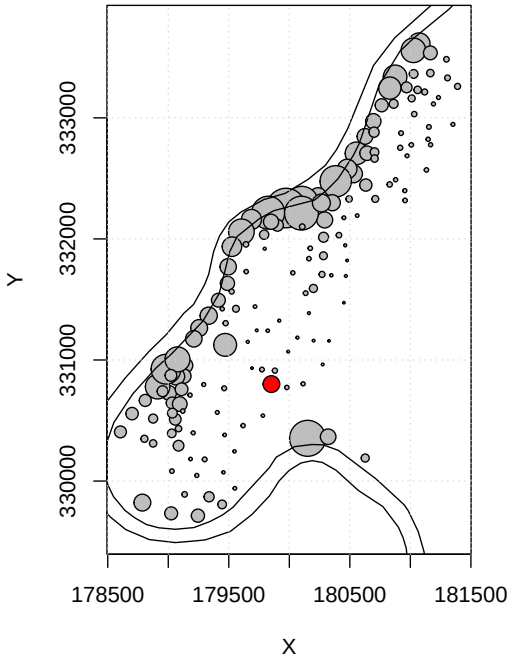
```
fitted(m.g.dist.elev.i)[ix]
```

```
[1] 2.920624
```

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

```
Warning in data(meuse.riv):  data set 'meuse.riv' not found
```

```
lines(meuse.riv)
grid()
```

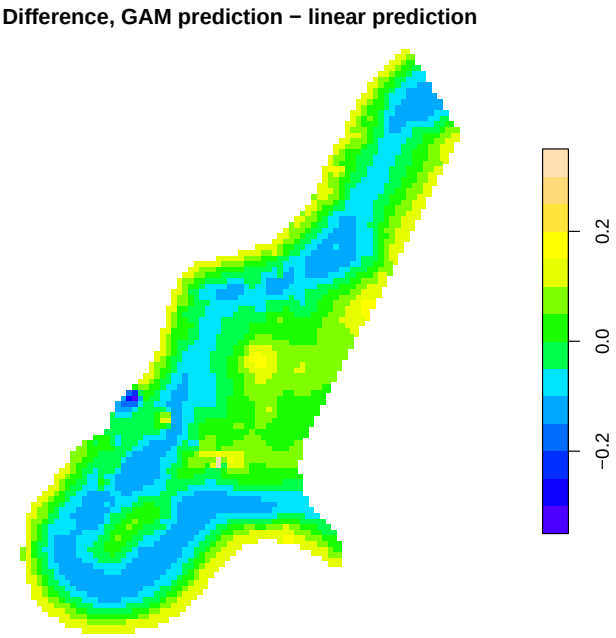


Q102 : *What is the anomaly at this observation?* *Jump to A102* •

TASK 126 : 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.sf$diff.gam.lm <- meuse.grid.sf$k.g.i - meuse.grid.sf$k.i
plot(meuse.grid.sf["diff.gam.lm"],
     pal = topo.colors, pch = 15,
     main="Difference, GAM prediction - linear prediction")
```



Q103 : *What are the differences between the two models? Which appears more realistic?* *Jump to A103*

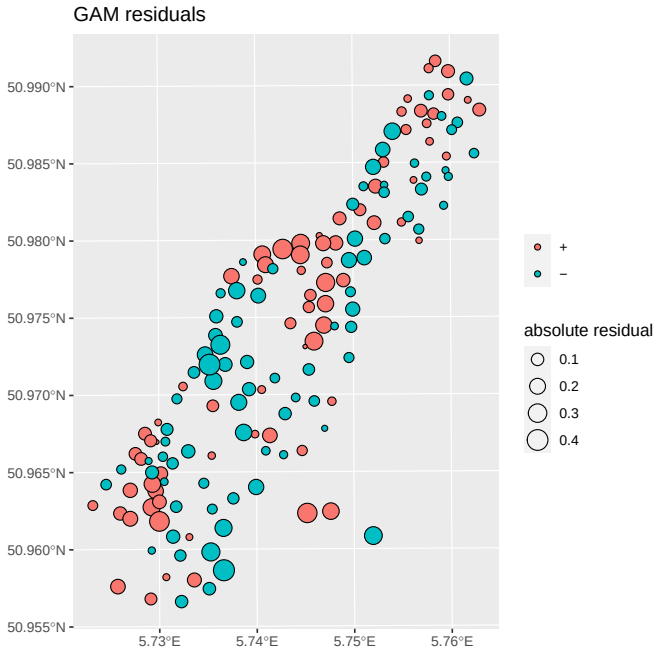
•

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

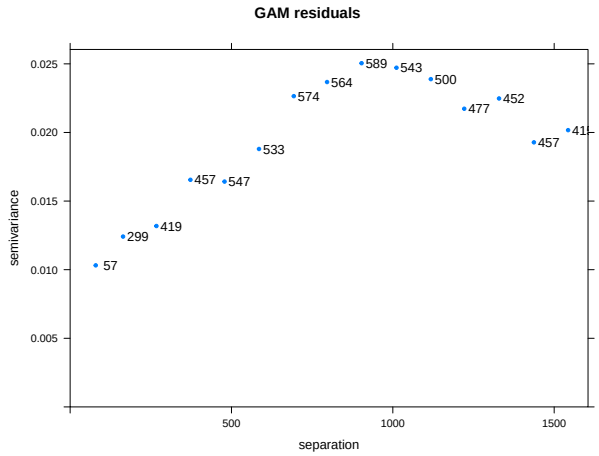
Now that we've seen that the GAM does a fairly good job, let's see if there is any residual spatial correlation.

TASK 127 : Display a bubble plot of the GAM model residuals and the residual variogram •

