Applied geostatistics
Exercise 3: Modelling spatial structure from point samples

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

June 26, 2010

Contents

1 introduction **1**

2 Fitting trend surfaces **1**
2.1 Fitting a first-order surface . . . 2
2.2 Fitting a second-order surface . . . 6
2.3 Stepwise regression to simplify a higher-order surface . . . 7
2.4 * Comparing first- and higher-order surfaces . . . 10
2.5 * Linear models on sp objects . . . 14
2.6 Answers . . . 17

3 Calibration and validation datasets **18**
3.1 * Random selection of validation points . . . 20
3.2 Answers . . . 23

4 Fitting variogram models **24**
4.1 Selecting a variogram model . . . 25
4.2 Fitting a variogram model . . . 29
4.3 * Visual comparison of several variogram fits . . . 31
4.4 * Goodness-of-fit . . . 34
4.5 * Fitting models with multiple structures . . . 37
4.6 * Fitting anisotropic variograms . . . 41
4.7 * The effect of sampling on variogram modelling . . . 45

Version 2.21 Copyright © 2006-2010 University of Twente, Faculty ITC All rights reserved. Reproduction and dissemination of the work as a whole (not parts) freely permitted if this original copyright notice is included. Sale or placement on a web site where payment must be made to access this document is strictly prohibited.
To adapt or translate please contact the author (http://www.itc.nl/personal/rossiter).

4.8 Answers 50

5 * Fitting a trend surface with generalized least squares 52
5.1 Theoretical background 53
5.2 The 'topo' sample dataset 54
5.3 Step 1: OLS trend surface 54
5.4 Step 2: OLS residuals 57
5.5 Step 3: Determining the covariance structure 60
5.6 Step 4: Computing the GLS trend surface 63
5.7 Step 5: Re-computing spatial correlation 66
5.8 Answers 68

6 Self-test 70

References 72

Index of R concepts 73

不入虎穴，焉得虎子 (Bù rù hǔ xuè, yān dé hǔzǐ ?)
"If you do not enter the tiger's cave, how can you expect to
catch the tiger cub?"
– Chinese proverb

1 introduction

In this exercise you will learn two ways to model spatial structure: as **regional trend surfaces** and as realizations of locally-autocorrelated **random fields**.

After completing this exercise you should be able to:

1. compute and evaluate the goodness-of-fit of polynomial trend surfaces;
2. divide a dataset into calibration and validation subsets;
3. fit authorized models to experimental variograms;
4. (optionally) compute trend surfaces with generalized least squares.

In exercises 5 §4 and 5a we will see how to combine regional and local structures in one prediction, but for now we consider them separately.

This exercise includes several sections (marked with a *) with topics that are **optional** and not covered in the self-test. You should at least read the first paragraph of these sections to know their topic, since you may want to work through them later.

Task 1 : If R is not already running, start it. If you haven't already done so, load the `gstat` and `sp` libraries, as shown in the previous exercise. •

```
> require(lattice)
> require(sp)
> require(gstat)
```

2 Fitting trend surfaces

In Exercise 2 we used the Cameroon soil properties dataset to compute and visualize a trend surface. However, we did not investigate the linear model underlying it, that is, we did not evaluate the validity of the model. In this Exercise we are concerned with the modelling process; in the following Exercise 4 we will see how to predict at unsampled points from a fitted trend surface.

We continue with the Cameroon soil proprties dataset. This is supplied on the course CD, in directory `datasets\tcp`, as an R data file named `tcp.RData`; it is also on the course website.

Task 2 : Load the R data file `tcp.RData` into your working directory. •

```
> load("tcp.RData")
> ls()
```

```
[1] "tcp"

> str(tcp)

'data.frame':        147 obs. of  4 variables:
 $ UTM_E : num  702638 701659 703488 703421 703358 ...
 $ UTM_N : num  326959 326772 322133 322508 322846 ...
 $ clay35: num  78 80 66 61 53 57 70 72 70 62 ...
 $ pH35  : num  4.8 4.4 4.2 4.54 4.4 ...
```

There are four fields:

UTM_E : Easting

UTM_N : Northing

clay35 : clay content of the fine earth, % by weight, 30–50 cm layer

pH35 : reaction of the soil in 1:1 (by volume) water, 30–50 cm layer

2.1 Fitting a first-order surface

In the previous Exercise 2 (§6) we converted the `tcp` dataset to a spatial (`sp`) object so that we could use the `krige` method of the `gstat` package to interpolate using a trend surface. However in this section we want to use the standard R `lm` ("fit linear models") method, and this is easiest if we use the original data frame, i.e. where the coördinates are just fields without special status.

To compare various modelling approaches, we begin with the simplest trend, i.e. a plane.

Task 3 : Compute the first-order trend surface of subsoil clay content, modelled by the two metric coördinates, using ordinary least squares and summarize the model. •

We use the `lm` method:

```
> ts1 <- lm(clay35 ~ UTM_E + UTM_N, data = tcp)
> summary(ts1)

Call:
lm(formula = clay35 ~ UTM_E + UTM_N, data = tcp)

Residuals:
    Min      1Q  Median      3Q     Max
-31.601  -5.106  -0.363   3.607  20.467

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.50e+02   5.19e+01   -4.83  3.5e-06 ***
UTM_E        6.51e-04   5.97e-05   10.91  < 2e-16 ***
UTM_N       -4.50e-04   9.24e-05   -4.88  2.8e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.11 on 144 degrees of freedom
```

```
Multiple R-squared: 0.506,        Adjusted R-squared: 0.499
F-statistic: 73.7 on 2 and 144 DF,  p-value: <2e-16
```

Q1 : *What is the equation which predicts subsoil clay content from the coördinates?* *Jump to A1* •

Q2 : *How much of the variation in clay content is explained by this regional trend?* *Jump to A2* •

After fitting a model, the next step is always to see its **diagnostics**, to see how well it fits the assumptions underlying the modelling.

The key information in model diagnostics are the **residuals**, i.e. the **observed** (actual) value less the **fitted** (predicted) value.

Task 4 : Compute the residual for the first observation directly, and compare it with the recorded residual in the model summary. •

We use the `fitted` access method to extract the value predicted by the fitted model, and the `residuals` access method to extract the residuals.

```
> tcp$clay35[1]

[1] 78

> fitted(ts1)[1]

     1
60.087

> tcp$clay35[1] - fitted(ts1)[1]

     1
17.913

> residuals(ts1)[1]

     1
17.913
```

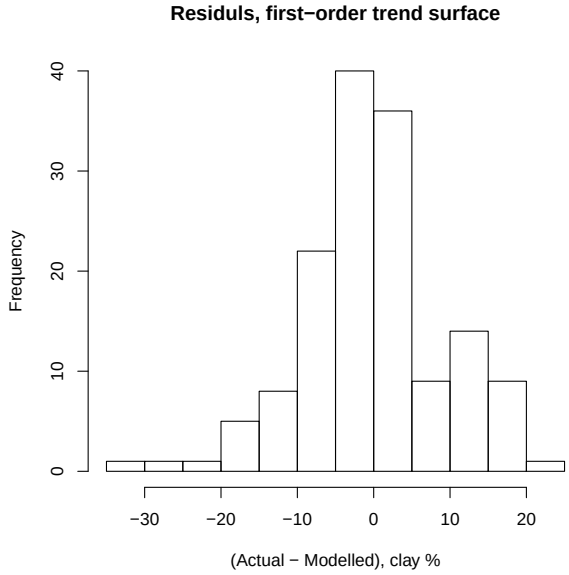
Q3 : *What are the actual and modelled values of clay content for the first observation? What is the residual? Does the model over- or under-predict the actual value?* *Jump to A3* •

First we examine the performance of the model in **feature** (attribute) space, ignoring for the moment that these are geographic data.

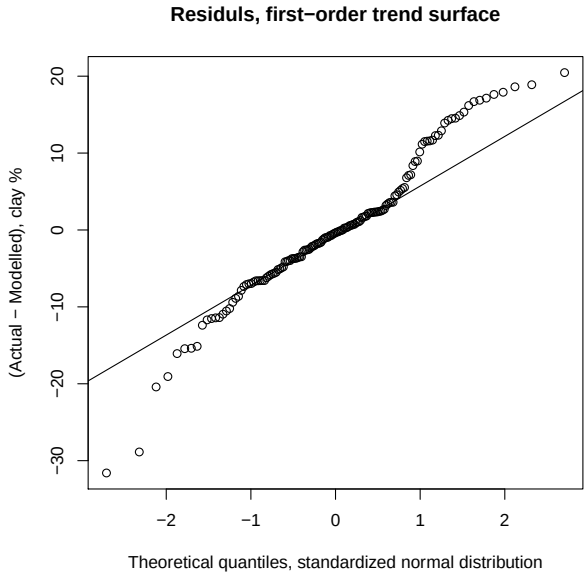
The residuals should be approximately **normally-distributed** to fulfill the assumptions of the linear model.

Task 5 : Display a histogram and normal quantile plot of the residuals. •

```
> hist(residuals(ts1), main = "Residuls, first-order trend surface",
+     xlab = "(Actual - Modelled), clay %")
```



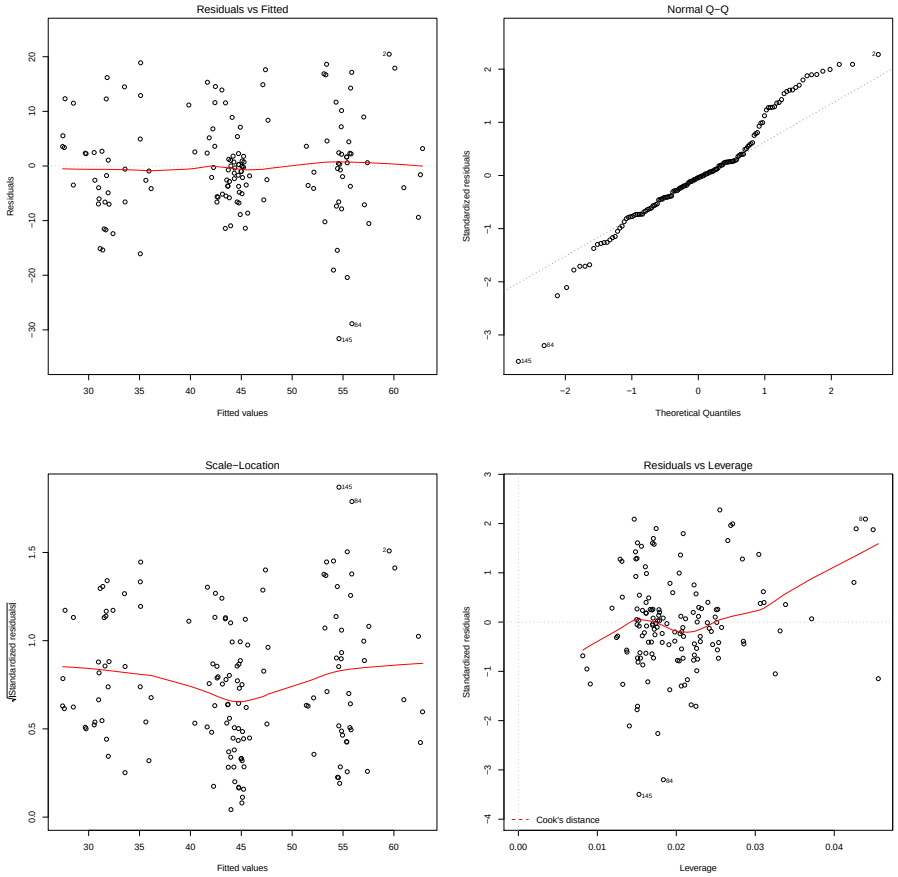

```
> qqnorm(residuals(ts1), main = "Residuls, first-order trend surface",
+     xlab = "Theoretical quantiles, standardized normal distribution",
+     ylab = "(Actual - Modelled), clay %")
> qqline(residuals(ts1))
```



Q4 : *Do the residuals appear to be approximately normally-distributed?*
Jump to A4 •

As in Exercise 1, we can see four diagnostic plots, including the QQ-plot of the residuals, in one figure:

```
> par(mfrow = c(2, 2))
> plot(ts1)
> par(mfrow = c(1, 1))
```



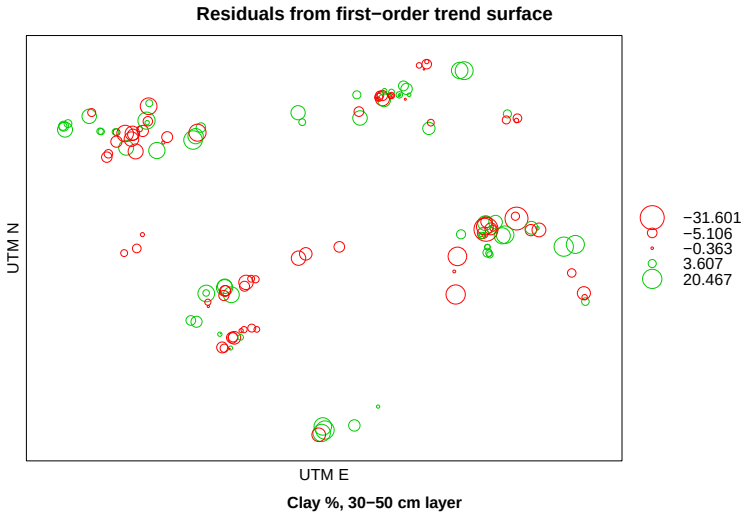
Q5 : *Which two points are quite poorly-fit by the plane?* *Jump to A5* •

Now we examine the residuals in **geographic** space.

Task 6 : Display a post-plot of the residuals. •

The `bubble` method of the `sp` package does just this; but it requires a spatial object. We use the `pch` (printing character) optional graphics argument; `bubble` defaults to `pch=20` (a filled circle); since we have points so close together we specify `pch=1` (an open circle) to see overlapping circles more clearly:

```
> tmp <- data.frame(e = tcp$UTM_E, n = tcp$UTM_N, residuals = residuals(ts1))
> coordinates(tmp) <- c("e", "n")
> print(bubble(tmp, zcol = "residuals", main = "Residuals from first-order trend surface",
+     sub = "Clay %, 30-50 cm layer", xlab = "UTM E", ylab = "UTM N",
+     pch = 1))
```



Q6 : *Do the positive and negative residuals appear to be evenly distributed in geographic space? Does there appear to be any pattern to them?* *Jump to A6* •

2.2 Fitting a second-order surface

We saw that the first-order surface only explains half of the variation; perhaps a second-order surface is better.

Task 7 : Compute a full second-order trend surface. •

We follow the sequence just above:

```
> ts2 <- lm(clay35 ~ I(UTM_E^2) + I(UTM_N^2) + I(UTM_E *
+     UTM_N) + UTM_E + UTM_N, data = tcp)
> summary(ts2)

Call:
lm(formula = clay35 ~ I(UTM_E^2) + I(UTM_N^2) + I(UTM_E * UTM_N) +
    UTM_E + UTM_N, data = tcp)

Residuals:
   Min     1Q Median     3Q    Max
-29.48  -5.00  -0.56   4.21  20.83

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     1.18e+04   6.74e+03    1.75   0.0822 .
I(UTM_E^2)      1.27e-08   8.44e-09    1.50   0.1349
```

```
I(UTM_N^2)         3.58e-08   1.26e-08    2.83   0.0053 **
I(UTM_E * UTM_N)   1.02e-08   1.33e-08    0.77   0.4425
UTM_E             -2.00e-02   1.48e-02   -1.35   0.1777
UTM_N             -3.10e-02   1.24e-02   -2.50   0.0135 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.94 on 141 degrees of freedom
Multiple R-squared: 0.534,        Adjusted R-squared: 0.518
F-statistic: 32.3 on 5 and 141 DF,  p-value: <2e-16
```

Q7 : *How much of the variation in clay content is explained by the full second-order regional trend?* *Jump to A7* •

Q8 : *Are all terms in the regression equation significantly different from zero?* *Jump to A8* •

2.3 Stepwise regression to simplify a higher-order surface

Clearly the equation can be simplified without loss of fit. We could examine nested models one-by-one (e.g. removing the interaction term first), but a more efficient way is to use **stepwise regression** on the full model, with the `step` method. This discards one term at a time until the model is most **parsimonious** as measured by the Akaike Information Criterion (AIC); in general this will also result in the highest **adjusted** R^2.

Task 8 : Simplify the second-order trend surface with stepwise regression. •

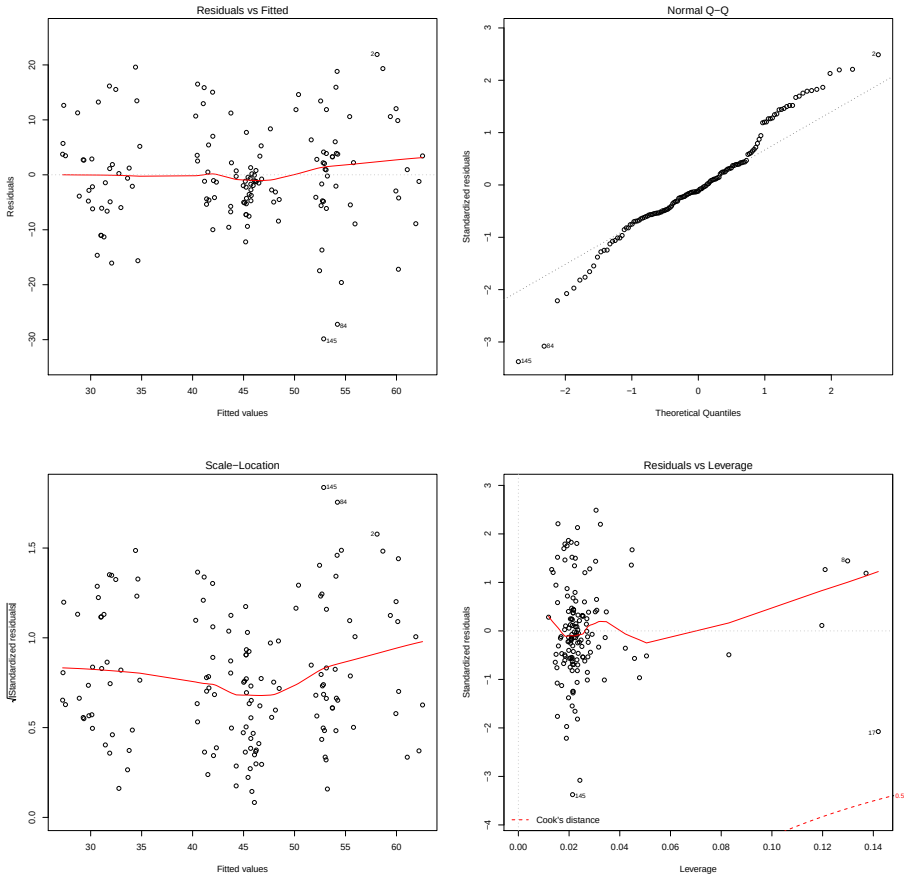
```
> ts2.step <- step(ts2)

Start:  AIC=649.77
clay35 ~ I(UTM_E^2) + I(UTM_N^2) + I(UTM_E * UTM_N) + UTM_E +
    UTM_N

                   Df Sum of Sq   RSS AIC
- I(UTM_E * UTM_N)  1        47 11307 648
- UTM_E             1       147 11406 650
<none>                          11260 650
- I(UTM_E^2)        1       181 11440 650
- UTM_N             1       500 11760 654
- I(UTM_N^2)        1       641 11901 656

Step:  AIC=648.39
clay35 ~ I(UTM_E^2) + I(UTM_N^2) + UTM_E + UTM_N

             Df Sum of Sq   RSS AIC
- UTM_E       1       124 11431 648
- I(UTM_E^2)  1       140 11447 648
<none>                    11307 648
- I(UTM_N^2)  1       636 11943 654
- UTM_N       1       661 11968 655
```

```
Step:  AIC=647.99
clay35 ~ I(UTM_E^2) + I(UTM_N^2) + UTM_N

             Df Sum of Sq   RSS AIC
<none>                     11431 648
- I(UTM_N^2)  1       512 11943 652
- UTM_N       1       537 11968 653
- I(UTM_E^2)  1     10227 21658 740

> summary(ts2.step)

Call:
lm(formula = clay35 ~ I(UTM_E^2) + I(UTM_N^2) + UTM_N, data = tcp)

Residuals:
   Min     1Q Median     3Q    Max
-29.86  -4.87  -1.05   3.82  21.90

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.06e+03   1.22e+03    2.51    0.013 *
I(UTM_E^2)   4.89e-10   4.32e-11   11.31   <2e-16 ***
I(UTM_N^2)   2.86e-08   1.13e-08    2.53    0.012 *
UTM_N       -1.93e-02   7.44e-03   -2.59    0.011 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.94 on 143 degrees of freedom
Multiple R-squared: 0.527,        Adjusted R-squared: 0.517
F-statistic: 53.1 on 3 and 143 DF,  p-value: <2e-16
```