```
meuse.sf$resid.gam <- residuals(m.g.dist.elev.i)
ggplot(data = meuse.sf) +
    geom_sf(aes(size = abs(resid.gam),
                fill = ifelse(resid.gam > 0, "green", "red")),
            pch = 21) +
    labs(title = "GAM residuals",
         size = "absolute residual", fill = "") +
    scale_fill_discrete(labels=c('+', '-'))
```




```
vr <- variogram(resid.gam ~ 1, locations=meuse.sf)
print(plot(vr, plot.numbers=T, main = "GAM residuals",
      pch = 20, xlab = "separation"))
```



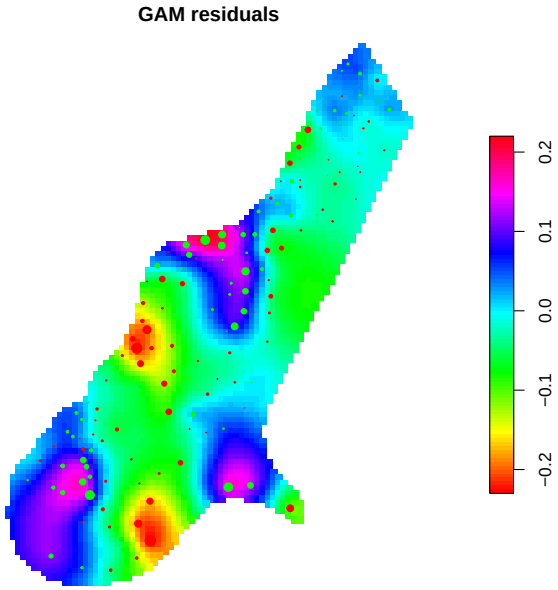
Q104 : *Do the residuals show spatial dependence?* *Jump to A104* •

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%:

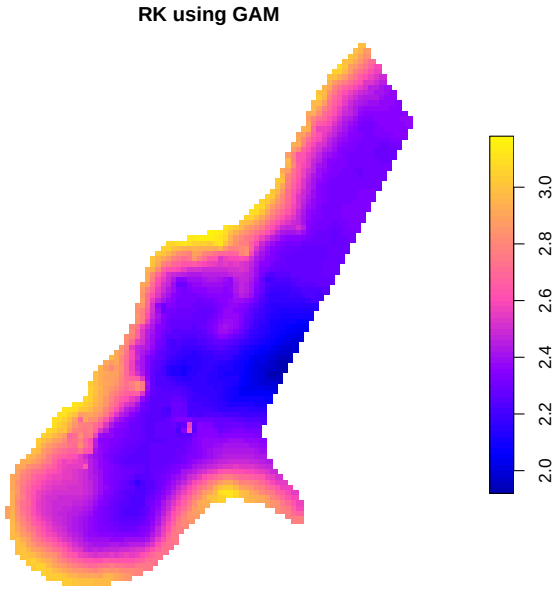
```
max(vr$gamma)/max(variogram(logZn ~ 1, locations=meuse.sf)$gamma)

[1] 0.1887966
```

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:



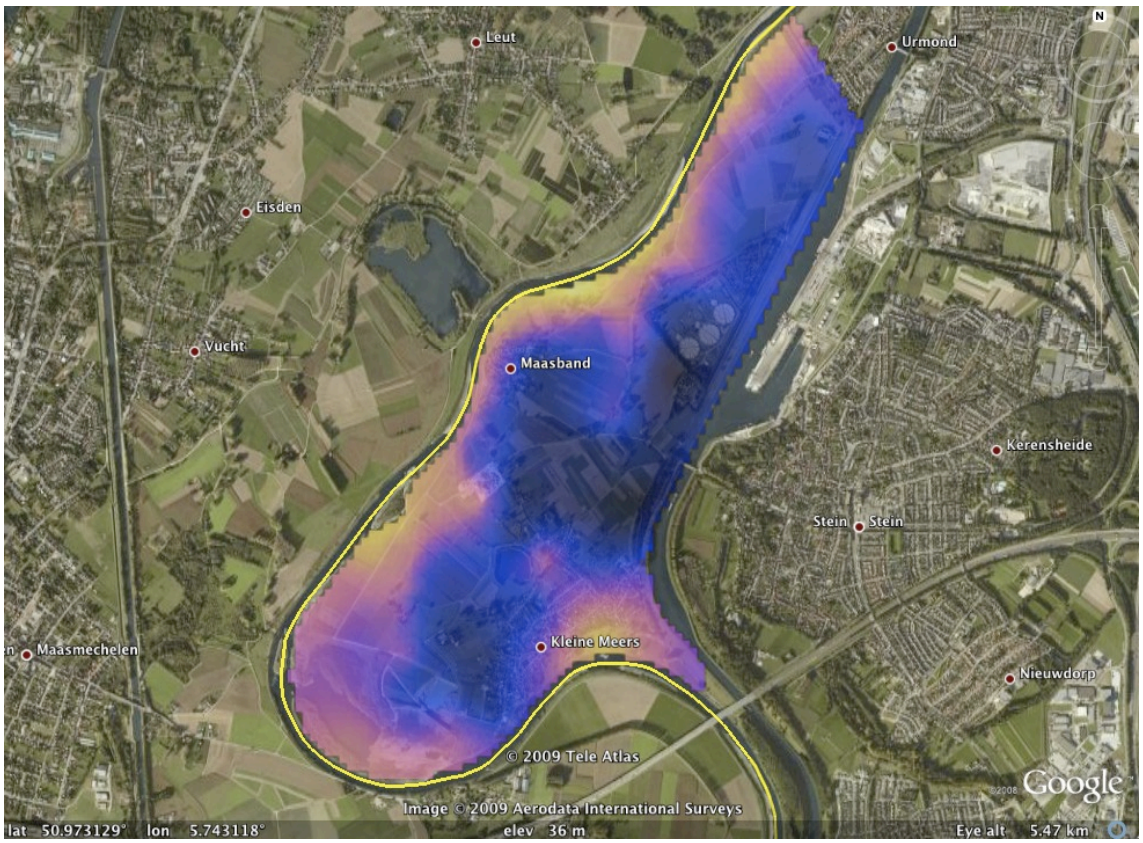
The Regression Kriging using the GAM and SK of the residuals look like this:



17 Final words

There is much more to geostatistics than this simple introduction. Some good reference texts are by Goovaerts [12], Webster & Oliver [28] and Isaaks & Srivastava [16]. For spatial analysis in R, the essential reference is Bivand *et al.* [2]. For general modelling in R, a good text is Fox [10]. For data-driven approaches, see Hastie *et al.* [13] and a simplified version in James *et al.* [17]

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:



18 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 : *Minimum: 14; Median: 31; Maximum: 128* *Return to Q5* •

A6 : *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 Q6* •

A7 : *Minimum: 113, first quartile: 198, median: 326, third quartile: 674.5, maximum: 1839, mean: 469.72.* *Return to Q7* •

A8 : *The mean is well above the median, this implies a right-skew.* *Return to Q8* •

A9 : *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 Q9* •

A10 : *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 Q10* •

A11 : *Variance explained: 80.4%.* *Return to Q11* •

A12 : *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 Q12* •

A13 : *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 Q13* •

A14 : *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 Q14* •

A15 : *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*[34]*. This point does have a high Zn concentration, 703 mg kg^{-1}.* *Return to Q15* •

A16 : *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 Q16* •

A17 : *No, the high-leverage points do not have high Cook's distance, that is, they are consistent with the overall model fit.* *Return to Q17* •

A18 : *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 Q18* •

A19 : *The most frequently flooded soils have all the high metal concentrations, as well as a higher median. The other two classes are almost identical.*

[34] 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.

But, all three classes contain observations with low concentrations. *Return to Q19* •

A20 : *Variance explained: 24.3%* *Return to Q20* •

A21 : *To answer this, look at the summary table of the linear model. The number 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}*).*

The mean of each group can also be computed directly:

```
round(tapply(meuse$logZn, meuse$ffreq, mean),3)

    1     2     3
2.700 2.368 2.425
```

Return to Q21 •

A22 : *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 Q22* •

A23 : *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 Q23*
•

A24 : *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 Q24* •

A25 : *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 Q25* •

A26 : *The field named* `geometry` *contains the spatial information; note that fields* `x` *and* `y` *were removed from the* `data.frame` *to this special status. The geometry is shows as class* `sfc_POINT`*, a 2D matrix.* *Return to Q26* •

A27 : *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 Q27* •

A28 : *Flood frequency class* 1 *is the highest frequency, "1 = once in two years". It is mostly close to the river but there are some swales or backswamps away from the river.* *Return to Q28* •

A29 : *We suspect that 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 agriculture.*

Certain soils may retain heavy metals more easily than others. *Return to Q29* •

A30 : *The tree has 36 leaves and 35 internal nodes.* *Return to Q30* •

A31 : *Normalized distance to river is the most important, followed by elevation. Flooding frequency and soil type are less important.* *Return to Q31* •

A32 : *The model appears to be overfit; after 6 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 Q32* •

A33 : *The minimum cross-validation error is 0.296; this corresponds to 6 splits and a complexity parameter of 0.0106.* *Return to Q33* •

A34 : *The pruned tree has 7 leaves and 6 internal nodes. These are 19.4% and 17.1% of the original number, respectively. The tree is much smaller. with many fewer unique fitted values.* *Return to Q34* •

A35 : *Only normalized 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 Q35* •

A36 : *The first split is on distance: closer than 0.1602 normalized units 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. The interpretation is that the distance floodwater travels is the most important, and the elevation then separates deeper from shallower floods.* *Return to Q36* •

A37 : *The extreme values are smoothed because each prediction is a weighted average, so at the extreme elevations several less-extreme values are included in the result of the weighted averaging.* *Return to Q37* •

A38 : *The parsimonious model only predicts 7 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 variability not captured by these factors.* *Return to Q38* •

A39 : *(1) The full model correctly classifies 51% of the observations. This is the sum of the CP values for all the splits in the tree – each CP in the table is the increase in* R^2 *due to that split.*

(2) It is minimum at 10 splits.

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

Return to Q39 •

A40 : *The model is moderately successful. It correctly classifies* 76 *out of the total* 81 *observations in class 1 (frequent),* 41 *out of the total* 54 *observations in class 1 (moderate),* 16 *out of the total* 20 *observations in class 1 (rare), for an overall success rate for the fitted model of* 85.8%

Note this is not the predictive power, just the model fit with the least cross-validation error. Still, this is a fairly successful model. Flooding frequency can be well-modelled by distance from the river and elevation. The soil type is used in the model for somewhat higher points; these are likely on the levee. *Return to Q40* •

A41 : *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 Q41* •

A42 : *As with the single regression tree, distance to river is the most important, accounting for about 61% of the sum of the increase in MSE from all the*

permutations. This is followed by elevation, accounting for 25%. *Return to Q42* •

A43 : *Distance to river is the root in about 3/5 of the trees, which is likely the trees where it was a possible predictor at this level. Elevation was also used as a root, likely when distance was not used. Elevation was the most common predictor at the second split.* *Return to Q43* •

A44 : *In general it is a synergistic effect: further distances and higher elevations result in lower metal concentrations. There are some interesting deviations, for example around 0.15 distance and 8 m elevation.* *Return to Q44* •

A45 : *The concentration drops rapidly with increasing distance, and then decreases much more slowly. The relation is not monotonic, with some local deviations where increasing distance slightly increases the concentration.* *Return to Q45* •

A46 : *As the flood frequency decreases the pattern of interaction between distance and elevation is similar; however, the maximum values of $log_{10}Zn$ decrease.* *Return to Q46* •

A47 : *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 Q47* •

A48 : *The out-of-bag RMSE is 0.15 log(mg kg^{-1}), about double the fits RMSE, 0.083 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 Q48* •

A49 : *This map is much smoother and has more predicted values than the map produced by the regression tree.* *Return to Q49* •

A50 : *(1) Class 1: 77.2%; class 2: 49.1%; class 3: 0%. 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 Q50* •

A51 : *Yes, big circles tend to be near other big ones, same for small circles.* *Return to Q51* •

A52 : *There are (155 * (155-1))/2 =* 1.1935×10^4 *point-pairs.* *Return to Q52* •

A53 : *Separation is 70.84 m, semivariance is 0.001144 log(mg* $kg^{-1})^2$. *Return to Q53* •

A54 : *Most of the point-pairs have very low semi-variance, i.e., they have quite similar values of* log_{10}*Zn. However, points 76 and 138 are only separated by 112 m but have a semivariance of 0.3025.* *Return to Q54* •

A55 : *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 Q55* •

A56 : *The evidence is that at closer separations the semivariance is, on average, lower. This increases steadily until the range.* *Return to Q56* •

A57 : *At about 850 m separation there is not any reduction in average semivariance; this is the range.* *Return to Q57* •

A58 : *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 Q58* •

A59 : *At the range and beyond the average semivariance is about 0.13 log(mg* $kg^{-1})^2$*; this is the total sill.* *Return to Q59* •

A60 : *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 Q60* •

A61 : *The kriged map is very smooth. "Hot" and "cold" spots are clearly controlled by observations with especially high and low values.* *Return to Q61* •

A62 : *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 Q62* •

A63 : *Only NA of the 155 observations above the threshold.* *Return to Q63* •

A64 : *The range is 1527 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 Q64* •

A65 : *The total sill is 0.1457; units are dimensionless.* *Return to Q65* •

A66 : *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 Q66* •

A67 : *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 prediction variance can also be computed from the linear model and is also uniform for each class; it depends on the variance of that class' observations and the mean value of the class. 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.

See §7.3 for the linear model from the flood frequency class. *Return to Q67* •

A68 : *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 Q68* •

A69 : *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 Q69* •

A70 : *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 Q70* •

A71 : *Distance from river* `dist`*, flooding frequency class* `ffreq`*, and soil type* `soil`*.* *Return to Q71* •

A72 : *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 Q72* •

A73 : *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 Q73* •

A74 : *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 (mean) of the distances:*