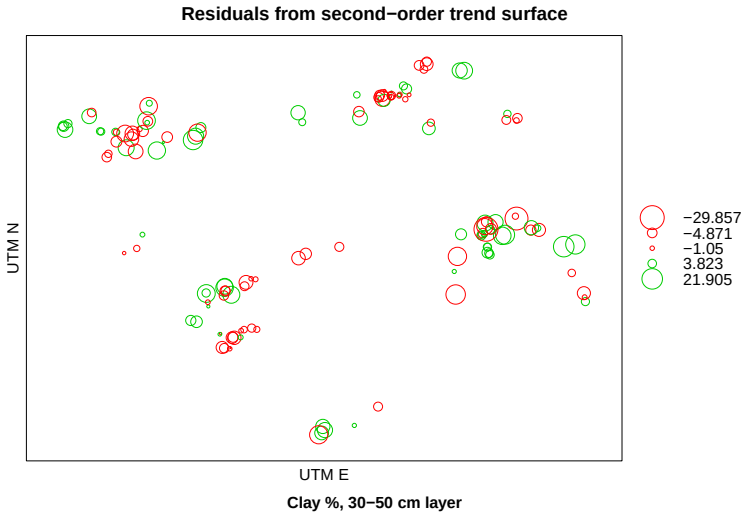
Q9 : *How many equations did* `step` *examine? What terms are left in the final equation? How much of the variation does it explain?* *Jump to A9* •

Task 9 : Plot the regression diagnostics for the fitted second-order surface, in both feature and geographic space. •

```
> par(mfrow = c(2, 2))
> plot(ts2.step)
> par(mfrow = c(1, 1))
```




```
> tmp <- data.frame(e = tcp$UTM_E, n = tcp$UTM_N, residuals = residuals(ts2.step))
> coordinates(tmp) <- c("e", "n")
> print(bubble(tmp, zcol = "residuals", main = "Residuals from second-order trend surface",
+     sub = "Clay %, 30-50 cm layer", xlab = "UTM E", ylab = "UTM N",
+     pch = 1))
```



2.4 * Comparing first- and higher-order surfaces

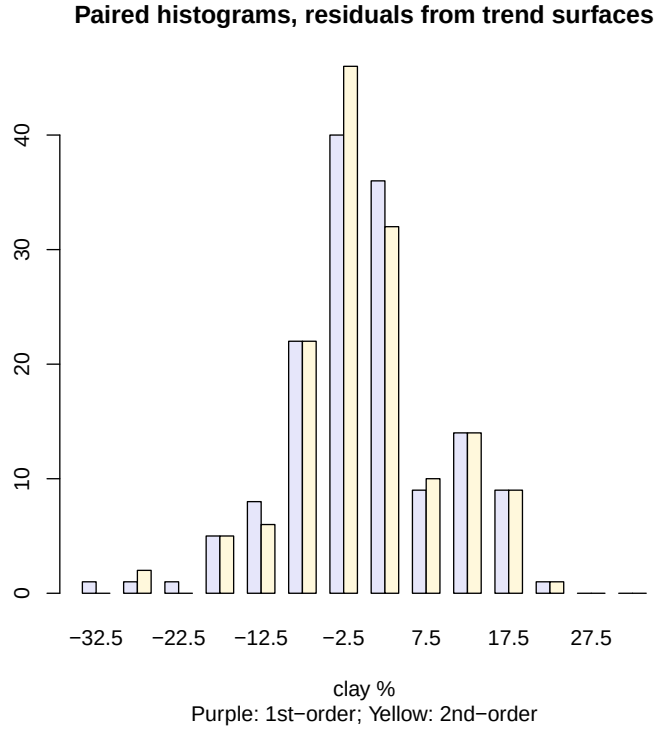
Task 10 : Compare the diagnostics from the second- and first-order surfaces. •

Q10 : *Is there much improvement with the higher-order surface? Consider: (1) goodness-of-fit; (2) range of residuals; (3) normality of residuals; (4) geographic distribution of residuals.* *Jump to A10* •

We compare the residuals with the `summary` method, show their range with the `diff` method, and display the two histograms side-by-side with the `barplot` method. For the latter, we compute the two histograms with the `hist` method, specifying `plot` as `FALSE`, the same break points, and saving the results; we then build a matrix as required by `barplot` wit the `matrix` method:

```
> summary(residuals(ts1))

     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-3.16e+01 -5.11e+00 -3.63e-01  4.62e-16  3.61e+00  2.05e+01

> summary(residuals(ts2.step))

     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-2.99e+01 -4.87e+00 -1.05e+00 -1.35e-16  3.82e+00  2.19e+01

> diff(range(residuals(ts1)))
```

```
[1] 52.067

> diff(range(residuals(ts2.step)))

[1] 51.762

> h1 <- hist(residuals(ts1), breaks = seq(-35, +35, 5),
+     plot = F)
> str(h1)

List of 7
 $ breaks     : num [1:15] -35 -30 -25 -20 -15 -10 -5 0 5 10 ...
 $ counts     : int [1:14] 1 1 1 5 8 22 40 36 9 14 ...
 $ intensities: num [1:14] 0.00136 0.00136 0.00136 0.0068 0.01088 ...
 $ density    : num [1:14] 0.00136 0.00136 0.00136 0.0068 0.01088 ...
 $ mids       : num [1:14] -32.5 -27.5 -22.5 -17.5 -12.5 -7.5 -2.5 2.5 7.5 12.5 ...
 $ xname      : chr "residuals(ts1)"
 $ equidist   : logi TRUE
 - attr(*, "class")= chr "histogram"

> h2 <- hist(residuals(ts2.step), breaks = seq(-35, +35,
+     5), plot = F)
> tmp <- matrix(rbind(h1$counts, h2$counts), nrow = 2)
> barplot(tmp, beside = T, col = c("lavender", "cornsilk"),
+     main = "Paired histograms, residuals from trend surfaces",
+     xlab = "clay %", names.arg = h1$mids, sub = "Purple: 1st-order; Yellow: 2nd-order")
```



Task 11 : Compare the regression residuals from the reduced second- and first-order surfaces in geographic space. •

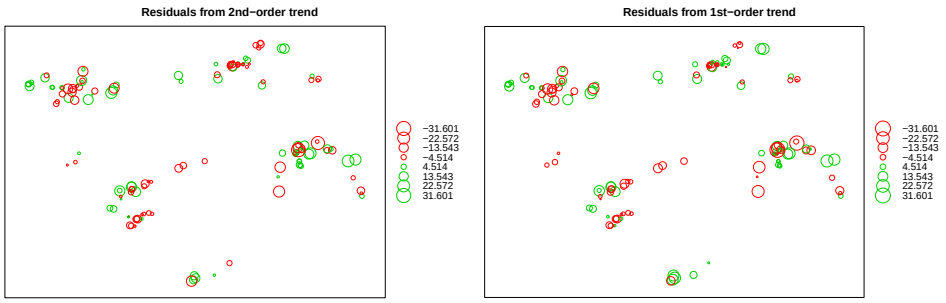
To do this, we make side-by-side post-plots on a common scale. We first

make a spatial points dataframe with coöordinates and the residuals from the first-order surface, second-order surface, and the difference:

```
> tmp <- data.frame(e = tcp$UTM_E, n = tcp$UTM_N, resid1 = residuals(ts1))
> coordinates(tmp) <- c("e", "n")
> tmp$resid2 <- residuals(ts2.step)
> tmp$resid.diff <- tmp$resid2 - tmp$resid1
> str(tmp)

Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
  ..@ data       :'data.frame':        147 obs. of  3 variables:
  .. ..$ resid1    : num [1:147] 17.91 20.47 3.19 -1.6 -9.41 ...
  .. ..$ resid2    : num [1:147] 19.34 21.9 3.44 -1.21 -8.9 ...
  .. ..$ resid.diff: num [1:147] 1.422 1.438 0.257 0.395 0.513 ...
  ..@ coords.nrs : int [1:2] 1 2
  ..@ coords     : num [1:147, 1:2] 702638 701659 703488 703421 703358 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:2] "e" "n"
  ..@ bbox       : num [1:2, 1:2] 659401 310897 703488 342379
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "e" "n"
  .. .. ..$ : chr [1:2] "min" "max"
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
  .. .. ..@ projargs: chr NA
```

Now plot them on the same scale; note the use of the range function to evaluate the range of a list of vectors, the abs function to compute the absolute value of the extremes, and of course the `max` function to obtain the maximum of a vector.

```
> extreme <- max(abs(range(tmp$resid1, tmp$resid2)))
> print(bubble(tmp, zcol = "resid2", main = "Residuals from 2nd-order trend",
+     pch = 1, maxsize = 3 * max(abs(range(tmp$resid2)))/extreme,
+     key.entries = seq(-extreme, extreme, length = 8)),
+     split = c(1, 1, 2, 1), more = T)
> print(bubble(tmp, zcol = "resid1", main = "Residuals from 1st-order trend",
+     pch = 1, maxsize = 3 * max(abs(range(tmp$resid1)))/extreme,
+     key.entries = seq(-extreme, extreme, length = 8)),
+     split = c(2, 1, 2, 1), more = F)
> rm(extreme)
```



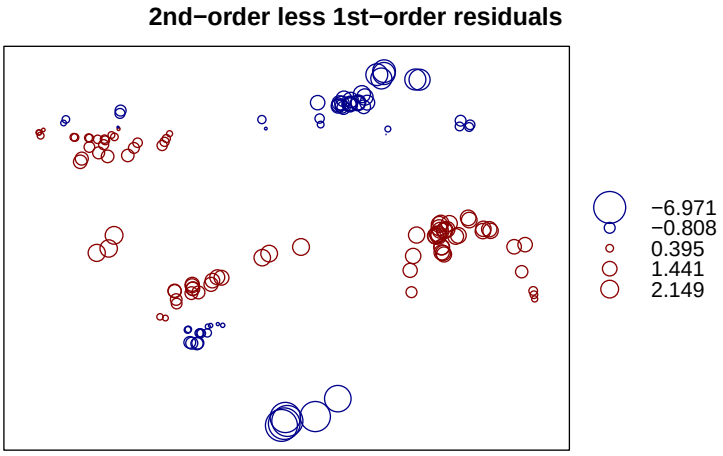
Note: As explained in Exercise 2, the `print` method here is actually the `print.trellis` method, i.e. a specialization of the generic `print` method, which is automatically called by `print` when the argument (here, the results of the `bubble` function of the `lattice` package) is a `trellis` object.

The `print.trellis` method is the default print method for objects of class `trellis` which are produced by calls to `lattice` functions like `xyplot` or `bubble`; the `spplot` method of the `sp` package also uses lattice graphics and hence `print.trellis`. In interactive use it is called automatically when a trellis object is produced. However, here we call it explicitly in order to specify the `split` optional argument, which specifies the layout of multiple trellis graphs on one page.

In this example `split=c(1,1,2,1)` means: place this trellis plot at column 1 (x-position), row 1 (y-position) in an array of 2 columns and 1 row. The optional `more` argument tells whether more graphs are to follow or not. For more details see `help(print.trellis)`.

And we can plot the difference of differences, i.e. how much the fit changed at each point:

```
> print(bubble(tmp, zcol = "resid.diff", main = "2nd-order less 1st-order residuals",
+     pch = 1, col = c("blue4", "red4")))
```



Q11 : *Where are the largest changes (both positive and negative) in the trend surface, between the two fits? Explain this spatial pattern of changes.*
Jump to A11 •

Task 12 : Clean up the workspace from this section. •

```
> rm(ts1, ts2, ts2.step, h1, h2, tmp)
```

2.5 * Linear models on sp objects

Linear modelling can also be done on coördinates of `sp` objects. In this **optional** subsection, we repeat much of the above analysis, first converting the `tcp` dataset to a spatial (`sp`) object, as we did in Exercise 2 §6. The purpose of this optional subsection is to show the syntax of the R code; no new ideas are introduced. Steps from §2 that are not repeated here have identical code in both cases.

Task 13 : Convert the `tcp` data frame to a spatial object. •

```
> coordinates(tcp) <- c("UTM_E", "UTM_N")
> str(tcp)

Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
  ..@ data       :'data.frame':        147 obs. of  2 variables:
  .. ..$ clay35: num [1:147] 78 80 66 61 53 57 70 72 70 62 ...
  .. ..$ pH35  : num [1:147] 4.8 4.4 4.2 4.54 4.4 ...
  ..@ coords.nrs : int [1:2] 1 2
  ..@ coords     : num [1:147, 1:2] 702638 701659 703488 703421 703358 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:2] "UTM_E" "UTM_N"
  ..@ bbox       : num [1:2, 1:2] 659401 310897 703488 342379
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "UTM_E" "UTM_N"
  .. .. ..$ : chr [1:2] "min" "max"
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
  .. .. ..@ projargs: chr NA
```

Task 14 : Compute the first-order trend surface of subsoil clay content, modelled by the two metric coördinates, using ordinary least squares and summarize the model. •

The right-hand side of the formula operator is simply the `coordinates` extractor; there are two coördinates and they are assumed to be linear additive factors in the `lm` model:

```
> ts1 <- lm(clay35 ~ coordinates(tcp), data = tcp)
> summary(ts1)

Call:
lm(formula = clay35 ~ coordinates(tcp), data = tcp)

Residuals:
    Min      1Q  Median      3Q     Max
-31.601  -5.106  -0.363   3.607  20.467

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)           -2.50e+02   5.19e+01   -4.83  3.5e-06 ***
coordinates(tcp)UTM_E  6.51e-04   5.97e-05   10.91  < 2e-16 ***
coordinates(tcp)UTM_N -4.50e-04   9.24e-05   -4.88  2.8e-06 ***
```

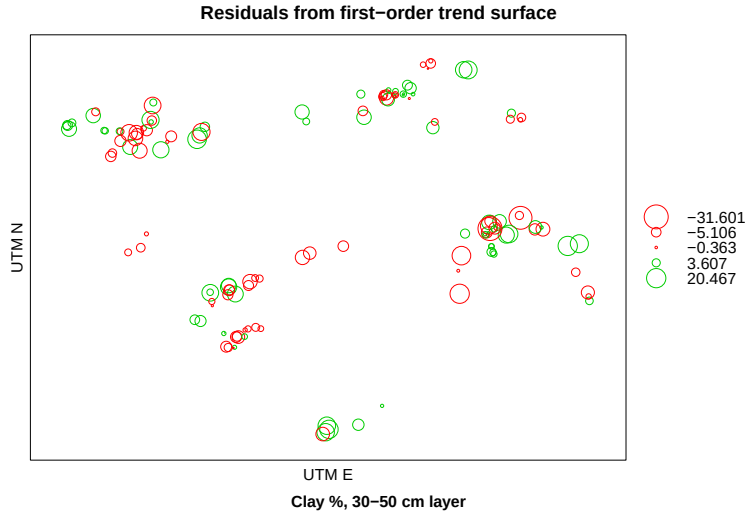
```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.11 on 144 degrees of freedom
Multiple R-squared: 0.506,        Adjusted R-squared: 0.499
F-statistic: 73.7 on 2 and 144 DF,  p-value: <2e-16
```

The results of this model are identical to the one where we specified the coördinates directly.

Task 15 : Display a post-plot of the residuals. •

The `bubble` method of the `sp` package requires a spatial object.

```
> tmp <- data.frame(coordinates(tcp), residuals = residuals(ts1))
> coordinates(tmp) <- c("UTM_E", "UTM_N")
> print(bubble(tmp, zcol = "residuals", main = "Residuals from first-order trend surface",
+     sub = "Clay %, 30-50 cm layer", xlab = "UTM E", ylab = "UTM N",
+     pch = 1))
```



Task 16 : Compute a full second-order trend surface. •

To specify the square of the coördinates, we first extract them with the `coordinates` method, then square them within the `I` "as is" method, so that the `^2` is *not* interpreted as a formula operator. The cross-product of the coördinates is tricker; each coördinate must be extracted separately, then they can be multiplied.

```
> ts2 <- lm(clay35 ~ I(coordinates(tcp)^2) + I(coordinates(tcp)[,
+     1] * coordinates(tcp)[, 2]) + coordinates(tcp), data = tcp)
> summary(ts2)

Call:
lm(formula = clay35 ~ I(coordinates(tcp)^2) + I(coordinates(tcp)[,
    1] * coordinates(tcp)[, 2]) + coordinates(tcp), data = tcp)

Residuals:
   Min     1Q Median     3Q    Max
-29.48  -5.00  -0.56   4.21  20.83

Coefficients:
                                              Estimate Std. Error
(Intercept)                                   1.18e+04   6.74e+03
I(coordinates(tcp)^2)UTM_E                    1.27e-08   8.44e-09
I(coordinates(tcp)^2)UTM_N                    3.58e-08   1.26e-08
I(coordinates(tcp)[, 1] * coordinates(tcp)[, 2])  1.02e-08   1.33e-08
coordinates(tcp)UTM_E                        -2.00e-02   1.48e-02
coordinates(tcp)UTM_N                        -3.10e-02   1.24e-02
                                              t value Pr(>|t|)
(Intercept)                                      1.75   0.0822 .
I(coordinates(tcp)^2)UTM_E                       1.50   0.1349
I(coordinates(tcp)^2)UTM_N                       2.83   0.0053 **
I(coordinates(tcp)[, 1] * coordinates(tcp)[, 2])    0.77   0.4425
coordinates(tcp)UTM_E                           -1.35   0.1777
coordinates(tcp)UTM_N                           -2.50   0.0135 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.94 on 141 degrees of freedom
Multiple R-squared: 0.534,        Adjusted R-squared: 0.518
F-statistic: 32.3 on 5 and 141 DF,  p-value: <2e-16
```

The results of this model are identical to the one where we specified the coördinates directly.

Task 17 : Simplify the second-order trend surface with stepwise regression. •

```
> ts2.step <- step(ts2)

Start:  AIC=649.77
clay35 ~ I(coordinates(tcp)^2) + I(coordinates(tcp)[, 1] * coordinates(tcp)[,
    2]) + coordinates(tcp)

                                              Df Sum of Sq   RSS
- I(coordinates(tcp)[, 1] * coordinates(tcp)[, 2])  1        47 11307
<none>                                                     11260
- coordinates(tcp)                             2       570 11829
- I(coordinates(tcp)^2)                        2       666 11926
                                              AIC
- I(coordinates(tcp)[, 1] * coordinates(tcp)[, 2]) 648
<none>                                         650
- coordinates(tcp)                             653
- I(coordinates(tcp)^2)                        654

Step:  AIC=648.39
```

```
clay35 ~ I(coordinates(tcp)^2) + coordinates(tcp)

                        Df Sum of Sq   RSS AIC
<none>                                11307 648
- I(coordinates(tcp)^2)  2       637 11944 652
- coordinates(tcp)       2       661 11968 653

> summary(ts2.step)

Call:
lm(formula = clay35 ~ I(coordinates(tcp)^2) + coordinates(tcp),
    data = tcp)

Residuals:
    Min      1Q  Median      3Q     Max
-29.725  -4.762  -0.222   3.912  20.190

Coefficients:
                             Estimate Std. Error t value Pr(>|t|)
(Intercept)                  7.49e+03   3.76e+03    2.00   0.0479 *
I(coordinates(tcp)^2)UTM_E   8.41e-09   6.34e-09    1.33   0.1870
I(coordinates(tcp)^2)UTM_N   3.57e-08   1.26e-08    2.83   0.0054 **
coordinates(tcp)UTM_E       -1.08e-02   8.64e-03   -1.25   0.2138
coordinates(tcp)UTM_N       -2.39e-02   8.30e-03   -2.88   0.0046 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.92 on 142 degrees of freedom
Multiple R-squared: 0.532,        Adjusted R-squared: 0.519
F-statistic: 40.4 on 4 and 142 DF,  p-value: <2e-16
```

Here we do *not* get the same result as when using the coördinates as separate variables. This is because the two coördinates are here specified together in both the linear and quadratic terms; and they must be either both or neither in the equation. So the predictor linear `UTM_E` can not, by itself, be removed, as previously.

2.6 Answers

A1 : Clay $= -250 + 0.000651 \cdot \text{UTM_E} - 0.00045 \cdot \text{UTM_N}$ *Return to Q1* •

A2 : 49.9% *(see adjusted* R^2*).* *Return to Q2* •

A3 : *Actual:* 78% *clay; modelled:* 60.1% *clay; residual* $78 - 60.1 = 17.9\%$ *clay. The model* ***under-predicts*** *this point.* *Return to Q3* •

A4 : *No; the histogram shows left-skew (longer negative than positive tail); the QQ-plot shows that both tails (beyond* -1 *and* $+1$ *standardized residuals) are "too extreme". That is, the negative tail is below the expected residuals, so that there are residuals from* -20 *to* -30*, none of which would be expected if the residuals are from a normal distribution; similarly the positive tail is "too positive", above*

the line. *Return to Q4* •

A5 : *Points 84 and 145 have especially high negative residuals, i.e. the predictions are far too high.* *Return to Q5* •

A6 : *The residuals seem to be fairly evenly distributed. Large negative and positive residuals occur near to each other in no apparent pattern. This suggests that there would be little benefit in attempting to model a higher-order surface.* *Return to Q6* •

A7 : *51.8% (see adjusted R^2).* *Return to Q7* •

A8 : *No, only the quadratic N term* `I(UTM_N^2)` *($p = 0.0053$), the linear N term* `UTM_N` *($p = 0.0135$), and the intercept ($p = 0.0822$) are significant at $p < 0.1$. The interaction term* `I(UTM_N * UTM_E)` *is almost surely not significant.* *Return to Q8* •

A9 : *It examined three equations: (1) the full second-order surface (AIC = 649.77), (2) with the interaction term removed (AIC = 648.39), (3) with the linear E coördinate also removed (AIC= 647.99). The simplified model fits 51.7% of the variation. Although this is slightly less than the full model (51.9%) the equation is more parsimonious (two fewer terms).* *Return to Q9*
•

A10 : *(1) Slightly better goodness-of-fit (adjusted R^2 = 51.8% (see adjusted R^2). vs. 49.9%).); (2) slightly narrower range of residuals; however the maximum residual is larger; (3) slightly closer fit to the QQ-plot; (4) the residuals are somewhat more evenly distributed in geographic space.*

In summary, there is not much improvement over the first-order trend surface. *Return to Q10* •

A11 : *The largest positive changes (second-order residual larger than first-order) is on a NNW-SSE axis through the middle of the map. These are never larger than about 2% clay; the second-order is a worse fit. However, there are some large negative changes, up to almost -7% (first-order residual larger than second-order); these are at the N and S, where the second-order could adjust (by curvature) to these values that did not fit the first-order plane, dipping NNW.* *Return to Q11* •

3 Calibration and validation datasets

For the rest of the exercise we continue with the Jura soil samples introduced in Exercise 2, §2. We saved this as an R object in Exercise 2, §3.2.

Task 18 : If the `jura.all` spatial object is not already in the workspace, load it from the saved image. •

```
> load("Jura.RData")
```

In geostatistical modelling based on random fields we make **assumptions** about the nature of the spatial process; we then fit the assumed model. However since there is only one **realization** of the random field, i.e. only one reality, we need to find some way to **validate** the assumptions.

A common approach is to **split the dataset** into two:

Calibration : to model the spatial structure;

Validation : to validate the model

A typical split is approximately 2/3 calibration, 1/3 validation. The Jura data set contains 359 observations; in previous studies with this same dataset the last 100 of these have been held out for validation, while the first 259 have been used for calibration. The dataset has been organized in this way by Goovaerts [3], from whom we obtained it.

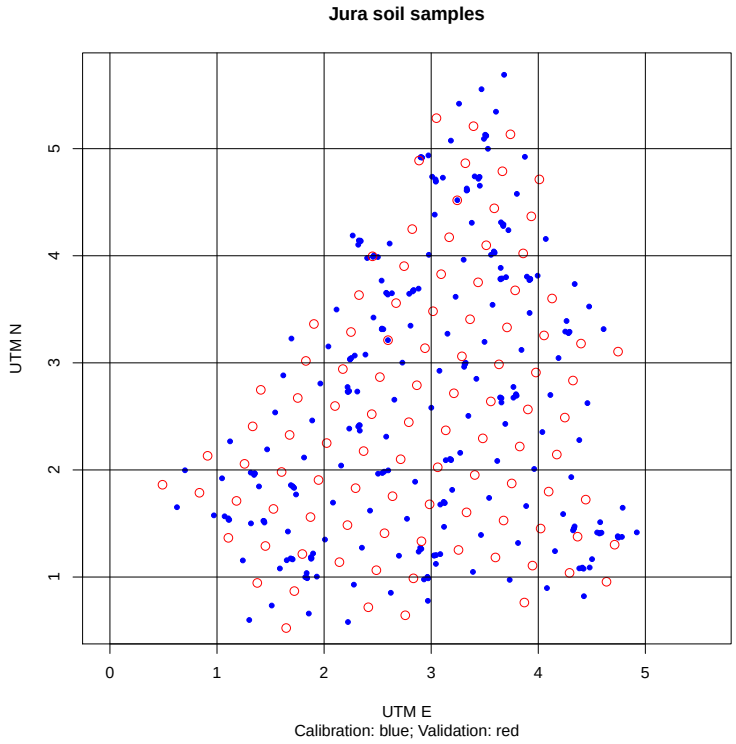
Task 19 : Divide the Jura dataset into a **calibration** set `jura.cal`, made up of the first 259 observations from the full set, and a **validation** set `jura.val`, made up of the remaining 100 observations from the full set. •

This is just row selection from a matrix using the `[]` operator, which also works with the `@data` slot `sp` object:

```
> jura.cal <- jura.all[1:259, ]
> jura.val <- jura.all[260:359, ]
```

Task 20 : Plot the locations of the two sets, showing them with different colours. •

```
> plot(coordinates(jura.cal), col = "blue", asp = 1, pch = 20,
+     xlab = "UTM E", ylab = "UTM N", main = "Jura soil samples",
+     sub = "Calibration: blue; Validation: red")
> points(coordinates(jura.val), col = "red", pch = 1, cex = 1.2)
> grid(lty = 1, col = "black")
```



Q12 : *What is the spatial distribution of the calibration and validation points? Do they equally represent the study area?* *Jump to A12* •

3.1 * Random selection of validation points

This **optional** subsection discusses selection of validation points in the case that a dataset is not nicely ordered for this purpose. In the case of the Jura set, the author had already placed the 100 validation samples at the end of the set, gridded and covering the whole area. In general this will not be the case, and we have to draw a **random sample** from the data frame.

Task 21 : Select 100 observations at random from the Jura dataset. •

The first step is to identify which observations will be in the sample.

For this we use the `sample` "random sample" method, which draws a completely random sample with or (by default) without replacement from a vector, here the row number in the data frame. We get the number of rows as the first dimension of the data frame, extracted with the `dim` "dimensions" method.

> **Note:** Of course we know the number of rows from examining the data set, but it is more elegant to compute it.

```
> dim(jura.all@data)

[1] 359   9
```

```
> dim(jura.all@data)[1]

[1] 359

> valid.id <- sample(1:dim(jura.all@data)[1], 100)
> sort(valid.id)

  [1]   2   4   5   9  19  28  33  38  41  44  49  50  57  58  59  66
 [17]  67  76  78  86  87  88  91  93  98 101 107 111 114 117 118 119
 [33] 120 123 127 130 131 133 142 146 148 150 154 155 165 167 172 174
 [49] 180 185 188 189 191 201 205 206 208 211 213 220 225 229 230 231
 [65] 234 235 238 240 243 251 259 262 264 268 273 274 280 286 288 290
 [81] 292 301 305 309 312 313 320 323 325 329 334 335 337 342 343 347
 [97] 350 351 352 358
```

Q13 : *Your list of selected observations will be different; why?* *Jump to A13* •

Now we can extract these observations, by selecting the relevant rows of the data frame. The spatial information is implicitly extracted as well.

```
> jura.val.rand <- jura.all[valid.id, ]
> str(jura.val.rand)

Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
  ..@ data       :'data.frame':        100 obs. of  9 variables:
  .. ..$ Rock: Factor w/ 5 levels "Argovian","Kimmeridgian",..: 2 1 2 3 5 2 3 2 1 1 ...
  .. ..$ Land: Factor w/ 4 levels "Forest","Pasture",..: 3 3 1 3 2 1 3 1 1 3 ...
  .. ..$ Cd  : num [1:100] 0.765 0.394 0.51 0.63 1.31 ...
  .. ..$ Cu  : num [1:100] 42 39.56 5.96 18.04 17.7 ...
  .. ..$ Pb  : num [1:100] 70.8 105.6 62.4 38.2 48.4 ...
  .. ..$ Co  : num [1:100] 13.84 4.44 1.65 12 12.72 ...
  .. ..$ Cr  : num [1:100] 29.72 21.64 3.32 31.28 34.8 ...
  .. ..$ Ni  : num [1:100] 20.96 8.92 1.98 18.68 19.6 ...
  .. ..$ Zn  : num [1:100] 90.8 72.4 60.4 80.8 80.4 ...
  ..@ coords.nrs : int [1:2] 1 2
  ..@ coords     : num [1:100, 1:2] 3.65 2.14 3.9 3.04 4.44 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:2] "X" "Y"
  ..@ bbox       : num [1:2, 1:2] 0.491 0.58 4.742 5.345
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "X" "Y"
  .. .. ..$ : chr [1:2] "min" "max"
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
  .. .. ..@ projargs: chr NA
```

Finally, the calibration set is the complement of the validation set. We determine this with the `is.element` set operator[1]; we select the complement of the set with the `!` "not" logical operator.

```
> jura.cal.rand <- jura.all[!is.element(1:dim(jura.all@data)[1],
+     valid.id), ]
> str(jura.cal.rand)
```

[1] The other set operators are `union`, `intersection`. `setdiff` and `setequal`

```
Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
  ..@ data       :'data.frame':        259 obs. of  9 variables:
  .. ..$ Rock: Factor w/ 5 levels "Argovian","Kimmeridgian",..: 3 3 5 5 1 3 1 1 5 2 ...
  .. ..$ Land: Factor w/ 4 levels "Forest","Pasture",..: 3 2 3 3 3 3 3 1 3 1 ...
  .. ..$ Cd  : num [1:259] 1.74 1.61 1.145 0.894 0.525 ...
  .. ..$ Cu  : num [1:259] 25.72 8.88 31.28 27.44 66.12 ...
  .. ..$ Pb  : num [1:259] 77.4 30.8 72.4 60 141 ...
  .. ..$ Co  : num [1:259] 9.32 10.6 3.5 15.08 4.2 ...
  .. ..$ Cr  : num [1:259] 38.3 47 40.4 30.5 25.4 ...
  .. ..$ Ni  : num [1:259] 21.32 21.4 22.04 21.76 9.92 ...
  .. ..$ Zn  : num [1:259] 92.6 64.8 75.2 72.4 72.1 ...
  ..@ coords.nrs : int [1:2] 1 2
  ..@ coords     : num [1:259, 1:2] 2.39 2.81 3.24 3.92 2.12 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:2] "X" "Y"
  ..@ bbox       : num [1:2, 1:2] 0.836 0.524 4.92 5.69
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "X" "Y"
  .. .. ..$ : chr [1:2] "min" "max"
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
  .. .. ..@ projargs: chr NA
```

Q14 : *Do the two sets cover the same area?* *Jump to A14* •

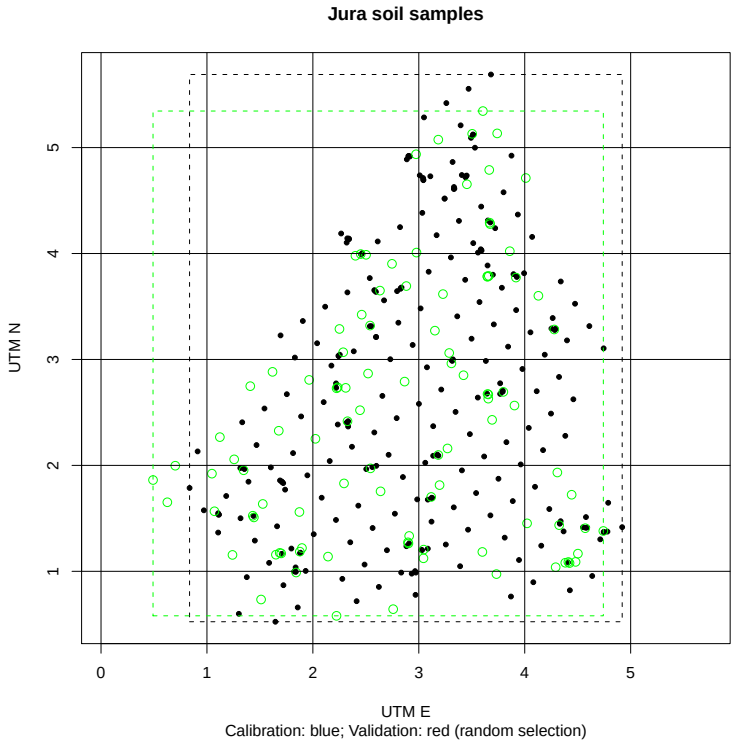
We can answer this with the `bbox` "bounding box" spatial method, and by visual evaluation. To visualize the bounding box, we write a small function with the `function` method. This plots the bounding box, using the `lines` base graphics method, after extracting the corners of the bounding box with `bbox`:

```
> plot.bbox <- function(sp.object, colour = "black", line.type = 2) {
+     b <- bbox(sp.object)
+     lines(c(b["X", "min"], b["X", "max"], b["X", "max"],
+         b["X", "min"], b["X", "min"]), c(b["Y", "min"],
+         b["Y", "min"], b["Y", "max"], b["Y", "max"],
+         b["Y", "min"]), col = colour, lty = line.type)
+ }
```

Now we plot the points and use the function to plot the bounding boxes:

```
> bbox(jura.val.rand)

    min   max
X 0.491 4.742
Y 0.580 5.345

> bbox(jura.cal.rand)

    min  max
X 0.836 4.92
Y 0.524 5.69

> plot(coordinates(jura.cal.rand), col = "black", asp = 1,
+     pch = 20, xlab = "UTM E", ylab = "UTM N", main = "Jura soil samples",
+     sub = "Calibration: blue; Validation: red (random selection)")
> points(coordinates(jura.val.rand), col = "green", pch = 1,
+     cex = 1.2)
```

```
> grid(lty = 1, col = "black")
> plot.bbox(jura.cal.rand)
> plot.bbox(jura.val.rand, colour = "green")
```



For the remainder of the exercise we will use the gridded validation sample; this section was just to illustrate how a random selection can be made.

Task 22 : Remove the temporary objects from the workspace. •

```
> rm(valid.id, jura.cal.rand, jura.val.rand, plot.bbox)
```

3.2 Answers

A12 : *The validation points appear to be on a regular grid; the calibration points are at alternate points on the grid but also have some clusters; this is good because it helps estimate the variogram. Both sets cover the area.* *Return to Q12* •

A13 : *It is a random sample.* *Return to Q13* •

A14 : *This depends on the random sample, but in general the validation set will be contained in a somewhat smaller bounding box. They both cover the area; because of the random selection of the smaller validation set it may be some spatial clusters and holes.* *Return to Q14* •

4 Fitting variogram models

In the previous exercise we learned how to compute **empirical variograms** from observations. There we used them to qualitatively evaluate local spatial dependence. Now we are in a position to **model** this dependence, i.e. fit a theoretical variogram model to the empirical variogram.

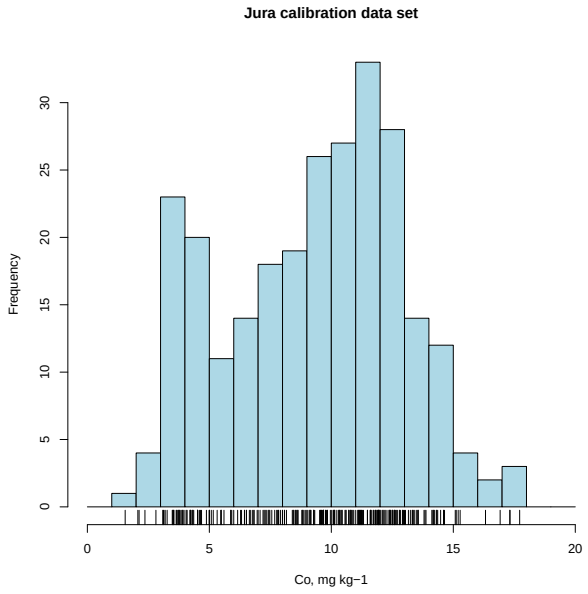
We now repeat the computation of the empirical variogram from Exercise 2, §7.2, but with the calibration set only.

There are seven metals which were measured; we begin with one that is fairly simple to model, namely Cobalt (Co).

Task 23 : Evaluate the feature-space distribution of Co in the calibration dataset and decide if it needs to be transformed prior to variogram analysis. •

The `hist` method shows a histogram, which can be used to evaluate the distribution:

```
> hist(jura.cal$Co, breaks = 0:20, main = "Jura calibration data set",
+     xlab = "Co, mg kg-1", col = "lightblue")
> rug(jura.cal$Co)
```



Note the use of the `:` operator to specify a sequence of integers from 0 to 20; the `seq` method can be used for more general sequences.

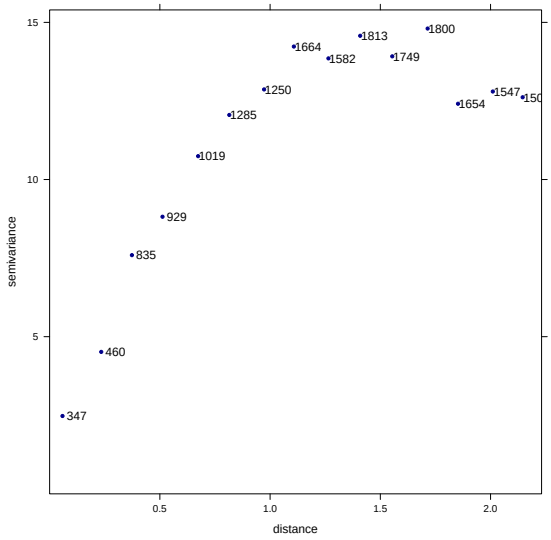
Q15 : *Describe the feature-space distribution of Co in the calibration dataset. Would a transformation make this more symmetrically-distributed?* *Jump to A15* •

So we compute the empirical variogram of the untransformed variable `Co`.

Task 24 : Compute the default empirical variogram of the Co values in the calibration dataset; and plot it. •

We use the `variogram` method of the `gstat` package. Note the use of the `loc` ("location") argument to specify the dataset from which the coördinates should be extracted; by default this is also where the attribute values are to be found.

```
> v <- variogram(Co ~ 1, loc = jura.cal)
> print(plot(v, plot.numbers = T, pch = 20, col = "darkblue"))
```



4.1 Selecting a variogram model

The first question to ask when modelling an empirical variogram is whether it shows the full **range** of local spatial variability, in other words whether it reaches or approaches a **sill**.

Q16 : *Does this variogram approach or reach a sill? Approximately what value? At approximately what range?* *Jump to A16* •

The second question is whether the variogram shows point-pairs with more separation than the local spatial dependence; if so the variogram should be re-computed with a shorter cutoff.

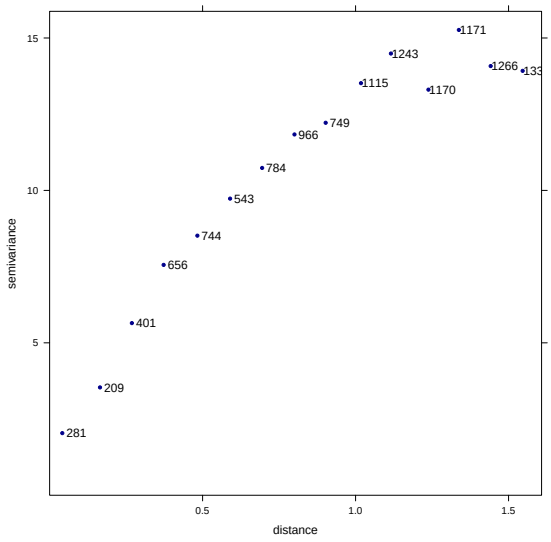
Q17 : *Does this variogram show point-pairs with more separation than the local spatial dependence? I.e. is the range shorter than the maximum separation shown in the variogram? What would be an appropriate maximum?* *Jump to A17* •

Task 25 : Compute the empirical variogram of the Co values with a shorter

cutoff; and plot it. •

The `cutoff` argument to `variogram` is used to specify the maximum separation:

```
> v <- variogram(Co ~ 1, loc = jura.cal, cutoff = 1.6)
> plot(v, plot.numbers = T, pch = 20, col = "darkblue")
```



Note that this variogram also has 15 bins (the default); since the maximum separation is shorter there are fewer point-pairs in each bin.

To select a model, there are several criteria:

1. Which model has been applied to similar data in other studies?
2. Which model is suggested by the assumed spatial process?
3. Which model best fits the form shown on the empirical variogram?

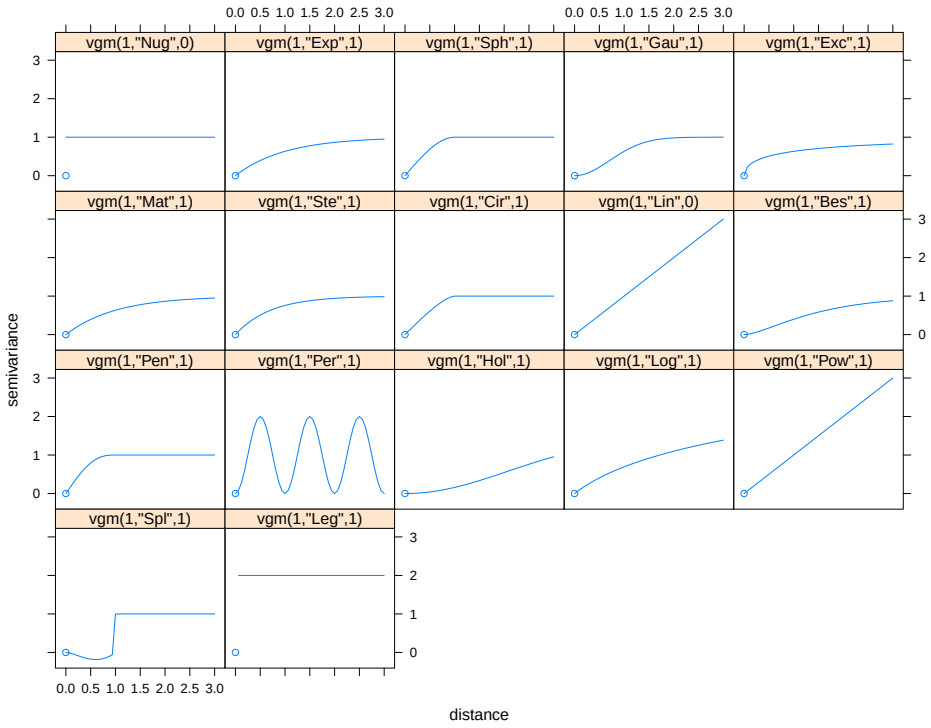
In this example, many studies of trace elements in soils have successfully used a model with a definite sill and transition zone, such as the **spherical**, **circular** or **pentaspherical** models. These are also suggested by any process that generates patches.