```
ix <- which.min(k.dist$var1.var)
k.dist[ix,]

Simple feature collection with 1 feature and 2 fields
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: 179700 ymin: 331500 xmax: 179700 ymax: 331500
Projected CRS: Amersfoort / RD New
     var1.pred   var1.var              geometry
1312  2.550972 0.04512927 POINT (179700 331500)

meuse.grid[ix, "dist"]

[1] 0.244441

mean(meuse$dist)

[1] 0.2400169
```

These are not exactly the same because there is no cell exactly at the mean distance of the observation points. *Return to Q74* •

A75 : *Yes, the two-predictor models give significantly lower residual sum-of-squares.* *Return to Q75*
•

A76 : *Yes, the interaction model has lower RSS than the additive model. The probability this is due to chance is only 0.0025.* *Return to Q76* •

A77 : *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 Q77* •

A78 : *Variance explained: 63.5%.* *Return to Q78* •

A79 : *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, identified in the graph.* Return to Q79 •

A80 : *Yes, normally-distributed except for the three most positive residuals.* Return to Q80 •

A81 : *The high-leverage residuals do not have high influence as shown by Cook's distance, this is good.* Return to Q81 •

A82 : *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 Q82 •

A83 : *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 Q83 •

A84 : *Adding distance to river reduces the prediction variances and makes them almost uniform across the area.* Return to Q84 •

A85 : *All coefficients vary; the intercept is only a little bit different, but the interaction terms are quite different. This is likely because GLS accounts for geographic clustering of particularly high and low values, which are at the "edges" of multivariate feature space and thus have high leverage in the interaction terms.* Return to Q85
•

A86 : *The range of spatial dependence has been adjusted; from the residual variogram fit, i.e., 665 m; the REML estimate is somewhat longer, 1163 m. These are 1/3 of the effective range, since we fit an exponential model. The effective range found by* `gls` *is thus 3489 m.* Return to Q86 •

A87 : *The areas near the river and more flooded are predicted to have somewhat lower* $\log_{10}$*Zn concentrations with the GLS model. Again, this is because GLS accounts for geographic clustering of the highest values (along the river).* Return to Q87 •

A88 : *The largest differences are where the interaction terms were most*

important, i.e., high flood frequency and far distance from river. We saw that these terms were the ones most affected by the GLS fit. *Return to Q88* •

A89 : *The empirical variogram estimates of the GLS residuals (blue points) are similar to those from the OLS residuals (red points). The variogram model fit by* `gls` *(blue line) is reasonable, but has stronger spatial correlation at close ranges than the fit to the empirical variogram (red dashed line). The fit from the OLS residuals adjusts better to the close-range bins; but recall, the correlation structure uses all the points, not as summarized in an empirical variogram.* *Return to Q89*
•

A90 : *The differences are fairly small, never more than* ± 0.11 log_{10}*Zn mg kg*$^{-1}$*. The patterns are almost identical.* *Return to Q90* •

A91 : *The means of the cross-validations are* -1.47×10^{-4} *(OK),* 9.37×10^{-4} *(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 Q91*
•

A92 : *The one-factor KED prediction, with flood frequency as co-variable, is most precise; RMSE is 0.141 log(mg kg*$^{-1}$*).* *Return to Q92* •