To see which model forms match this empirical variogram, we display all the possibilities.

Task 26 : Display all the variogram models available in `gstat`. •

We can see the forms of all the models in `gstat` with the `show.vgms` method:

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



The **model name** as used by `gstat` is shown above each panel; we an see the full names with the `vgm` method with no arguments:

```
> vgm()
```

	short	long
1	Nug	Nug (nugget)
2	Exp	Exp (exponential)
3	Sph	Sph (spherical)
4	Gau	Gau (gaussian)
5	Exc	Exclass (Exponential class)
6	Mat	Mat (Matern)
7	Ste	Mat (Matern, M. Stein's parameterization)
8	Cir	Cir (circular)
9	Lin	Lin (linear)
10	Bes	Bes (bessel)
11	Pen	Pen (pentaspherical)
12	Per	Per (periodic)
13	Hol	Hol (hole)
14	Log	Log (logarithmic)
15	Pow	Pow (power)
16	Spl	Spl (spline)
17	Leg	Leg (Legendre)
18	Err	Err (Measurement error)
19	Int	Int (Intercept)

Note: The mathematical form of these models is given in the `gstat` User's Manual [5] Table 4.1 and in many texts. The `Exc` ("Exponential class") and `Mat` (Matérn) models are *classes* of models that require an additional parameter `kappa` to adjust their shape. The Matérn model can be viewed as a generalization of the Exponential, Power, Logarithmic and Gaussian

models [4]. These model classes require sophisticated techniques to fit them properly, as well as good theoretical understanding of what they represent. The beginning is advised to use the common models that do not have a shape parameter.

Q18 : *Which **model forms** match this empirical variogram?* *Jump to A18* •

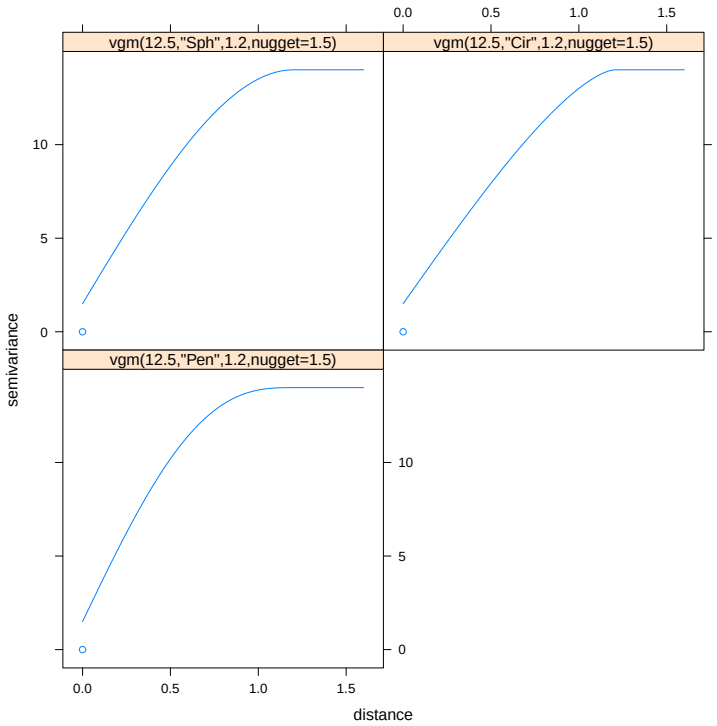
The three possibilities are quite similar, differing only in the shape of the "shoulder" transition to the sill. We can compare them in more detail, with approximate values for range, sill and nugget.

Q19 : *What are approximate values for range, total sill and nugget?* *Jump to A19* •

> **Note:** In `gstat` models the sill is specified as a **partial sill**, that is, the difference between the **total sill** and the **nugget**. This is also known as the **structural** sill.

Task 27 : Compare the shapes of the spherical, circular, and pentaspherical models with the approximate parameters for this empirical variogram. •

```
> print(show.vgms(models = c("Sph", "Cir", "Pen"), sill = 12.5,
+     nugget = 1.5, range = 1.2, max = 1.6))
```



Q20 : *Which of these three best matches the empirical variogram?* *Jump to A20* •

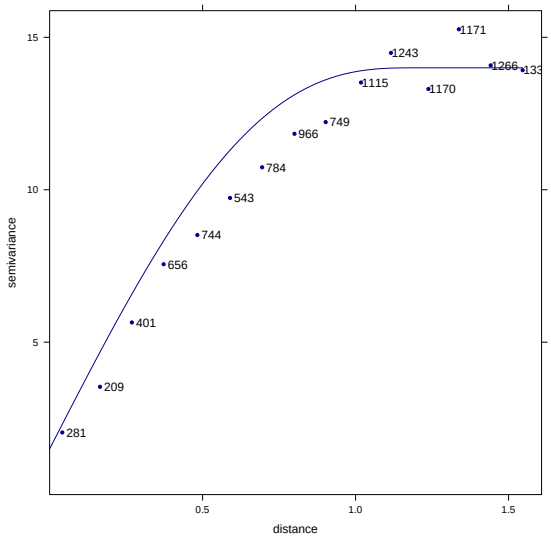
4.2 Fitting a variogram model

Once a model form has been selected, we can adjusted the **model parameters** to best match the empirical variogram.

Task 28 : Plot the empirical variogram with the model, fitted by eye, superimposed. •

For this we use the `vgm` method to create a variogram model object, and then the `model` optional argument to the `plot.gstatVariogram` method; this method is automatically called by R's `plot` method if the object to be plotted is of class `gstatVariogram`.

```
> vm <- vgm(12.5, "Pen", 1.2, 1.5)
> print(vm)

  model psill range
1   Nug   1.5   0.0
2   Pen  12.5   1.2

> class(vm)

[1] "variogramModel" "data.frame"

> print(plot(v, plot.numbers = T, pch = 20, col = "darkblue",
+     model = vm))
```



Q21 : *How well does this model, estimated by eye, fit the empirical variogram?* *Jump to A21*
•

This result is typical; the first "eyeball" estimate is often not so good. There are two strategies:

1. Adjust the range, sill and nugget by eye to get a better fit;
2. Automatically adjust the parameters, e.g. by some form of **weighted least-squares**.

> **Note:** If you wish, you can try to improve the estimate by eye before proceeding to the next step (automatic fitting).

If the variogram is fairly regular, as in this case, the automatic fit usually works well, even starting from an estimate that is not too good.

Task 29 : Fit the variogram model to the empirical variogram, starting from the original estimate; compare the fit to the estimate. •

We use the `fit.variogram` method of `gstat`:

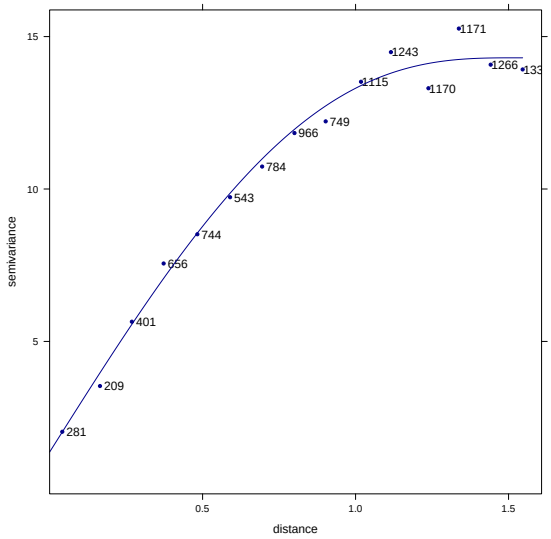
```
> vmf <- fit.variogram(v, vm)
> print(vm)

  model psill range
1   Nug   1.5   0.0
2   Pen  12.5   1.2

> print(vmf)

  model   psill  range
1   Nug  1.3712 0.0000
2   Pen 12.9322 1.5239

> vmf$range - vm$range

[1] 0.0000 0.3239

> vmf$psill - vm$psill

[1] -0.12882  0.43220

> sum(vmf$psill) - sum(vm$psill)

[1] 0.30338
```

> **Note:** The `fit.variogram` method has an optional argument `fit.method`, which specified how the points of the empirical variogram are to be **weighted** for the least-squares fit. The default method `fit.method=7` gives weights proportional to the number of point-pairs and inversely proportional to the separation distance squared; this criterion is not supported by any theory, but has proven sucessful in practice.

Q22 : *How much did the automatic fit change the estimated fit?* *Jump to A22* •

Task 30 : Plot the empirical variogram with the fitted model superimposed. •

```
> print(plot(v, plot.numbers = T, pch = 20, col = "darkblue",
+     model = vmf))
```



Q23 : *How well does the fitted model match the empirical variogram?* *Jump to A23* •

The ratio of the nugget to the total sill gives the proportion of the total variance that is not explained by the variogram model even at the closest separation; its converse is the proportion of total variance explained by the spatial model.

Q24 : *What proportion of the total variance in Co is explained by the fitted variogram model?* *Jump to A24* •

```
> 1 - vmf$psill[1]/sum(vmf$psill)
```

```
[1] 0.90414
```

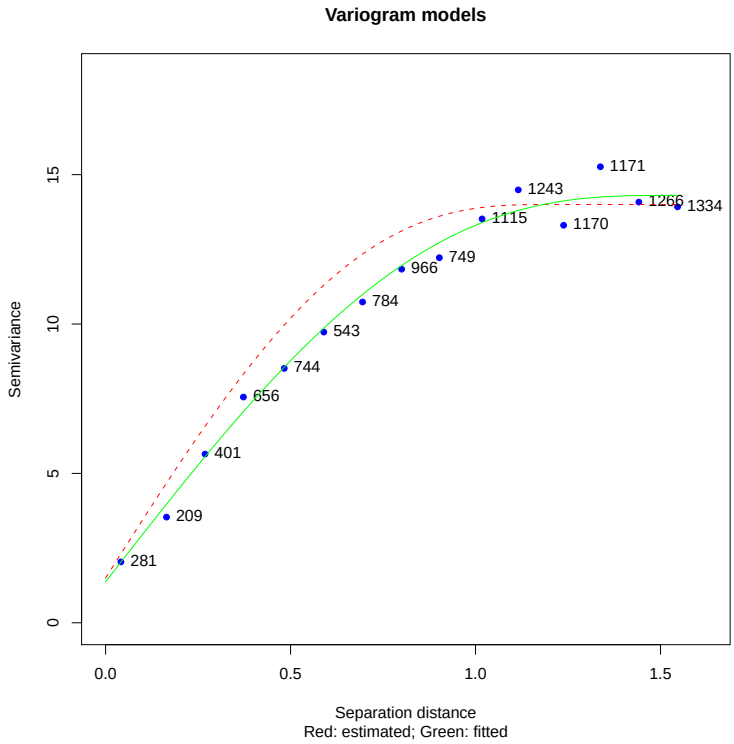
4.3 * Visual comparison of several variogram fits

This **optional** section explains how to visualize different variogram model fits on the same empirical variogram.

Task 31 : Plot emprical variogram with both the estimated and fitted variogram models superimposed, to visualise the effect of the automatic fit. •

The `model` optional argument to the `plot.gstatVariogram` can only show one model; so we have to build the variogram plot with base graphics. The semivariances vs. separation scatterplot can be displayed with the base graphics `plot` method; the variogram functions are computed with the `variogramLine` method of `gstat` and added to the plot with the `lines` method:

```
> plot(v$gamma ~ v$dist, xlim = c(0, max(v$dist) * 1.05),
+     ylim = c(0, max(v$gamma) * 1.2), pch = 20, col = "blue",
+     cex = 1.2, xlab = "Separation distance", ylab = "Semivariance",
+     main = "Variogram models", sub = "Red: estimated; Green: fitted")
> text(v$dist, v$gamma, v$np, pos = 4)
> lines(variogramLine(vm, maxdist = max(v$dist)), col = "red",
+     lty = 2)
> lines(variogramLine(vmf, maxdist = max(v$dist)), col = "green")
```



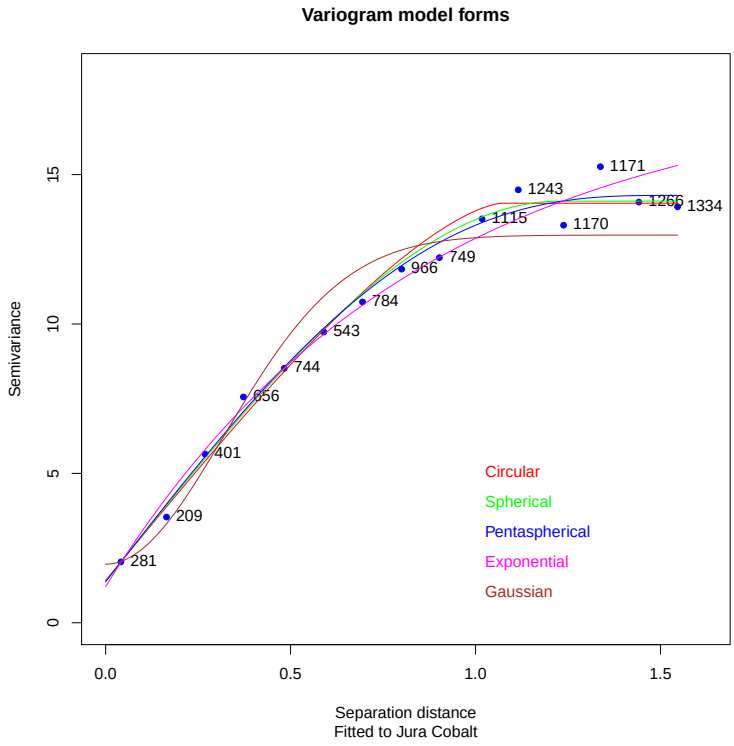
We can extend this approach to visually compare different model forms.

Task 32 : Plot emprical variogram with the fitted variogram models of different classes superimposed, to visualise the effect of the automatic fit. •

We use the same basic plot, but then compute different model fits and plot them. The fits all begin with the same parameters; the fitting method `fit.variogram` will adjust them all with no problem, as long as the starting point is reasonable and the variogram is well-structured, as is here the case.

Notice the use of the `text` method to add text to the plot, and the `pos` argument to this method to position the text, in this case to the right of the given position.

```
> plot(v$gamma ~ v$dist, xlim = c(0, max(v$dist) * 1.05),
+     ylim = c(0, max(v$gamma) * 1.2), pch = 20, col = "blue",
+     cex = 1.2, xlab = "Separation distance", ylab = "Semivariance",
+     main = "Variogram model forms", sub = "Fitted to Jura Cobalt")
> text(v$dist, v$gamma, v$np, pos = 4)
> lines(variogramLine(fit.variogram(v, vgm(12.5, "Cir",
+     1.2, 1.5)), maxdist = max(v$dist)), col = "red")
> lines(variogramLine(fit.variogram(v, vgm(12.5, "Sph",
+     1.2, 1.5)), maxdist = max(v$dist)), col = "green")
> lines(variogramLine(fit.variogram(v, vgm(12.5, "Pen",
+     1.2, 1.5)), maxdist = max(v$dist)), col = "blue")
> lines(variogramLine(fit.variogram(v, vgm(12.5, "Exp",
+     1.2/3, 1.5)), maxdist = max(v$dist)), col = "magenta")
> lines(variogramLine(fit.variogram(v, vgm(12.5, "Gau",
+     1.2/sqrt(3), 1.5)), maxdist = max(v$dist)), col = "brown")
> text(1, 5, col = "red", "Circular", pos = 4)
> text(1, 4, col = "green", "Spherical", pos = 4)
> text(1, 3, col = "blue", "Pentaspherical", pos = 4)
> text(1, 2, col = "magenta", "Exponential", pos = 4)
> text(1, 1, col = "brown", "Gaussian", pos = 4)
```



Note that the approximate range, here set to 1.2 km, must be divided by 3 for the Exponential model, and $\sqrt{3}$ for the Gaussian model, to obtain the range parameter.

Q25 : *What are the similarities and differences in these model forms, for the models fitted to this empirical variogram?* *Jump to A25* •

4.4 * Goodness-of-fit

This **optional** section explains how to quantify how well the variogram model fits the empirical variogram.

A variogram model fitted by `fit.variogram` method receives an **attribute** that reports how closely the method was able to fit the empricial variogram; this is called **internal goodness-of-fit**.

Task 33 : Determine the goodness-of-fit of this automatic fit. •

We use the very general `attributes` method to extract the attributes of an object; these are metadata about the object which are created by various methods:

```
> str(attributes(vmf))

List of 5
 $ names    : chr [1:9] "model" "psill" "range" "kappa" ...
 $ row.names: int [1:2] 1 2
 $ class    : chr [1:2] "variogramModel" "data.frame"
 $ singular : logi FALSE
 $ SSErr    : num 4804
```

The goodness-of-fit is attribute `SSErr`; we extract it as a field:

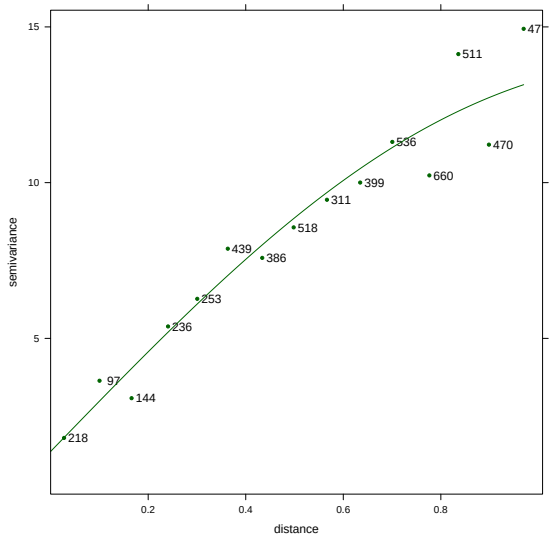
```
> attributes(vmf)$SSErr

[1] 4803.9
```

This is the **residual sum of squares**, weighted as in the model fitting method, from the model fit. The lower the better, of course.

It is possible to fit several models and compare their goodness-of-fit; however this is a poor substitute for understanding the process that generated the realization of the spatial field sampled by the empirical variogram. Also, the goodness-of-fit depends on the cutoff and number of bins, as we can see by re-fitting this same model with a different empirical variogram from the same data, but using a cutoff of 1 km insteaf of 1.6 km. In this case we can start with the automatic fit from the previous fit:

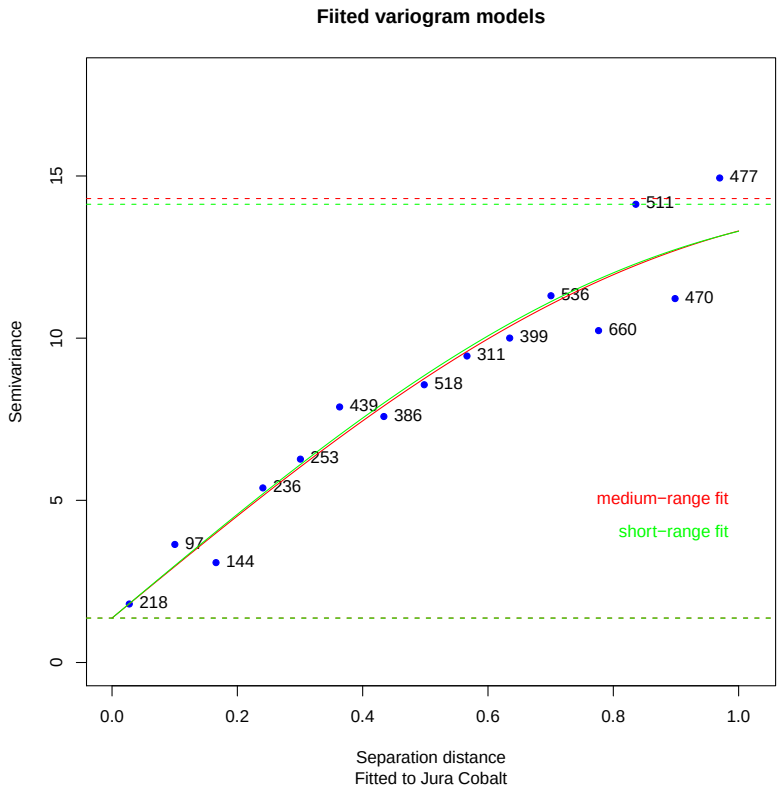
```
> v2 <- variogram(Co ~ 1, loc = jura.cal, cutoff = 1)
> vmf2 <- fit.variogram(v2, vmf)
> print(vmf)

  model   psill  range
1   Nug  1.3712 0.0000
2   Pen 12.9322 1.5239

> print(vmf2)

  model   psill  range
1   Nug  1.3696 0.0000
2   Pen 12.7581 1.4774

> attributes(vmf)$SSErr
```

```
[1] 4803.9

> attributes(vmf2)$SSErr

[1] 20569

> print(plot(v2, plot.numbers = T, pch = 20, col = "darkgreen",
+     model = vmf2))
```



We can visualize the two fits together, also showing the two nugget and sill values as horizontal lines:

```
> plot(v2$gamma ~ v2$dist, xlim = c(0, max(v2$dist) * 1.05),
+     ylim = c(0, max(v2$gamma) * 1.2), pch = 20, col = "blue",
+     cex = 1.2, xlab = "Separation distance", ylab = "Semivariance",
+     main = "Fiited variogram models", sub = "Fitted to Jura Cobalt")
> text(v2$dist, v2$gamma, v2$np, pos = 4)
> lines(variogramLine(vmf, maxdist = 1), col = "red")
> lines(variogramLine(vmf2, maxdist = 1), col = "green")
> text(1, 5, col = "red", "medium-range fit", pos = 2)
> text(1, 4, col = "green", "short-range fit", pos = 2)
> abline(h = vmf$psill[1], col = "red", lty = 2)
> abline(h = sum(vmf$psill), col = "red", lty = 2)
> abline(h = vmf2$psill[1], col = "green", lty = 2)
> abline(h = sum(vmf2$psill), col = "green", lty = 2)
```



Q26 : *How much did the fitted model change, when the automatic fit considered only the short-range variogram (1 km), compared to the medium-range variogram (1.6 km)?* *Jump to A26*

•

Q27 : *How did the fitting error change? Explain why. Does a larger error in this case mean that the model is not suitable?* *Jump to A27* •

We are done with the short-range variogram and its model, so clean up:

```
> rm(v2, vmf2)
```

We can also compare the fit of different model forms on the same empirical variogram.

```
> attributes(fit.variogram(v, vgm(12.5, "Pen", 1.2, 1.5)))$SSErr

[1] 4803.9

> attributes(fit.variogram(v, vgm(12.5, "Cir", 1.2, 1.5)))$SSErr

[1] 6161.7

> attributes(fit.variogram(v, vgm(12.5, "Sph", 1.2, 1.5)))$SSErr

[1] 5113.6
```

```
> attributes(fit.variogram(v, vgm(12.5, "Exp", 1.2/3, 1.5)))$SSErr

[1] 7890

> attributes(fit.variogram(v, vgm(12.5, "Gau", 1.2/sqrt(3),
+     1.5)))$SSErr

[1] 17114
```

Q28 : *Which of these models has the closest fit? Are there major differences?* *Jump to A28*

•

Challenge: Display the empirical variogram with the different fitted models on one graph.

Task 34 : Clean up the workspace from this section. •

```
> rm(v, vm, vmf)
```

4.5 * Fitting models with multiple structures

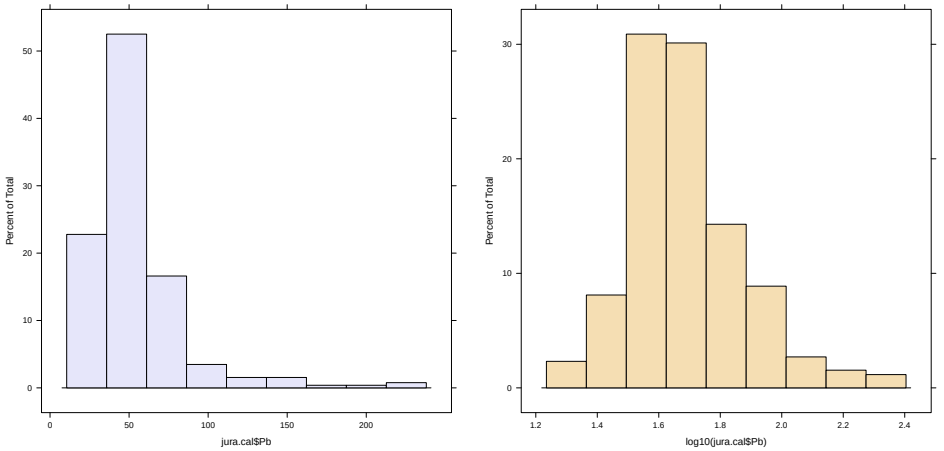
This **optional** section discusses variogram models with more than one structural component.

In the previous section we were able to model an empirical variogram with one **structural** component and a **nugget**. This suggests that there is only one local spatial process, plus noise that can not be modelled. However, it may be that several spatial processes, with different spatial scales and perhaps different forms, were responsible for the overall process which we see reflected in the empirical variogram. In this case we may need to fit several structures to one variogram.

In the Jura dataset, several of the metals exhibit an empirical variogram that suggest multiple structures. One of these is lead (Pb). Since it is strongly right-skewed, we log-transform it to an approximately symmetric form before computing the variogram.

Task 35 : Plot the untransformed and log-transformed histograms of Pb. •

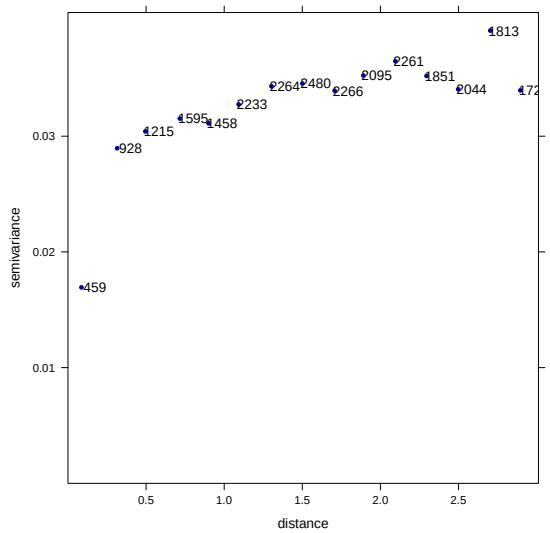
```
> h1 <- histogram(jura.cal$Pb, col = "lavender")
> h2 <- histogram(log10(jura.cal$Pb), col = "wheat")
> plot(h1, split = c(1, 1, 2, 1), more = T)
> plot(h2, split = c(2, 1, 2, 1), more = F)
> rm(h1, h2)
```



Q29 : *Describe the shapes of these two histograms. Which one should be used to compute semi-variances?* *Jump to A29* •

Task 36 : Compute the empirical variogram of the log10(Pb) values and plot it. •

```
> v <- variogram(log10(Pb) ~ 1, loc = jura.cal, cutoff = 3)
> plot(v, plot.numbers = T, pch = 20, col = "darkblue")
```



Q30 : *What is the evidence for two structures in this empirical variogram? What are their approximate ranges?* *Jump to A30* •

We first try to fit this with one structure and a nugget.

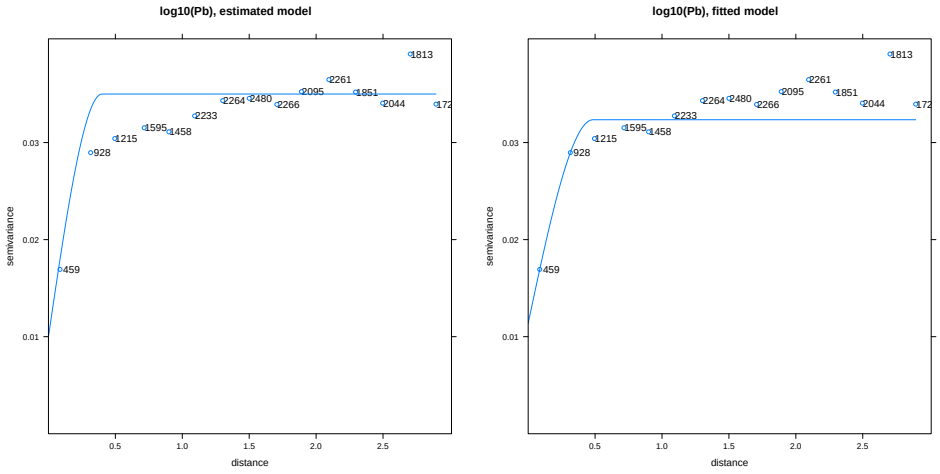
Task 37 : Estimate and fit a single-structure variogram model and nugget, with priority to the short-range structure. •

We first model the steep rise with a spherical model with short range:

```
> (vm <- vgm(0.025, "Sph", 0.4, 0.01))

  model psill range
1   Nug 0.010   0.0
2   Sph 0.025   0.4

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

  model    psill   range
1   Nug 0.011386 0.00000
2   Sph 0.020966 0.48408
```

Plot the estimated and fitted variogram models:

```
> p1 <- plot(v, model = vm, pl = T, main = "log10(Pb), estimated model")
> p2 <- plot(v, model = vmf, pl = T, main = "log10(Pb), fitted model")
> plot(p1, split = c(1, 1, 2, 1), more = T)
> plot(p2, split = c(2, 1, 2, 1), more = F)
> rm(p1, p2)
```



Q31 : *Describe this fit. What is the range of the model? Does this match the range of the variogram?* *Jump to A31* •

Q32 : *What is the approximate difference between the total sill of the fitted single-structure model and the total sill of the empirical variogram?* *Jump to A32* •

Task 38 : Add a second structural component, to account for the long-range structure. •

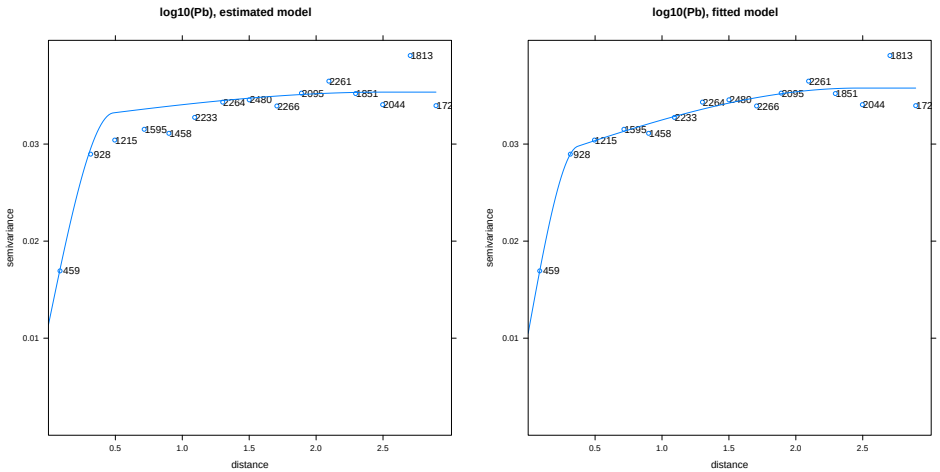
Since we used a spherical model for the short-range structure, we also use it for the long-range structure. Our first guess for the partial sill of the long-range model is difference between the total sill of the fitted single-structure model and the total sill of the empirical variogram (see previous question); our first guess for the range is the empirical range.

The `vgm` method takes an optional `add.to` argument, which is the variogram model to which we want to add a component. In this case we have a fitted short-range model (`vmf`); we add another component to it, and then re-fit both parts together:

```
> (vm <- vgm(0.003, "Sph", 2.5, add.to = vmf))

  model    psill   range
1   Nug 0.011386 0.00000
2   Sph 0.020966 0.48408
3   Sph 0.003000 2.50000

> (vmf2 <- fit.variogram(v, vm))

  model     psill  range
1   Nug 0.0105050 0.0000
2   Sph 0.0175409 0.3702
3   Sph 0.0077225 2.4660
```

We plot the estimated and fitted variogram models:

```
> p1 <- plot(v, model = vm, pl = T, main = "log10(Pb), estimated model")
> p2 <- plot(v, model = vmf2, pl = T, main = "log10(Pb), fitted model")
> print(plot(p1, split = c(1, 1, 2, 1), more = T))
> print(plot(p2, split = c(2, 1, 2, 1), more = F))
> rm(p1, p2)
```



Q33 : *Describe this fit. What are the two ranges? What does this fitted model imply about the spatial processes?* *Jump to A33* •

Q34 : *What are the relative proportions of spatial varability accounted for*

by the three model components (two structures and nugget)? *Jump to A34* •

```
> paste("Nugget:", round(100 * vmf2$psill[1]/sum(vmf2$psill),
+     0), "%")

[1] "Nugget: 29 %"

> paste("Short range:", round(100 * vmf2$psill[2]/sum(vmf2$psill),
+     0), "%")

[1] "Short range: 49 %"

> paste("Long range:", round(100 * vmf2$psill[3]/sum(vmf2$psill),
+     0), "%")

[1] "Long range: 22 %"
```

Task 39 : Clean up the workspace from this section. •

```
> rm(v, vm, vmf, vmf2)
```

4.6 * Fitting anisotropic variograms

In Exercise 2 §8 we saw how some variables exhibit **anisotropy**. In this **optional** section we see how to fit a variogram model to an anisotropic empirical variogram.

There are several kinds of anisotropy revealed in empirical variogram:

1. same model, same sill, but different ranges in different directions: **geometric**, also called **affine**, anisotropy;
2. same model, same range, but sill varies with direction: **zonal** anisotropy;
3. apparently different model in different directions.

The last case is theoretically very difficult to deal with, and would require different types of processes in different directions.

The second case (zonal anisotropy) is also difficult because it assumes different overall variance according to direction, i.e. the process was more variable in one direction.

The simplest case, and the only one we will consider, is the first case of geometric anisotropy. Essentially all we have to do is "stretch" the model in one direction: same sill, but different ranges. This results in an **anisotropy elipse** with a **major axis** of spatial structure in one direction and a **minor axis** in the orthogonal direction, i.e. at right angles

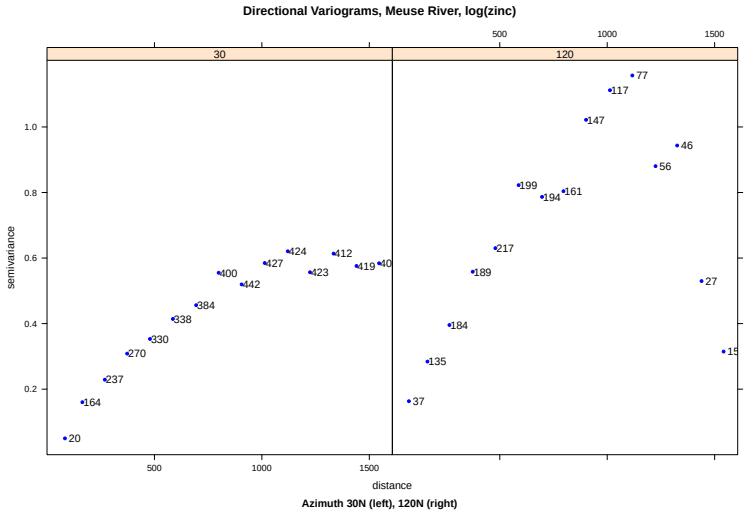
The Jura data exhibits no anisotropy, so for this section we return to the Meuse dataset used in Exercise 2 §8.

Task 40 : Load the `meuse` sample dataset from the `sp` package and convert it from a dataframe to a spatial object by specifying its coördinates. •

```
> data(meuse)
> coordinates(meuse) <- ~x + y
```

Task 41 : Compute and display the directional variograms of the logarithm of Zn content at 30° N and 120° N, i.e. the suspected major and minor axes of the anisotropy elipse. •

```
> v.a <- variogram(log(zinc) ~ 1, meuse, alpha = c(30,
+     120), cutoff = 1600)
> print(plot(v.a, main = "Directional Variograms, Meuse River, log(zinc)",
+     sub = "Azimuth 30N (left), 120N (right)", pl = T,
+     pch = 20, col = "blue"))
```



Q35 : *What variogram model form do you suggest to fit the variogram of the suspected major axis (30 ° N)?* *Jump to A35* •

Q36 : *Does it appear that the sills are the same for the major and minor axes?* *Jump to A36* •

Even though the anisotropy does not appear to be geometric, we will model it that way because (1) the small sample size makes the variogram for the minor axis not very reliable; (2) the dimension of the study area in the direction of the minor axis is small, so that modelling longer ranges is not important for interpolation. This is perhaps a questionable decision, but it does allow us to illustrate the modelling procedure.

> **Note:** Zonal anisotropy (i.e. anisotropy in the sill) can indeed be modelled, by combining two models (§4.5), one for each of the orthogonal axes, and in each case defining an affine (geometric) anisotropy structure with very small ratios. Then the "minor" axis in both cases will have almost no influence on the predictions.

To model geometric (affine) anisotropy with the `vgm` method of the `gstat` package, the two sills should be the same; all we need to specify is (1)

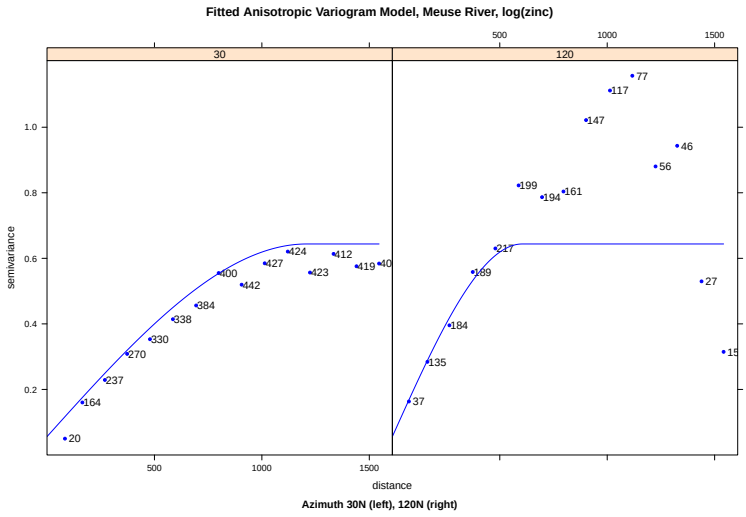
the **azimuth** of the major axis, in degrees from N, and (2) the **anisotropy ratio**, i.e. the ratio of the minor range to the major range. These two are combined with the `c` method as one list in the optional `anis` "anisotropy" argument to `vgm`, for example `anis=c(30, 0.5)`. So we need to determine two parameters:

1. The azimuth of the major axis; we do this by comparing various directional variograms as explained in Exercise 2 §8;
2. The ranges of the major and minor axes; the ratio of the minor to the major is the anisotropy ratio (i.e. the elipse flattening).

Q37 : *What is an appropriate anisotropy ratio? Note: since the two sills are different, answer this by finding the range where the minor axis variogram reaches the sill of the major axis variogram.* *Jump to A37* •

Task 42 : Fit a spherical variogram model, with geometric anisotropy, to the directional variograms and display the fit. •

```
> (vmf.a <- fit.variogram(v.a, vgm(0.55, "Sph", 1100, 0.05,
+     anis = c(30, 0.5))))

  model    psill  range ang1 anis1
1   Nug 0.056095    0.0    0   1.0
2   Sph 0.587719 1208.7   30   0.5

> attributes(vmf.a)$SSErr

[1] 0.00039621

> print(plot(v.a, main = "Fitted Anisotropic Variogram Model, Meuse River, log(zinc)",
+     model = vmf.a, sub = "Azimuth 30N (left), 120N (right)",
+     pl = T, pch = 20, col = "blue"))
```

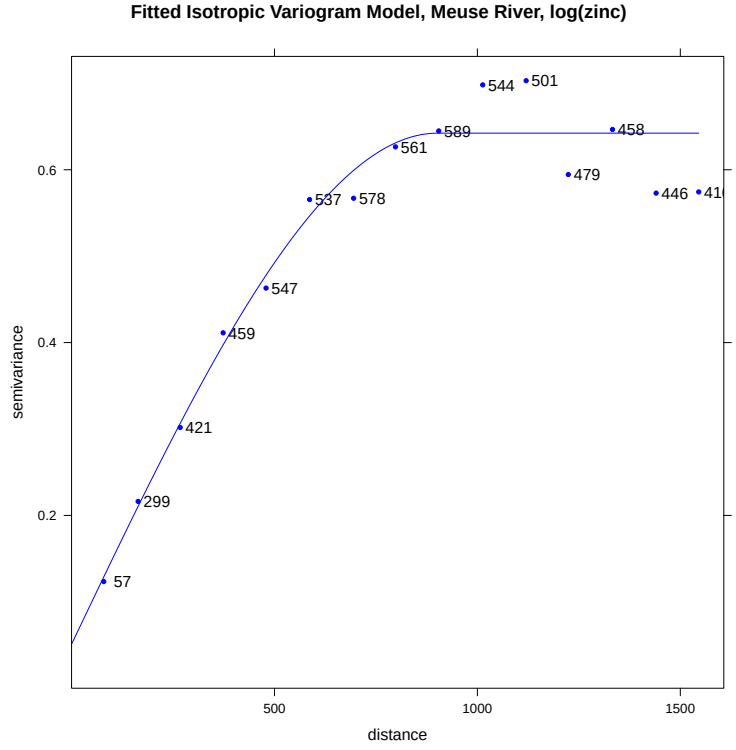


Q38 : *What variogram parameters did method* `fit.variogram` *alter?*

Jump to A38 •

Task 43 : Compute the best omnidirectional spherical model for the same variable and display the fitted model on the empirical variogram. •

```
> v <- variogram(log(zinc) ~ 1, meuse, cutoff = 1600)
> (vmf <- fit.variogram(v, vgm(0.55, "Sph", 1100, 0.05)))

  model   psill range
1   Nug 0.05097   0.0
2   Sph 0.59140 901.8

> attributes(vmf)$SSErr

[1] 9.4538e-06

> print(plot(v, main = "Fitted Isotropic Variogram Model, Meuse River, log(zinc)",
+     model = vmf, pl = T, pch = 20, col = "blue"))
```



Q39 : *Compare the two models: parameters and fit (visual and numerical).* *Jump to A39* •

Q40 : *Would you use the isotropic or anisotropic variogram models for geostatistical prediction (kriging) in this study area?* *Jump to A40* •

Task 44 : Clean up the workspace from this section. •

```
> rm(v.a, vmf.a, v, vmf, meuse)
```

4.7 * The effect of sampling on variogram modelling

This **optional** section investigates the causes of uncertainty in variogram modelling.