A93 : *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 Q93* •

A94 : *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 Q94* •

A95 : *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 Q95* •

A96 : *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 Q96* •

A97 : *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 Q97* •

A98 : *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 Q98* •

A99 : *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 Q99* •

A100 : *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 Q100* •

A101 : *The GAM prediction clearly mostly depends on the distance from river, with some adjustments for elevation.* *Return to Q101* •

A102 : *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 Q102* •

A103 : *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 Q103* •

A104 : *Yes, there is strong spatial dependence.* *Return to Q104* •

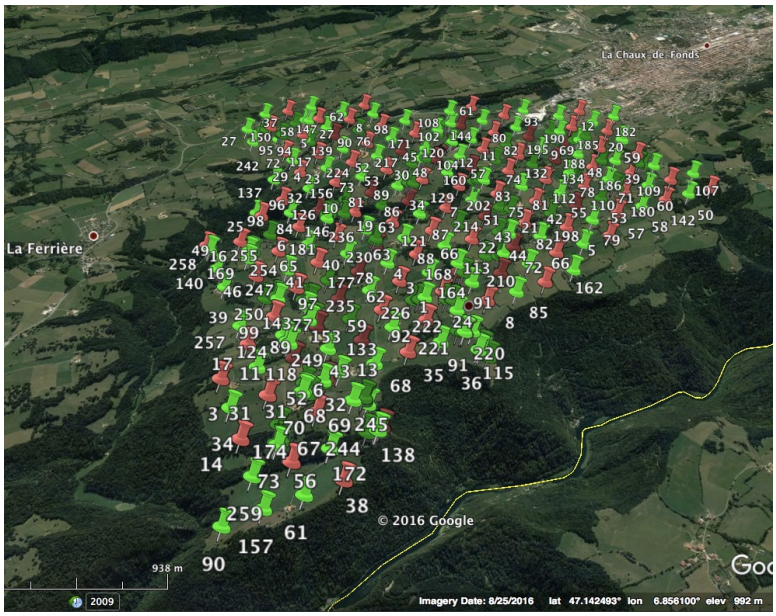
19 Assignment

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] Bates, D. 2005. *Fitting linear mixed models in R. R News* **5**(1):27–30 119

[2] Bivand, R. S.; Pebesma, E. J.; & Gómez-Rubio, V. 2008. *Applied Spatial Data Analysis with R*. UseR! Springer. http://www.asdar-book.org/ 149

[3] Breiman, L. 2001. *Statistical modeling: The two cultures (with comments and a rejoinder by the author). Statistical Science* **16**(3):199–231 18, 40

[4] Breiman, L.; Friedman, J. H.; Olshen, R. A.; & Stone, C. J. 1983. *Classification and regression trees*. Wadsworth 40

[5] 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 174

[6] Brus, D.; Kempen, B.; & Heuvelink, G. 2011. *Sampling for validation of digital soil maps. European Journal of Soil Science* **62**:394–407. ISSN 13510754 130

[7] Cook, R. & Weisberg, S. 1982. *Residuals and influence in regression*. New York: Chapman and Hall. ISBN 0-412-24280-X 23, 24, 108

[8] Draper, N. R. & Smith, H. 1998. *Applied Regression Analysis*. Wiley-Interscience, third edition edition. ISBN 978-0-471-17082-2 23

[9] Efron, B. & Gong, G. 1983. *A leisurely look at the bootstrap, the jackknife & cross-validation. American Statistician* **37**:36–48 60

[10] Fox, J. 2002. *An R and S-PLUS Companion to Applied Regression*. Newbury Park: Sage 149

[11] Fox, J. 2016. *Applied regression analysis and generalized linear models*. SAGE, 3rd ed edition. ISBN 978-1-4522-0566-3 23, 24, 108

[12] Goovaerts, P. 1997. *Geostatistics for natural resources evaluation*. Applied Geostatistics. New York; Oxford: Oxford University Press 149

[13] 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 18, 40, 54, 60, 134, 149

[14] Hengl, T. 2009. *A Practical Guide to Geostatistical Mapping*. Amsterdam. ISBN 978-90-9024981-0
URL http://spatial-analyst.net/book/ 9, 174

[15] Ihaka, R. & Gentleman, R. 1996. *R: A language for data analysis and graphics. Journal of Computational and Graphical Statistics* **5**(3):299–314 1

[16] Isaaks, E. H. & Srivastava, R. M. 1990. *An introduction to applied geostatistics*. New York: Oxford University Press 149

[17] 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 18, 40, 54, 60, 134, 149

[18] 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 72

[19] 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):799–813 117, 119

[20] Pebesma, E. 2018. *Simple Features for R: standardized support for spatial vector data. The R Journal* **10**(1):439–446. ISSN 2073-4859 URL https://journal.r-project.org/archive/2018/RJ-2018-009/index.html 1