We must never forget that the **empirical variogram** is just that, not the "true" variogram which, we assume (presume?) represents the spatially-correlated random process that resulted in the realization. We would like to model the true variogram (the process), but we only have an empirical variogram.

To review the sources of uncertainty in estimating a variogram model:

1. We only have **one realization** of the random process;
2. We only have a **small sample** of the realization;
3. We must specify **empirical variogram parameters**: cutoff distance and number of bins (equivalently, bin width); this can have a large influence especially on automated fits.

How can we estimate the effects of these?

1. It is impossible to generate another realization in the same sample area, but in a "similar" sample area, where we can assume that the spatially-correlated random process is the same, we should (in theory) determine the same variogram model;
2. With one sample in hand, we can estimate the "true" variogram model from several **sub-samples**; if these are similar we can have confidence in our estimate;
3. Even with just one sample and one empirical variogram, we can try different cutoffs and bin widths; again, if the estimated "true" models are similar, we can have confidence in our estimate. We saw one example of this in §4.4 when we compared the fit for the short- and medium-range empirical variograms.

We first investigate the effect of sample size and sub-sampling. The calibration dataset `jura.cal` has 259 observations; we will simulate the case where we have only 160 observations; but distributed differently on the landscape.

Task 45 : Draw four random samples of 160 observations from the calibration dataset. Plot the histograms of their Co concentrations side-by-side on the same scale, and the default empirical variograms, also of Co concentrations. •

We use the `sample` method (as in §3.1) to select rows (without replacement). We build a **list** data structure with the four samples, using the `for` operator to loop and the `[[]]` operator to specify the list element. We must first initialize the 4-element list with the `vector` "create a vector" method, then we can add structures in the slots.

```
> jura.cal.sub <- vector("list", 4)
> for (i in 1:4) jura.cal.sub[[i]] <- jura.cal[sample(1:dim(jura.cal@data)[1],
+     160), ]
```

Task 46 : Compute the empirical variograms for each of the four samples •

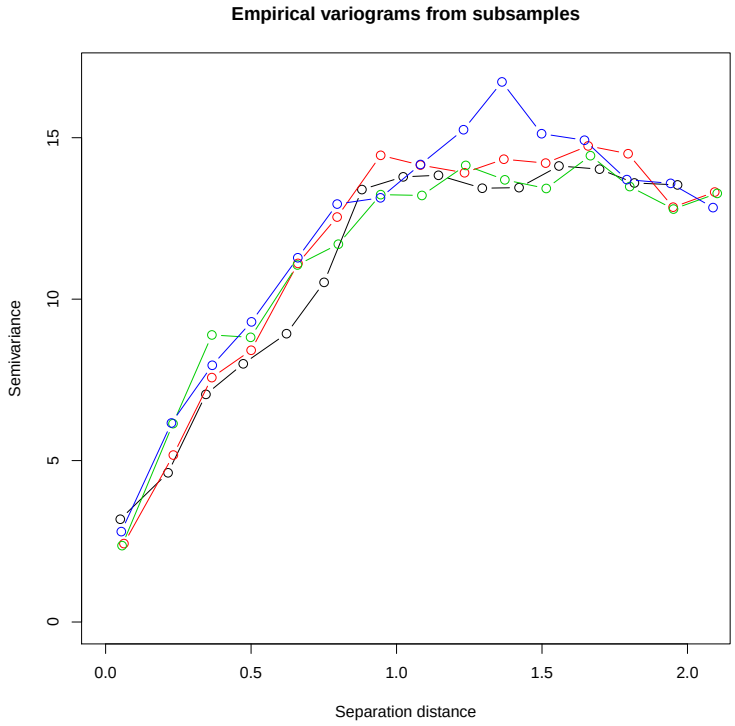
```
> v.sub <- vector("list", 4)
> for (i in 1:4) v.sub[[i]] <- variogram(Co ~ 1, loc = jura.cal.sub[[i]])
```

Task 47 : Display the four empirical variograms on the same graph. •

We use the `plot` method to display the first scatterplot of semivariance against separation, thereby setting up the plot (axes, labels, title etc.). We then use the `points` method to add point sets to the scatterplot; this is called once for each remaining sub-sample, inside a `for` loop.

> **Note:** We show both the points (average semi-variance in the bin) and lines to connect them, with the `type="b"` ("both" points and lines) optional graphics parameter. This line is not a variogram model, it's just to help us visually separate the four empirical variograms.

```
> plot(v.sub[[1]]$gamma ~ v.sub[[1]]$dist, xlim = c(0,
+     max(v.sub[[1]]$dist) * 1.05), ylim = c(0, max(v.sub[[1]]$gamma) *
+     1.2), col = 1, cex = 1.2, xlab = "Separation distance",
+     ylab = "Semivariance", main = "Empirical variograms from subsamples",
+     type = "b")
> for (i in 2:4) {
+     points(v.sub[[i]]$gamma ~ v.sub[[i]]$dist, col = i,
+         cex = 1.2, type = "b")
+ }
```



Note that your graph will differ from the one shown here, since each random sample is different.

Q41 : *How different are the four empirical variograms from each other?* *Jump to A41* •

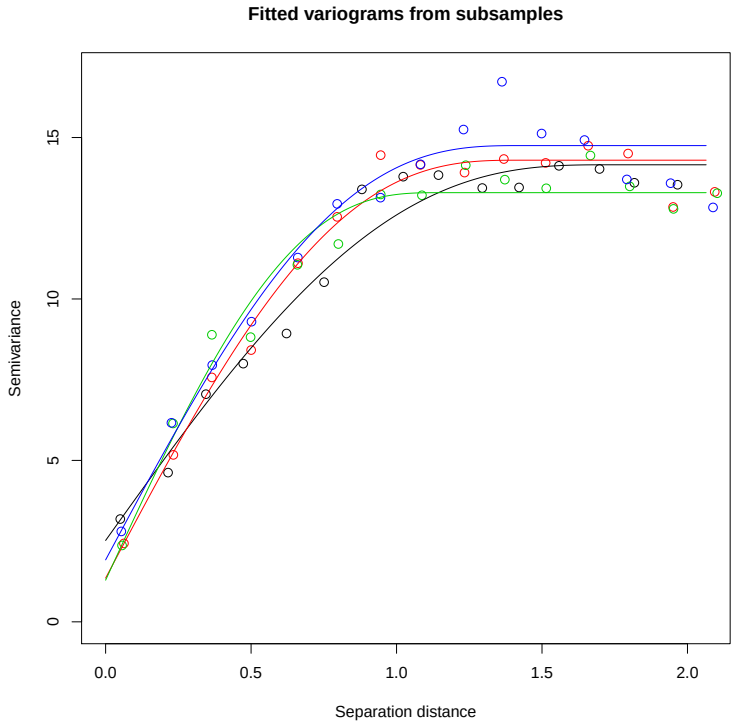
Task 48 : Fit a variogram model to each of the four sub-sample empirical variograms, using as a starting point the initial estimate we used for the full calibration set in §4.2. •

```
> vmf.sub <- vector("list", 4)
> for (i in 1:4) {
+     vmf.sub[[i]] <- fit.variogram(v.sub[[i]], vgm(12.5,
+         "Pen", 1.2, 1.5))
+ }
> print(vmf.sub)

[[1]]
  model   psill  range
1   Nug  2.5209 0.0000
2   Pen 11.6348 1.7304

[[2]]
  model   psill range
1   Nug  1.3543 0.000
2   Pen 12.9436 1.426

[[3]]
  model   psill  range
```

```
1   Nug  1.2903 0.0000
2   Pen 12.0017 1.1457

[[4]]
  model   psill  range
1   Nug  1.9191 0.0000
2   Pen 12.8310 1.4293
```

Task 49 : Display the four fitted variogram models on the same graph, along with their empirical variograms. •

```
> xlim.max <- max(v.sub[[1]]$dist) * 1.05
> plot(v.sub[[1]]$gamma ~ v.sub[[1]]$dist, xlim = c(0,
+     xlim.max), ylim = c(0, max(v.sub[[1]]$gamma) * 1.2),
+     col = 1, cex = 1.2, xlab = "Separation distance",
+     ylab = "Semivariance", main = "Fitted variograms from subsamples")
> for (i in 2:4) {
+     points(v.sub[[i]]$gamma ~ v.sub[[i]]$dist, col = i,
+         cex = 1.2)
+ }
> for (i in 1:4) {
+     lines(variogramLine(vmf.sub[[i]], maxdist = xlim.max),
+         col = i)
+ }
> rm(xlim.max)
```



Q42 : *How different are the four fitted variograms? Where do they differ most?* *Jump to A42* •

We can answer this question visually (looking at the figure) but also numerically. We collect the four values of each parameter into one vector for each, then report the sorted values and a non-parametric indicator of the variation (range divided by the median)[2].

```
> nuggets <- NULL
> psills <- NULL
> ranges <- NULL
> for (i in 1:4) {
+     nuggets <- c(nuggets, vmf.sub[[i]]$psill[1])
+     psills <- c(psills, vmf.sub[[i]]$psill[2])
+     ranges <- c(ranges, vmf.sub[[i]]$range[2])
+ }
> print(sort(nuggets))

[1] 1.2903 1.3543 1.9191 2.5209

> diff(range(nuggets))/median(nuggets)

[1] 0.75184

> print(sort(psills))

[1] 11.635 12.002 12.831 12.944

> diff(range(psills))/median(psills)

[1] 0.10541

> print(sort(ranges))

[1] 1.1457 1.4260 1.4293 1.7304

> diff(range(ranges))/median(ranges)

[1] 0.40955

> rm(nuggets, psills, ranges)
```

Challenge: Repeat this exercise with smaller sub-samples (120, 80, 60 . . .). What is the effect of sample size on between-sample variability of the empirical variogram and fitted variogram model?

Q43 : *Which of these fits best models the spatial process?* *Jump to A43* •

Task 50 : Remove the temporary objects created in this section from the workspace. •

```
> rm(jura.cal.sub, v.sub, vmf.sub, i)
```

[2] This is similar to the well-known **coefficient of variation**, i.e. the standard deviation divided by the mean. The non-parametric version doesn't assume any distribution, it just gives an idea of the normalized magnitude of the spread

4.8 Answers

A15 : *The distribution is more-or-less normal with a mode near 12 mg kg^{-1}, but with a second mode near 3 mg kg^{-1}. Transformation would not improve this. There are no extreme values that would result in unreliable semivariances, so the untransformed attribute is suitable for variogram estimation.* *Return to Q15* •

A16 : *Yes, it does not keep increasing; it reaches a value of about 14 mg kg^{-12} near 1 km and then is more-or-less constant past this range.* *Return to Q16* •

A17 : *Yes, past about 1.6 km the semi-variances decrease slightly; this indicates that this is past the range of local spatial dependence.* *Return to Q17* •

A18 : *Spherical, circular, pentaspherical.* *Return to Q18* •

A19 : *The total sill is about 14 (mg kg^{-1})^2, of which the Nugget is about 1.5 (mg kg^{-1})^2, so the structural sill is about 14 - 1.5 = 12.5 (mg kg^{-1})^2; Range 1.2 km.* *Return to Q19* •

A20 : *Because of the gradual transition "shoulder", the pentaspherical model appears best.* *Return to Q20* •

A21 : *The form is more-or-less correct but the semivariances of the model are greater than the empirical semivariances for most of the range.* *Return to Q21* •

A22 : *The range was increased by 0.324 km; the nugget was decreased by* 0.129 *and the partial sill of the structural component was increased by* 0.432*; the total sill was thus increased by* 0.303. *Return to Q22* •

A23 : *Very well.* *Return to Q23* •

A24 : 90.4% *Return to Q24* •

A25 : *The three "spherical"-type models (Circular, Spherical, and Pentaspherical) are very similar; only the width of the "shoulder" transition is somewhat different. Their fitted sills are almost identical. The Exponential model agrees with these three until about 0.5 km, when it is lower, and after 1.2 km higher, since it does not reach its sill in the range of this plot. It suggest a longer-range dependence. The Gaussian model fits quite poorly, because it models a continuity at the origin which is not found. So it has to include an extra inflection point, causing an overestimation in the 0.5 - 1.0 km range, and a lower sill.* *Return to Q25* •

A26 : *The fit has changed somewhat: the nugget is almost the same but the partial sill and range somewhat lower.* *Return to Q26* •

A27 : *The error is much larger. This is because the short-range empirical variogram, with the same number of bins (15), is more erratic, so the model does not come so close to the empirical semivariances. Obviously, this can not mean that the model is less suitable to the process.* *Return to Q27* •

A28 : *The Pentaspherical model gives the best fit, but the Spherical is almost as good, and the Circular also close. The Exponential model (asymptotic to the sill and a rapid rise in semivariance near the origin) is much worse, and the Gaussian (strong continuity at the origin) by far the worst; this model is not suited to the form of this empirical variogram, nor to the process.* *Return to Q28* •

A29 : *Untransformed: strongly right-skewed; log-transformed: nearly symmetric. The log-transformed variable should be used, to avoid an erratic variogram.* *Return to Q29* •

A30 : *To about 600 m there is a very steep rise in semivariance; then to about 2 km there is a steady but slow increase.* *Return to Q30* •

A31 : *The structure to about 400 m is well-fitted but then the slowly-increasing semivariance out to 2.5 km is not at all modelled. The range of the model is about 500 m, much shorter than the actual overall range.* *Return to Q31* •

A32 : *The total fitted sill is about 0.32; the empirical sill is about 0.35; so we estimate the partial sill of the long-range structure to be 0.003.* *Return to Q32* •

A33 : *There is a short-range structure to 370 m, with a very rapid decrease in spatial dependence; then a long-range structure to 2.5 km, where spatial dependence slowly decreases. This implies two separate spatial processess responsible for the spatial dependence of Pb. The short-range could be from human activity or depositional hot spots; the long-range from rock type or more diffuse deposition (e.g. from dust).* *Return to Q33* •

A34 : *The short-range component accounts for about half the total variation; the long range about a fifth; and the nugget (unexplained) almost a third.* *Return to Q34* •

A35 : *There is a definite sill (so an exponential model is not indicated); the transition from the more linear part at close separations is gradual; this suggests that a spherical model is appropriate.* *Return to Q35* •

A36 : *No it does not! There seems to be more variability on the minor axis, i.e.*

at azimuth 120°N. *Return to Q36* •

A37 : *The major axis variogram reaches a sill of about 0.6, at a range of about 1100 m; this same sill value is reached at about 550 m on the minor axis. The ratio is therefore (550/1100) = 0.5.* *Return to Q37* •

A38 : *The range of the major axis, the structural sill, and the nugget. The anisotropy parameters (major axis and ratio) are* not *adjusted by the automatic fit.* *Return to Q38* •

A39 : *The partial sill and nugget are very close, but the range of the isotropic model is considerably shorter (900 vs. 1200 m); this is because of the averaging effect of the shorter-range (minor axis) and longer-range (major axis) ranges. The visual and numeric fit is much better for the isotropic model, but again this is probably due to averaging.* *Return to Q39* •

A40 : *The anisotropic model should be used, because there is a large difference in the strength of spatial dependence in the two directions. If an isotropic model is used, points separated along or near the minor axis direction will be weighted too much.* *Return to Q40* •

A41 : *This varies each time the four sub-samples are drawn. In general, the close-range structure is more consistent.* *Return to Q41* •

A42 : *Again, this varies each time the four sub-samples are drawn. In general, the nuggets and ranges are similar, with more variation in partial sills of the structural component. But, this may be different with your sub-samples.* *Return to Q42* •

A43 : *They are all equally-good estimates, there is no way to know which sub-sample gives (by change) the closest match to the presumed "true" spatial process.* *Return to Q43* •

5 * Fitting a trend surface with generalized least squares

This **optional** section discusses the difficulties with the naïve trend surface fit of §2, and explains how to compute a trend surface more correctly with **generalized least squares**. It also illustrates **stepwise regression** and the use of two additional packages: `spatial` and `MASS`.

For this part of the exercise we use the `spatial` package written by Ripley and documented in his texts [6, 7]. This gives us a different perspective on spatial data; in particular this package has efficient methods to compute and plot trend surfaces using both OLS and GLS. We will also use some functions from Venables & Ripley's `MASS` ("Modern Applied Statistics with S") package, extensively documented in the text of the same name [7].

5.1 Theoretical background

The problem with ordinary least squares (OLS) as implemented in the `lm` method is that the residuals may well be **spatially-correlated**. This violates a major assumption underlying OLS, and may result in a substantially incorrect trend surface equation. In particular, a large number of close-by points with similar values will "pull" a trend surface towards them. Furthermore, the OLS R^2 may be over-optimistic.

The solution is to use Generalised Least Squares (GLS) to estimate the trend surface. This allows a **covariance structure between residuals** to be included directly in the least-squares solution of the regression equation. GLS is a special case of **Weighted Least Squares** (WLS).

The GLS estimate of the regression coefficients is [1]:

$$\hat{\beta}_{\text{gls}} = (X^T \cdot C^{-1} \cdot X)^{-1} \cdot X^T C^{-1} \cdot y$$

where X is the design matrix, C the **covariance matrix** of the (spatially-correlated) **residuals**, and y the **observations**.

Note that if there is no spatial dependence among the errors, C reduces to $I\sigma^2$ and the estimate to OLS:

$$\hat{\beta}_{\text{ols}} = (X^T \cdot X)^{-1} \cdot X^T \cdot y$$

The covariance matrix C gives the covariance between the residuals at each pair of points used to determine the $\hat{\beta}_{\text{gls}}$. Clearly, there is no way to know this, so we **model** the covariance as a function of the **separation** (e.g. distance) between point pairs, similar to what we did in §4 to fit a variogram model. However, we instead fit a **spatial covariance** model.

This leads us to a further difficulty: the covariance structure refers to the residuals, but we can't compute these until we fit the trend ... but we need the covariance structure to fit the trend ... and so on. This is a classic "which came first: the chicken or the egg?" problem.

In practice, it is usually sufficient to:

1. make a first estimate of the trend surface with OLS;
2. compute the residuals;
3. model the covariance structure of the OLS residuals as a function of their separation;
4. use this covariance structure to determine the weights to compute the GLS trend surface.

The GLS residuals could again be modelled to see if their covariance structure differs from that estimated from the OLS residuals; in practice, unless the dataset is large it is not possible to see any such difference.

GLS trend surfaces can be computed in several R packages. The `lm` method itself can be used for weighted least squares (WLS), but the weights have to

be computed from the spatial correlation structure. This is implemented in the `gls` method of the `nlme` package, but we will use the somewhat simpler formulation in the `spatial` package written by Ripley and documented in his texts [6, 7]. This package is installed by default with the base R.

5.2 The 'topo' sample dataset

The Cameroon soil properties dataset used in §2 is not a good choice for this demonstration, because the residuals from the OLS trend surface (either first- or second-order) have very low spatial correlation, as can be appreciated in the plots of §2: high positive and negative residuals are located near to each other. It seems that what is not well-modelled by the trend surface is noise, due to some non-geographic process.

The `MASS` package includes a nice sample dataset, `topo`, that will illustrate the differences between OLS and GLS trend surfaces. This dataset is originally from Davis [2, Ch. 5], where it is called `NOTREDAM.TXT`[3]; it is a manual topographic survey of a small area by a student surveying class. The challenge is to make a contour map or a gridded prediction of the heights in the study area. Here we will only consider the trend surfaces; for a full treatment see Venables and Ripley [7, Ch. 15].