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

[22] 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/ 1

[23] Pinheiro, J. C. & Bates, D. M. 2000. *Mixed-effects models in S and S-PLUS*. Springer. ISBN 0387989579 118, 119

[24] 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 1

[25] R Development Core Team. 2015. *R Data Import/Export*. The R Foundation for Statistical Computing, version 3.6.2 (2019-12-12) edition
URL http://cran.r-project.org/doc/manuals/R-data.pdf 7

[26] 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 173

[27] Shalizi, C. 2010. *The bootstrap. American Scientist* **98**(3):186–190 URL http://www.americanscientist.org/issues/pub/2010/3/the-bootstrap/3 60

[28] Webster, R. & Oliver, M. A. 2008. *Geostatistics for environmental scientists*. John Wiley & Sons Ltd., 2nd edition 85, 149

[29] Wright, M. N. & Ziegler, A. 2017. *Ranger: a fast implementation of random forests for high dimensional data in C++ and R. Journal of Statistical Software* **77**(1):1–17. ISSN 1548-7660
URL https://www.jstatsoft.org/index.php/jss/article/view/v077i01 1

[30] Xie, Y. 2011. *knitr: Elegant, flexible and fast dynamic report generation with R.* Accessed 04-Mar-2016
URL http://yihui.name/knitr/ 1

Index of Commands

* formula operator, 33, 106, 138
+ formula operator, 32, 42, 106, 138
<- operator, 14, 21
== operator, 45
?, 8
[] operator, 17, 75, 109
$ operator, 11
%in% operator, 110
& operator, 17
~ formula operator, 15, 21, 29, 42, 78, 85

abline, 27
abs, 26
AIC, 107
anova, 32, 107
apply, 72
as.data.frame, 61, 120
as.matrix, 7
asp argument (plot.xy function), 27

beta argument (krige function), 147
block argument (krige function), 85
boxplot, 29
bpy.colors (sp package), 86
breaks argument (hist function), 12

c, 2, 36
cex graphics argument, 73
chisq.test, 31
class, 22, 35, 48
cloud argument (variogram function), 75
cm.colors (sp package), 87, 96
col argument (plot.xy function), 27
col.regions graphics argument, 92
colnames, 18
control argument (rpart function), 47
coordinates (sp package), 110
coords argument (st_as_sf function), 35
correlation argument (gls function), 120
corSpher (nlme package), 120
corStruct class, 120
cp argument (prune function), 49
cp argument (rpart function), 42
cutoff argument (variogram function), 75, 77

data, 6, 7, 37, 39, 170
data argument (predict.ranger function), 67
data function argument, 22
data.frame class, 18, 35, 36, 39, 54, 61, 153
diag, 72
dim, 8
dist, 75

factor class, 35
fit.variogram (gstat package), 81, 98
form argument (corSpher function), 120

geom_rug (ggplot2 package), 135
ggplot2 package, 2, 38, 101, 126, 135
gls (nlme package), 119, 120, 122, 125, 126, 159, 160
graphics package, 38
grid.arrange (gridExtra package), 135
gridExtra package, 135
gstat, 125
gstat package, 1, 2, 4, 7, 11, 79, 81, 99, 162

head, 8, 14
heat.colors, 92
hist, 12, 24

idp argument (idw function), 52
idw (gstat package), 52
ifelse, 110
importance (ranger package), 63
importance argument (ranger function), 61, 62, 70
install.packages, 2
intersect, 103
intervals (nlme package), 122

jura dataset, 162, 165
jura.grid dataset, 165
jura.pred dataset, 162, 164, 166

knitr package, 1
krige (gstat package), 52, 85, 91, 95, 99, 106, 113, 125, 147
krige.cv (gstat package), 131

lattice package, 126, 135
legend, 27

length, 74
library, 6, 35
list class, 35
lm, 20–23, 29, 95, 106, 119, 121
lm class, 24
locations gstat argument, 85
log10, 14
logical class, 35
ls, 7

MARGIN argument (apply function), 72
matrix class, 35, 37
max, 100
meuse dataset, 150