Task 51 : Load the `spatial` and `MASS` packages, and the `topo` dataset; examine the dataset structure. •

```
> library(spatial)
> library(MASS)
> help(topo)
> str(topo)
> summary(topo)
> attach(topo)
```

Q44 : *What do the three fields represent?* *Jump to A44* •

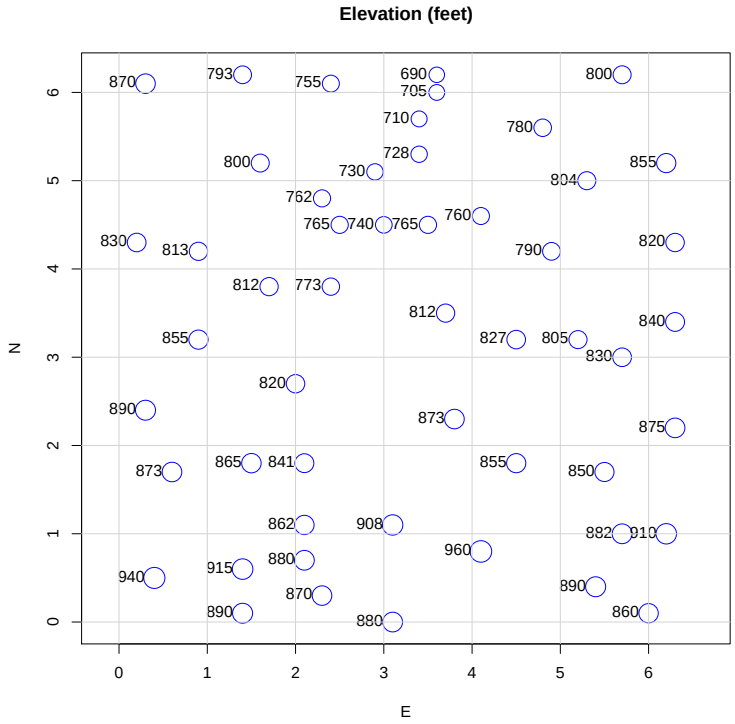
5.3 Step 1: OLS trend surface

We begin by fitting an OLS trend surface.

Task 52 : Make a postplot of the elevations. •

```
> plot(y ~ x, cex = 3 * z/max(z), pch = 1, col = "blue",
+     asp = 1, main = "Elevation (feet)", xlab = "E", ylab = "N")
> text(x, y, z, pos = 2)
> grid(lty = 1)
```

[3] Also available from the author's web site, http://www.kgs.ku.edu/Mathgeo/Books/Stat/index.html



Q45 : *What seems to be the regional trend?* *Jump to A45* •

The form suggests a second-order surface.

Task 53 : Use the `surf.ls` method to compute the second-order OLS trend surface; display its analysis of variance and coefficients. •

The `surf.ls` method works on a data frame with two coördinates and one value to be modelled. The first argument is the degree of the trend surface.

```
> ts2 <- surf.ls(2, x, y, z)
> class(ts2)

[1] "trls"

> summary(ts2)

Analysis of Variance Table
 Model: surf.ls(np = 2, x = x, y = y, z = z)
           Sum Sq Df  Mean Sq F value   Pr(>F)
Regression 156072  5 31214.31  35.934 8.44e-15
Deviation   39958 46   868.66
Total      196030 51
Multiple R-Squared: 0.796,        Adjusted R-squared: 0.774
AIC: (df = 6) 357.51
Fitted:
   Min     1Q Median     3Q    Max
   736    775    826    870    942
Residuals:
```

```
   Min     1Q Median     3Q    Max
-63.25 -14.36  -4.87  15.00  97.75

> ts2$beta

[1] 801.2176 -11.0189  68.2291 -73.9930   3.3436   8.3427
```

We now visualize the computed surface.

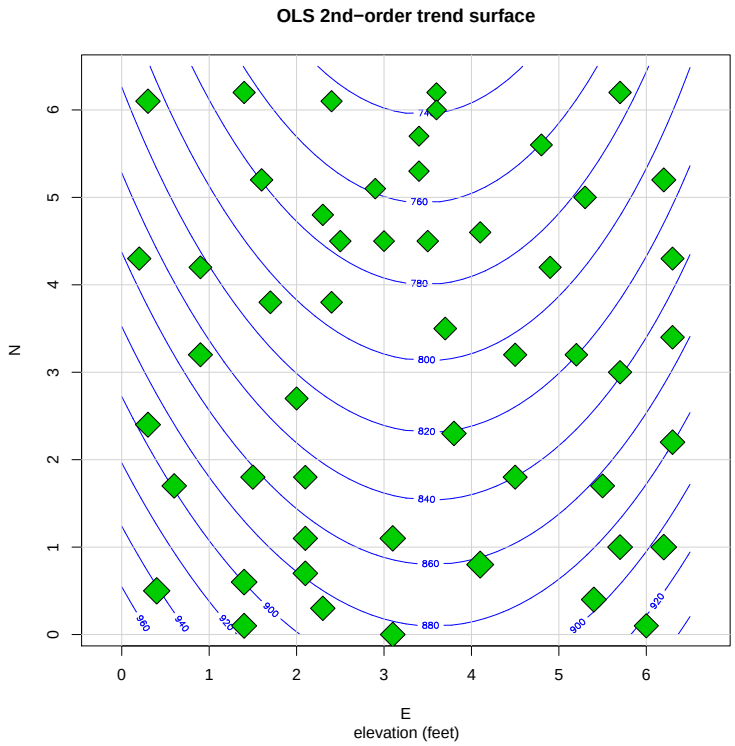
Task 54 : Plot the 2nd-order OLS trend surfaces, with the sample points superimposed. •

The trend surface to be displayed is computed with the `trmat` ("trend matrix") method of the `spatial` package. This just computes the prediction at a regular grid; the number of divisions of the study area in each dimension is given by the `n` argument:

```
> tmat2 <- trmat(ts2, 0, 6.5, 0, 6.5, n = 50)
> str(tmat2)

List of 3
 $ x: num [1:51] 0 0.13 0.26 0.39 0.52 0.65 0.78 0.91 1.04 1.17 ...
 $ y: num [1:51] 0 0.13 0.26 0.39 0.52 0.65 0.78 0.91 1.04 1.17 ...
 $ z: num [1:51, 1:51] 976 970 963 957 951 ...
```

We use the `eqscplot` ("equal scale plot") method of the `MASS` package, as well as the `contour` base graphics method.

```
> eqscplot(tmat2, type = "n", main = "OLS 2nd-order trend surface",
+     sub = "elevation (feet)", xlab = "E", ylab = "N")
> contour(tmat2, add = T, lty = 1, col = "blue")
> grid(lty = 1)
> points(x, y, cex = 3 * (z/max(z)), pch = 23, bg = 3)
```



Q46 : *Describe the visual fit of this trend surface to the sample points.* *Jump to A46* •

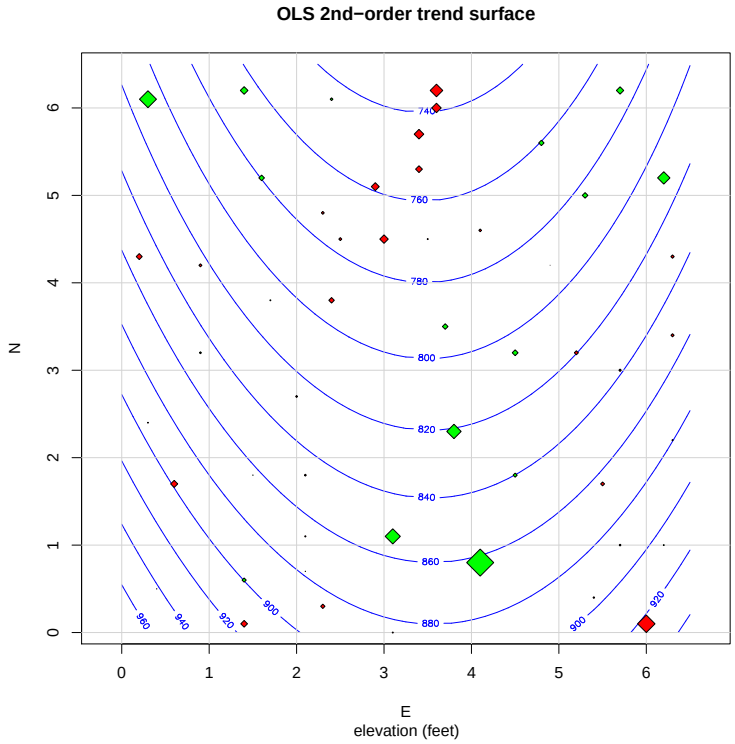
5.4 Step 2: OLS residuals

We now visualize the goodness-of-fit with the residuals.

Task 55 : Plot the 2nd-order OLS trend surfaces, with the residuals superimposed. •

First, we repeat the contour plot with superimposed points, but show the residuals as a colour-coded postplot. We colour-code the residuals by their sign (positive or negative) using the `ifelse` method, and make the symbol size proportional to their absolute value (computed with the `abs` method) using the `cex` graphics argument:

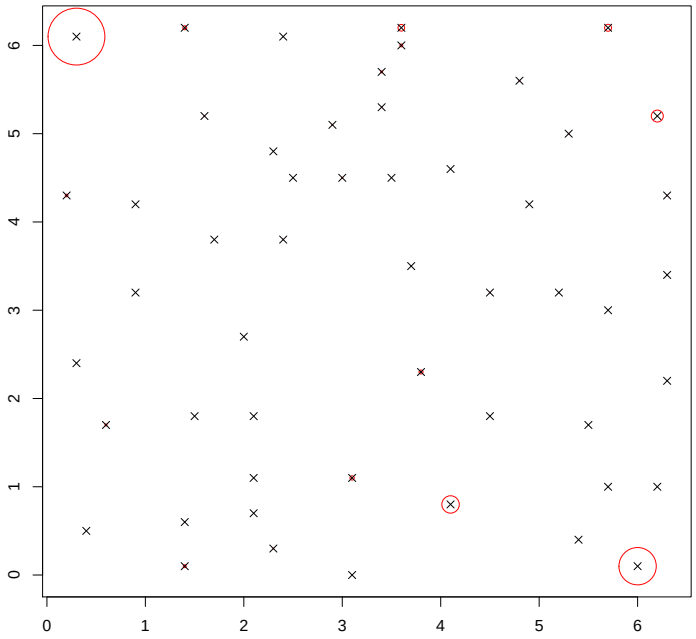
```
> eqscplot(tmat2, type = "n", main = "OLS 2nd-order trend surface",
+     sub = "elevation (feet)", xlab = "E", ylab = "N")
> contour(tmat2, add = T, lty = 1, col = "blue")
> grid(lty = 1)
> r <- residuals(ts2)
> points(x, y, cex = 3 * abs(r)/max(abs(r)), pch = 23,
+     bg = ifelse(r < 0, "red", "green"))
> rm(r)
```



Q47 : *Do the residuals appear to be spatially correlated?* *Jump to A47* •

Second, the `spatial` package has a `plot.trls` method, which is automatically called by the generic `plot` method, which shows the residuals as a post-plot, and also highlights the ones with the most **influence** on the fit.

```
> plot(ts2)
```



To see what is going on here, we directly compute the influence measures with the `trls.influence` method:

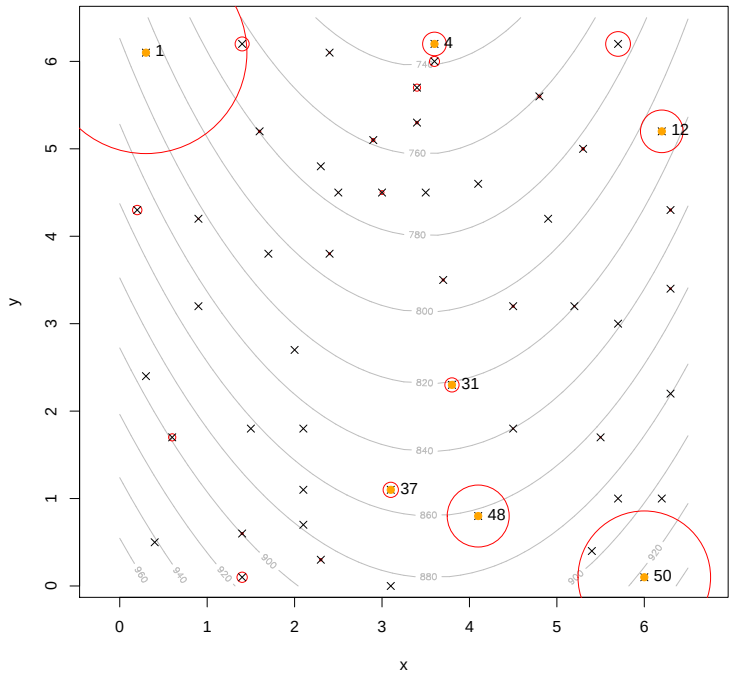
```
> infl.ts2 <- trls.influence(ts2)
> str(infl.ts2)

List of 4
 $ r      : num [1:52]  61.2  27.7  10.1 -45.6  26.6 ...
 $ hii    : num [1:52] 0.355 0.190 0.124 0.135 0.284 ...
 $ stresid: num [1:52]  2.586  1.043  0.366 -1.663  1.066 ...
 $ Di     : num [1:52] 0.61275 0.04259 0.00317 0.07189 0.07518 ...
```

The most interesting field here is `stresid`, which is the **standardized** residual (i.e. = 1 for one standard deviation). We can extract the records with the highest absolute values of the standardized residuals, say > 1.5:

```
> (cand <- as.data.frame(infl.ts2)[abs(infl.ts2$stresid) >
+     1.5, ])

         r      hii stresid       Di
1   61.219 0.354768  2.5859 0.612751
4  -45.585 0.134933 -1.6629 0.071889
12  44.717 0.210223  1.7072 0.129304
31  52.056 0.071542  1.8330 0.043150
37  54.759 0.069748  1.9263 0.046371
48  97.755 0.085741  3.4688 0.188073
50 -63.251 0.275301 -2.5210 0.402378

> rm(infl.ts2)
```

Now we can make this influence plot much more informative, showing the high standardized residuals in colour and labelling them:

```
> eqscplot(tmat2, type = "n")
> contour(tmat2, add = TRUE, col = "grey")
> plot.trls(ts2, add = TRUE, div = 2)
> cand.xy <- topo[as.integer(rownames(cand)), c("x", "y")]
> points(cand.xy, pch = 16, col = "orange")
> text(cand.xy, labels = rownames(cand.xy), pos = 4, offset = 0.5)
```




```
> rm(cand, cand.xy)
```

Q48 : *Which points have the highest standardized residuals? Is there any spatial pattern to these?* *Jump to A48* •

5.5 Step 3: Determining the covariance structure

The question here is, are the residuals spatially-correlated? If so, the OLS fit is not valid and we must re-fit with GLS.

In the previous subsection we suspected some correlation at short ranges. In the next subsection (§5.6) we will use the `surf.gls` method to compute the GLS trend surface; this is like `surf.ls` but also needs a **correlation structure**. To determine this, we need to examine the **correllogram**, which is a plot of the correlation coefficient versus separation between the residuals at point-pairs.

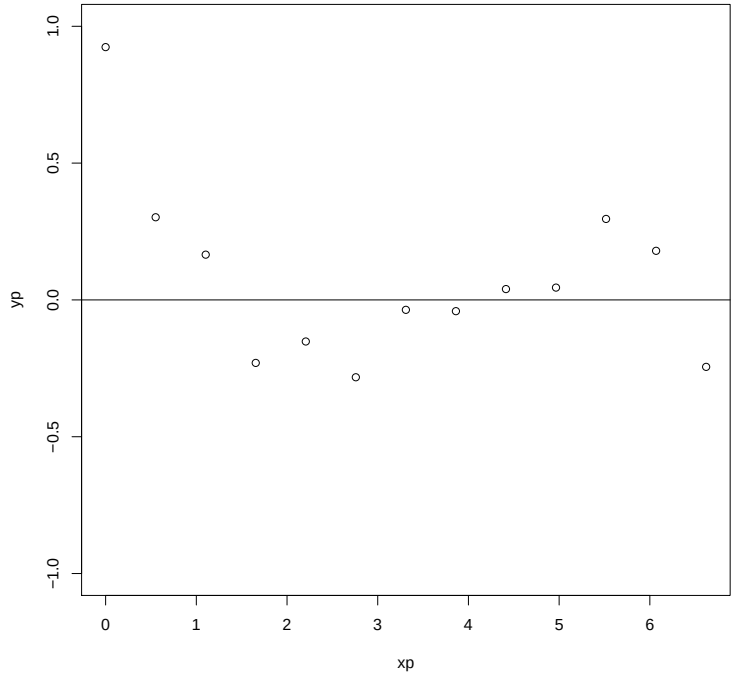
Task 56 : Compute and display the correlogram of the residuals. •

We use the `correlogram` method of the `spatial` package, also specifying

the number of bins with the `nint` argument. This method automatically computes for the residuals, once a trend surface is fit, as it was for object `ts2`, above:

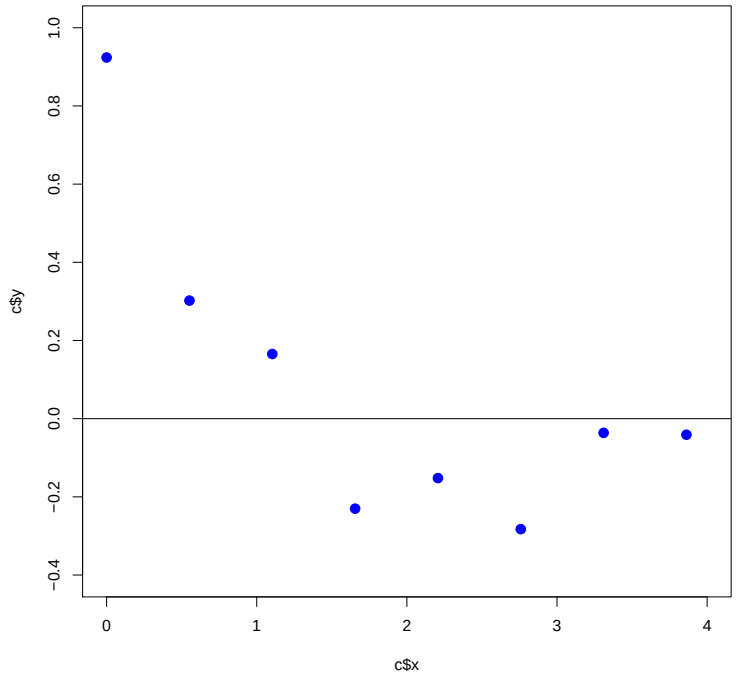
```
> c <- correlogram(ts2, nint = 16, plotit = T)
> as.data.frame(c)
```

```
         x         y cnt
1  0.00000  0.923919  65
2  0.55172  0.302105  73
3  1.10345  0.165462 107
4  1.65517 -0.230165 137
5  2.20690 -0.151937 137
6  2.75862 -0.282654 145
7  3.31035 -0.036221 155
8  3.86207 -0.041093 148
9  4.41380  0.039679 139
10 4.96552  0.045055 108
11 5.51725  0.295976  85
12 6.06897  0.179504  52
13 6.62069 -0.244652  19
```



The default plot shows spatial correlation on a $[-1 \ldots +1]$ scale; in this case the negative correlation never exceeds -0.4, so we should re-draw the correlogram with a reduced scale and also emphasize the short range. Note that after about 4 distance units the number of point-pairs decreases, so the estimate is not reliable. Also, the short-range structure is most important for computing the weights.

```
> plot(c, ylim = c(-0.4, 1), xlim = c(0, 4), col = "blue",
+     pch = 20, cex = 2)
> abline(h = 0)
```

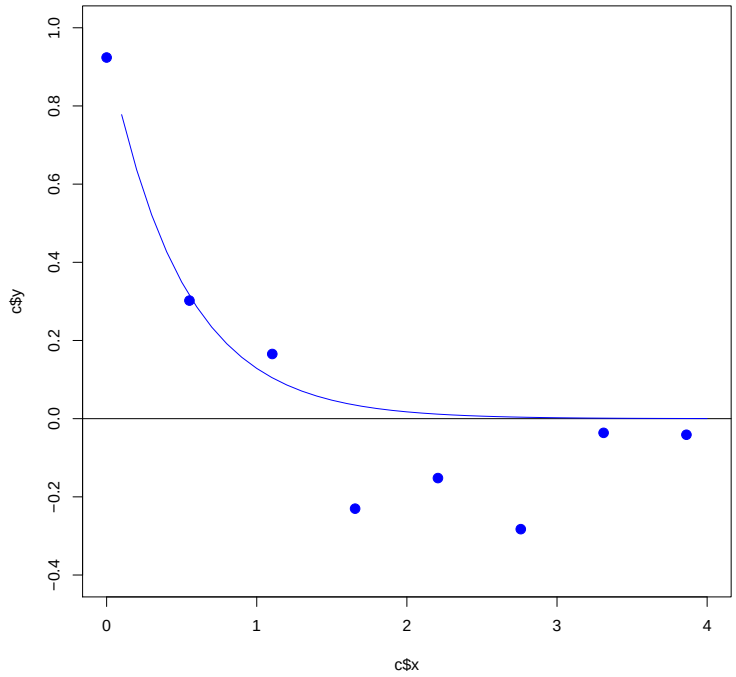


Q49 : *Describe the spatial correlation (strong, moderate, weak?). What is the approximate range? What is the nugget effect as a proportion of the maximum correlation (1)? Are there enough point-pairs to make a reliable model?* *Jump to A49* •

Task 57 : Fit a covariance function to the correlogram. •

The `spatial` package provides only three models: exponential (`expcov`), Gaussian (`gaucov`) and spherical (`sphercov`). Here there is no evidence of a Gaussian structure ("halo" effect at short ranges). We try the exponential model with an effective range of 1.5 units (i.e. range parameter `d` = 0.5) and a nugget (parameter `alpha`) of 0.05, which seems to fit fairly well:

```
> plot(c, ylim = c(-0.4, 1), xlim = c(0, 4), col = "blue",
+     pch = 20, cex = 2)
> abline(h = 0)
> d <- seq(0.1, 4, by = 0.1)
> lines(d, expcov(d, d = 0.5, alpha = 0.05), col = "blue")
```



There is no automatic fitting in `spatial`, unlike the variogram model fitting we saw above in `gstat`. However, the details of the model are not very important; any reasonable model will give quite similar corrections to the trend surface.

5.6 Step 4: Computing the GLS trend surface

Now that we have a model of spatial correlation we can compute the GLS trend surface, using this covariance function.

Task 58 : Compute and the 2nd-order GLS trend surface and compare it with the 2nd-order OLS trend surface. •

We use the `surf.gls` method of the `spatial` package:

```
> ts2.g <- surf.gls(2, expcov, d = 0.5, alpha = 0.05, x,
+     y, z)
```

Q50 : *How much did the internal goodness-of-fit (R^2) change from the OLS to the GLS surface?* *Jump to A50* •

```
> summary(ts2.g)

Analysis of Variance Table
 Model: surf.gls(np = 2, covmod = expcov, x = x, y = y, z = z, d = 0.5,     alpha = 0.05)
           Sum Sq Df  Mean Sq F value   Pr(>F)
Regression 154145  5 30829.01  33.858 2.44e-14
Deviation   41885 46   910.54
```

```
Total      196030 51
Multiple R-Squared: 0.786,        Adjusted R-squared: 0.763
AIC: (df = 6) 359.95
Fitted:
   Min     1Q Median     3Q    Max
   748    784    830    870    937
Residuals:
   Min     1Q Median     3Q    Max
-59.42 -18.31  -5.36  11.12  96.59

> summary(ts2)

Analysis of Variance Table
 Model: surf.ls(np = 2, x = x, y = y, z = z)
           Sum Sq Df  Mean Sq F value   Pr(>F)
Regression 156072  5 31214.31  35.934 8.44e-15
Deviation   39958 46   868.66
Total      196030 51
Multiple R-Squared: 0.796,        Adjusted R-squared: 0.774
AIC: (df = 6) 357.51
Fitted:
   Min     1Q Median     3Q    Max
   736    775    826    870    942
Residuals:
   Min     1Q Median     3Q    Max
-63.25 -14.36  -4.87  15.00  97.75
```

Q51 : *How much did the coefficients change from the OLS to the GLS surface?* *Jump to A51* •

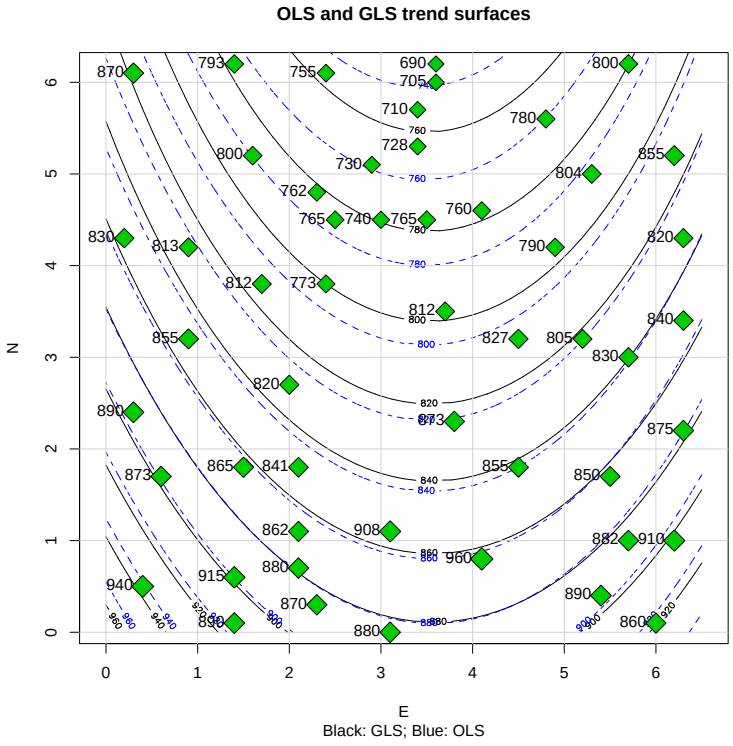
```
> ts2.g$beta

[1] 806.8856 -11.7446  62.0064 -67.8814   1.5970   8.9836

> ts2$beta

[1] 801.2176 -11.0189  68.2291 -73.9930   3.3436   8.3427

> 100 * (ts2.g$beta - ts2$beta)/ts2$beta

[1]   0.70742   6.58628  -9.12039  -8.25967 -52.23816   7.68139
```

Q52 : *How much did the residuals change from the OLS to the GLS surface?* *Jump to A52* •

```
> summary(residuals(ts2.g))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -59.40  -18.30   -5.36   -3.81   11.10   96.60

> summary(residuals(ts2))

     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-6.33e+01 -1.44e+01 -4.87e+00 -1.97e-14  1.50e+01  9.78e+01
```

Task 59 : Plot the GLS and OLS trend surfaces together, to visualize the difference. •

```
> tmat2.g <- trmat(ts2.g, 0, 6.5, 0, 6.5, n = 50)
> eqscplot(ts2, type = "n", main = "OLS and GLS trend surfaces",
+     xlab = "E", ylab = "N", sub = "Black: GLS; Blue: OLS")
> contour(tmat2.g, level = seq(720, 960, 20), add = T)
> contour(tmat2, level = seq(720, 960, 20), add = T, lty = 2,
+     col = "blue")
> grid(lty = 1)
> points(x, y, cex = z * 2.5/max(z), pch = 23, bg = 3)
> text(x, y, z, pos = 2)
```



Q53 : *Describe the differences between the OLS and GLS surfaces.* *Jump to A53* •

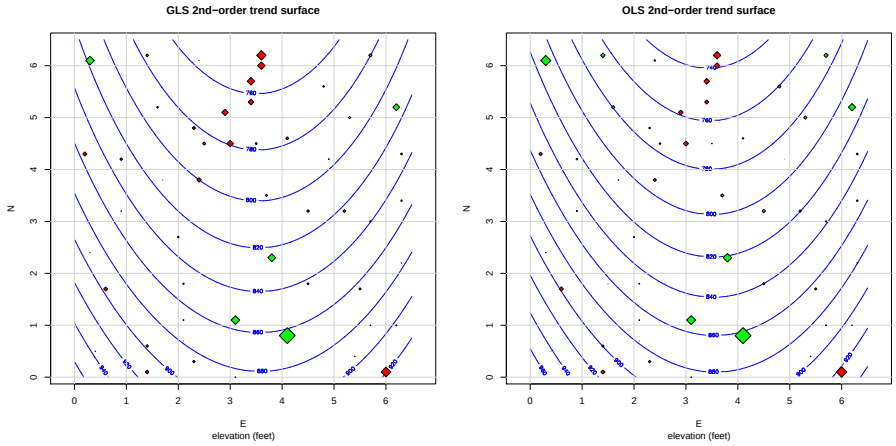
Now we see if the residuals from the GLS surface are spatially-independent.

Task 60 : Plot the 2nd-order GLS trend surfaces, with the residuals superimposed. •

We put the same plot for the OLS side-by-side, for comparaison.

```
> par(mfrow = c(1, 2))
> r <- residuals(ts2)
> r.g <- residuals(ts2.g)
> r.max <- max(abs(r), abs(r.g))
> eqscplot(tmat2.g, type = "n", main = "GLS 2nd-order trend surface",
```

```
+     sub = "elevation (feet)", xlab = "E", ylab = "N")
> contour(tmat2.g, add = T, lty = 1, col = "blue")
> grid(lty = 1)
> points(x, y, cex = 3 * abs(r.g)/r.max, pch = 23, bg = ifelse(r <
+     0, "red", "green"))
> eqscplot(tmat2, type = "n", main = "OLS 2nd-order trend surface",
+     sub = "elevation (feet)", xlab = "E", ylab = "N")
> contour(tmat2, add = T, lty = 1, col = "blue")
> grid(lty = 1)
> points(x, y, cex = 3 * abs(r)/r.max, pch = 23, bg = ifelse(r <
+     0, "red", "green"))
> rm(r, r.g, r.max)
> par(mfrow = c(1, 1))
```



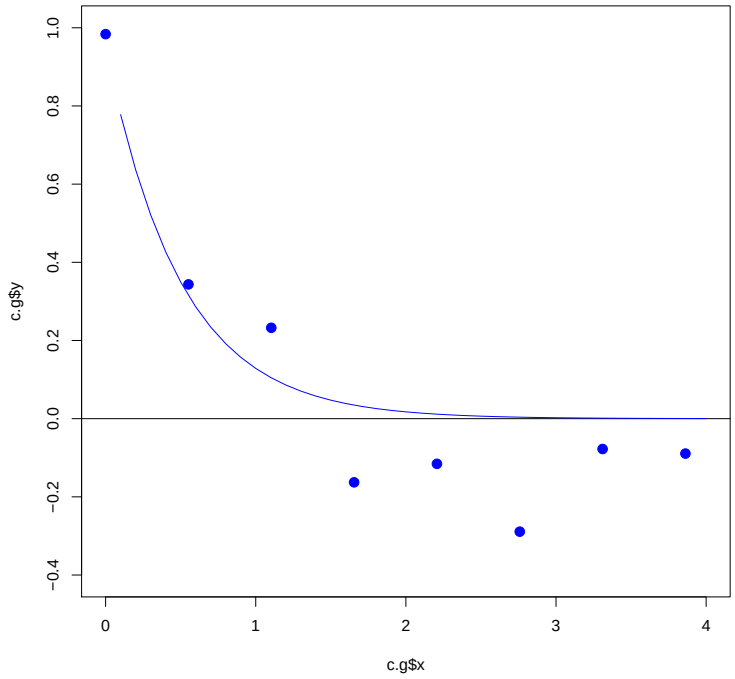
Q54 : *How is the spatial distribution of the residuals affected by a GLS fit, rather than an OLS fit?* *Jump to A54* •

5.7 Step 5: Re-computing spatial correlation

We check to see if there is still spatial correlation among the residuals, and if so, does it differ from that estimated from the OLS surface? If so, we have to interatively re-fit the GLS surface with the revised correlation structure.

Task 61 : Compute and plot the correlogram of the residuals from the GLS surface, with the spatial correlation model from the OLS residuals superimposed. •

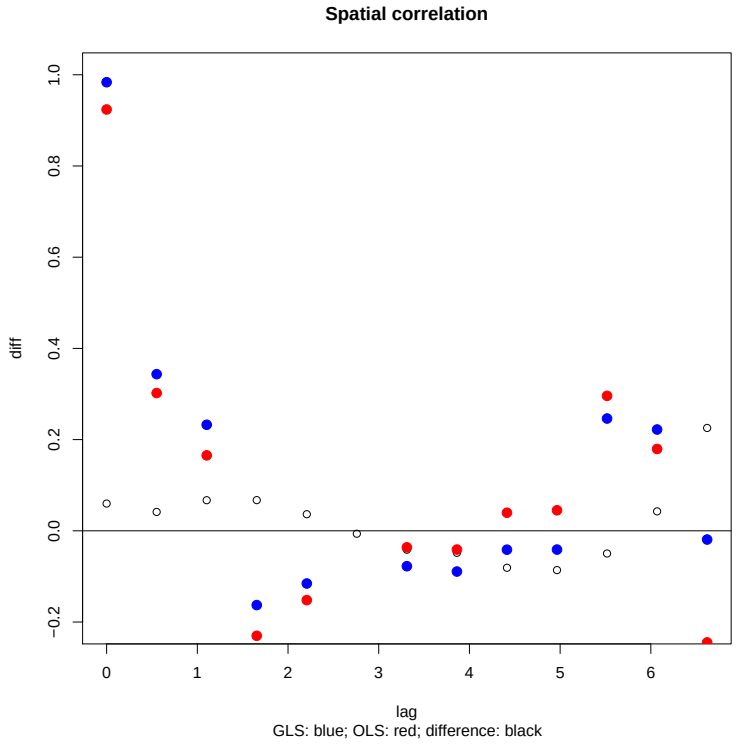
```
> c.g <- correlogram(ts2.g, nint = 16, plotit = F)
> plot(c.g, ylim = c(-0.4, 1), xlim = c(0, 4), col = "blue",
+     pch = 20, cex = 2)
> abline(h = 0)
> d <- seq(0.1, 4, by = 0.1)
> lines(d, expcov(d, d = 0.5, alpha = 0.05), col = "blue")
> rm(d)
```



Q55 : *Does the model from the OLS residuals appear to fit the GLS residuals?* *Jump to A55* •

Task 62 : Plot the correlograms of the OLS and GLS residuals on one graph, also showing their differences. •

```
> tmp <- as.data.frame(cbind(lag = c.g$x, gls.corr = c.g$y,
+     ols.corr = c$y, diff = c.g$y - c$y))
> plot(diff ~ lag, data = tmp, ylim = c(-0.2, 1), pch = 1,
+     main = "Spatial correlation", sub = "GLS: blue; OLS: red; difference: black")
> points(tmp$lag, tmp$gls.corr, col = "blue", pch = 20,
+     cex = 2)
> points(tmp$lag, tmp$ols.corr, col = "red", pch = 20,
+     cex = 2)
> abline(h = 0)
> rm(tmp)
```



Q56 : *Is there much difference between the two correlograms?* *Jump to A56* •

Q57 : *Is it necessary to iteratively fit the GLS surface?* *Jump to A57* •

Task 63 : Clean up the workspace from the previous three subsections. •

```
> detach(topo)
> rm(topo, ts2, ts2.g, tmat2, tmat2.g, c, c.g)
```

5.8 Answers

A44 : *`x` and `y` are local survey coördinates in units of 50 feet*[4]*; `z` is elevation above an unspecified base in feet.* *Return to Q44* •

A45 : *The lowest elevation is at the centre-top and increases south and eastward. It seems to be a bowl with the lowest point at centre-top.* *Return to Q45* •

A46 : *The fit appears quite good. The bowl shape adjusts well to the sample*

[4] 1' = 0.3048 m

points. Return to Q46 •

A47 : *There is clear spatial correlation: the negative residuals are mostly near the bottom of the bowl (top centre) and the positive residuals elsewhere; the larger absolute values also seem to be clustered.* Return to Q47 •

A48 : *The largest standardized residuals (points 1, 50, 48, 12) are towards the outside of the bowl.* Return to Q48 •

A49 : *The spatial correlation is strong to about 1.5 50 foot units (the range); if there is a nugget effect it is quite small, perhaps 0.05. There are not many point-pairs, especially at short ranges. If we use fewer bins to get more point-pairs, we lose resolution.* Return to Q49 •

A50 : *The* R^2 *decreased (got worse), from* 0.774 *(OLS) to* 0.763 *(GLS). This is fairly common, and means that the OLS estimate was too optimistic. Spatially-correlated points pulled the surface towards the clusters. Once spatial correlation is used for weighting, these are not so influential.* Return to Q50 •

A51 : *The coefficients changed significantly, mostly by about 8% relative.* Return to Q51 •

A52 : *The GLS residuals have a somewhat narrower range (smaller extremes); the mean residual is no longer* ≈ 0. Return to Q52 •

A53 : *The "bowl" is not so deep: note the first contour line for the OLS fit is 740, but for GLS this is not reached. The OLS surface looks more realistic near the top centre, but the GLS surface is better over much of the centre of the map.* Return to Q53 •

A54 : *The GLS fit has higher residuals in the "bowl" near the top centre; the spatial correlation appears to be as strong as for OLS.* Return to Q54 •

A55 : *Yes, the variogram model fitted to the OLS residuals fits the GLS residuals, although not quite as well as it fits the OLS residuals.* Return to Q55 •

A56 : *There is very little difference in the correlograms. The GLS residuals have slightly higheer spatial correlation to 2 distance units, mainly due to a reduced nugget (short-range correlation closer to 1).* Return to Q56 •

A57 : *There is no reason to iteratively fit the surface, especially because changing the nugget alone does not affect the weights derived from the covariance matrix.* Return to Q57 •

6 Self-test

This section is a small self-test of how well you mastered this exercise. You should be able to complete the tasks and answer the questions with the knowledge you have gained from the exercise. Please **submit your answers (including graphical output) to the instructor** for grading and sample answers.

For this self-test we continue with the Jura dataset, which should already be loaded from the exercise (§4).

Regional spatial structure

Task 1 : Compute a **first-order trend surface** of the cobalt content of the 259 soil samples in the calibration dataset `jura.cal`. •

> **Note:** The two coördinates for the right-hand side of the `lm` method can be extracted from the spatial object with the `coordinates` method, i.e. `coordinates(jura.cal)`. You can use this as the right-hand side of the linear model formula.

Q1 : *How much of the spatial variation is explained by this trend surface?* •

Task 2 : Display feature-space diagnostics of the linear model. •

Q2 : *Are the residuals approximately normally-distributed, as required by the linear modelling assumption?* •

Q3 : *Are the residuals independent of the fitted value, as required by the linear modelling assumption?* •

Task 3 : Display a post-plot of the residuals. •

> **Note:** The `bubble` method requires a spatial object. You will first have to build a data frame with just the residuals from the trend surface object; then convert this frame to a spatial object by assigning the coordinates with the `coordinates` method, copying the coördinates from `jura.cal`.

Q4 : *Describe the spatial pattern of the residuals.* •

Q5 : *Would a higher-order surface fit this attribute better than the first-order surface? Why or why not?* •

Local spatial structure

Task 4 : Compute and plot the experimental variogram for the base-10 logarithm (method `log10`) of the zinc (Zn) values in the calibration data set; adjust the cutoff to the approximate range. •

Q6 : *Does the variogram appear to reach a definite sill within this cutoff, or is it asymptotic to a sill, or is it unbounded?* •

Q7 : *What is the approximate range, sill and nugget of the experimental variogram?* •

Task 5 : Select a model form and adjust the fit by eye; plot the experimental variogram with your best fitted model. •

Q8 : *Which model form did you select, and why?* •

Task 6 : Fit the model to the experimental variogram with the default fitting method of `gstat`; plot the experimental variogram with the automatically-fitted model. •

Q9 : *What are the fitted parameters of the variogram model?* •

Task 7 : Remove temporary objects from the workspace. •

References

[1] N Cressie. *Statistics for spatial data*. John Wiley & Sons, revised edition, 1993. 53

[2] J C Davis. *Statistics and data analysis in geology*. John Wiley & Sons, New York, 3rd edition, 2002. 54

[3] P Goovaerts. *Geostatistics for natural resources evaluation*. Applied Geostatistics. Oxford University Press, New York; Oxford, 1997. 19

[4] B Minasny and A B McBratney. The Matérn function as a general model for soil variograms. *Geoderma*, 128(3-4):192–207, 2005. 28

[5] E J Pebesma. *gstat User's Manual*. Department of Physical Geography, Utrecht University, Utrecht, version 2.3.3 edition, 2001. 27

[6] B D Ripley. *Spatial statistics*. John Wiley and Sons, New York, 1981. 52, 54

[7] W N Venables and B D Ripley. *Modern applied statistics with S*. Springer-Verlag, New York, fourth edition, 2002. 52, 54

Index of R Concepts

: operator, 24
[[]] operator, 45
[] operator, 19

abs, 57
attributes, 34

barplot, 10
bbox (package:sp), 22
bubble (package:lattice), 13
bubble (package:sp), 5, 15, 70

c, 43
cex graphics argument, 57
contour, 56
coordinates (package:sp), 14, 15, 70
correlogram (package:spatial), 60

diff, 10
dim, 20

eqscplot (package:lattice), 56
expcov (package:spatial), 62

fit.variogram (package:gstat), 30, 32, 34, 43
fitted, 3
for operator, 45, 46
function, 22

gaucov (package:spatial), 62
gls, 54
gstat package, 1, 2, 25–28, 30, 32, 42, 63

hist, 10, 24

I, 15
ifelse, 57
intersection, 21
is.element, 21

krige (package:gstat), 2

lattice package, 13
lines, 22, 32
lm, 2, 14, 53, 70
loc method argument, 25
log10, 71

MASS package, 52, 54, 56
matrix, 10
max, 12
meuse dataset, 41
model method argument, 29, 32
more lattice graphics argument, 13

n method argument, 56
nlme package, 54

pch graphics argument, 5
plot, 29, 32, 46, 58
plot graphics argument, 10
plot.gstatVariogram (package:gstat), 29, 32
plot.trls (package:spatial), 58
points, 46
pos method argument, 32
print (package:lattice), 13
print, 13
print.trellis (package:lattice), 13

residuals, 3

sample, 20, 45
seq, 24
setdiff, 21
setequal, 21
show.vgms (package:gstat), 26
sp package, 1, 2, 5, 13–15, 41
spatial package, 52, 54, 56, 58, 60, 62, 63
sphercov (package:spatial), 62
split lattice graphics argument, 13
spplot (package:sp), 13
step, 7, 8
summary, 10
surf.gls (package:spatial), 60, 63
surf.ls (package:spatial), 55, 60

text, 32
topo dataset, 54
trellis class, 13
trls.influence (package:spatial), 59
trmat (package:spatial), 56
type graphics argument, 46

union, 21

variogram (package:gstat), 25, 26
variogramLine (package:gstat), 32

vector, 45
vgm (package:gstat), 27, 29, 40, 42, 43

xyplot (package:lattice), 13