meuse dataset (sp package), 7, 18, 134, 170
meuse.grid dataset (sp package), 39, 51, 85, 142, 170
meuse.riv dataset (sp package), 37, 170
mfrow argument (par function), 25
mgcv package, 134, 136
min, 100
min_depth_frame (randomForestExplainer package), 63
minsplit argument (rpart function), 42
mode, 35
model gstat argument, 85
model argument (krige function), 95, 106
model.matrix, 21
mtry argument (ranger function), 61

names, 103
newdata gstat argument, 85
newdata argument (predict.lm function), 31
newdata argument (predict.rpart function), 50
nlme, 120
nlme package, 119, 122
nmax argument (idw function), 52
notch argument (boxplot function), 29
numeric class, 35

order, 76
outer, 163

package argument (data function), 6, 7
pages argument (plot.gam function), 139
palette, 27
par, 25
partial (pdp package), 65
pch argument (plot function), 27
pdp package, 65
pi constant, 3
plot, 15, 24, 62, 135
plot (sf package), 39, 86, 87
plot.gam (mgcv package), 137, 139
plot.lm, 24, 25
plot.numbers function argument, 78
plot_predict_interaction (randomForestExplainer package), 64
points, 27
predict, 31, 50, 67
predict.gam (mgcv package), 142
predict.randomForest (ranger package), 67
predict.rpart (rpart package), 50
print, 61
printcp (rpart package), 45
prune (rpart package), 49

q, 9
qplot (ggplot2 package), 135

randomForestExplainer package, 63, 64
ranger (ranger package), 61, 68, 70
ranger class, 61, 62, 67
ranger package, 1, 60
read.csv, 7
read.table, 7
require, 6
reset argument (plot function), 87
residuals, 24, 26
row, 110
row.names, 17, 18, 109, 144
rpart, 47, 56
rpart (rpart package), 41, 42, 45, 47, 48, 55
rpart class, 48, 50
rpart package, 1, 40, 41, 54
rpart.control (rpart package), 47
rpart.plot (rpart.plot package), 43, 44
rpart.plot package, 44
rug, 12

s (mgcv package), 136, 138
sample, 54, 59
save.image, 11
scale_color_gradientn (ggplot2 package), 101

se argument (vis.gam function), 141
se.fit argument (predict.lm function), 31
search, 6
seq, 100
sf class, 35–37, 39, 51, 54, 61
sf package, 1, 2, 4, 34, 35, 38, 86
sfc_POINT class, 153
show.vgms (gstat package), 79
sort, 11
sp, 86
sp package, 2, 6, 7, 11, 52, 120, 170
spplot (sp package), 91, 92
st_as_sf (f package), 39
st_as_sf (sf package), 35
st_bbox (sf package), 77
st_crs (sf package), 36, 37
st_linestring (sf package), 37
st_sfc (‘ package), 37
str, 7
sum, 72
summary, 9, 12, 22

table, 29, 31, 71
ti (mgcv package), 138

unique, 50

value argument (corSpher function), 120
variogram (gstat package), 75, 77, 113
vgm (gstat package), 81, 98
vis.gam (mgcv package), 141

which, 16, 26, 109
which argument (plot.lm function), 25
which.max, 76
which.min, 76
width argument (variogram function), 77

A The Meuse dataset

The project that produced this data set is described in the fieldwork report of Rikken & Van Rijn [26].

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 Two prediction grids are provided:

meuse.grid : 40x40m prediction grid on the Dutch RD coördinate system, as used in the `meuse` dataframe.

meuse.grid_ll : same, after transformation to geographical coördinates on the WGS84 datum.

meuse.riv : River Meuse outline

meuse.area : study area outline (covers `meuse.grid`)

Tomislav Hengl has extended the Meuse dataset[35] for his " Practical Guide to Geostatistical Mapping" [14]; this includes a digital 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[36], 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.* [5].

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

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