

---

## Applied geostatistics

### Exercise 3: Modelling spatial structure from point samples

---

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

January 3, 2014

#### Contents

<b>1</b>	<b>introduction</b>	<b>1</b>
<b>2</b>	<b>Fitting trend surfaces</b>	<b>1</b>
2.1	Fitting a first-order surface . . . . .	2
2.2	Fitting a second-order surface . . . . .	7
2.3	Simplifying the higher-order surface . . . . .	8
2.4	* Comparing first- and higher-order surfaces . . . . .	12
2.5	* Linear models on sp objects . . . . .	15
2.6	Answers . . . . .	18
<b>3</b>	<b>Calibration and evaluation datasets</b>	<b>19</b>
3.1	* Random selection of evaluation points . . . . .	21
3.2	Answers . . . . .	24
<b>4</b>	<b>Fitting variogram models</b>	<b>25</b>
4.1	Selecting a variogram model . . . . .	26
4.2	Fitting a variogram model . . . . .	29
4.3	* Visual comparison of several variogram fits . . . . .	32
4.4	* Goodness-of-fit . . . . .	34
4.5	* Fitting models with multiple structures . . . . .	38
4.6	* Fitting anisotropic variograms . . . . .	42
4.7	* The effect of sampling on variogram modelling . . . . .	46
4.8	Answers . . . . .	51

---

Version 2.4 Copyright © 2006-2010, 2012, 2014 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>).

<b>5</b>	<b>* Fitting a trend surface with generalized least squares</b>	<b>53</b>
5.1	Theoretical background . . . . .	54
5.2	The ‘topo’ sample dataset . . . . .	55
5.3	Step 1: OLS trend surface . . . . .	55
5.4	Step 2: OLS residuals . . . . .	58
5.5	Step 3: Determining the covariance structure . . . . .	61
5.6	Step 4: Computing the GLS trend surface . . . . .	64
5.7	Step 5: Re-computing spatial correlation . . . . .	67
5.8	Answers . . . . .	69
<b>6</b>	<b>Self-test</b>	<b>71</b>
	<b>References</b>	<b>73</b>
	<b>Index of R concepts</b>	<b>74</b>

不入虎穴，焉得虎子

“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 evaluation (also called “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 properties 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 •*

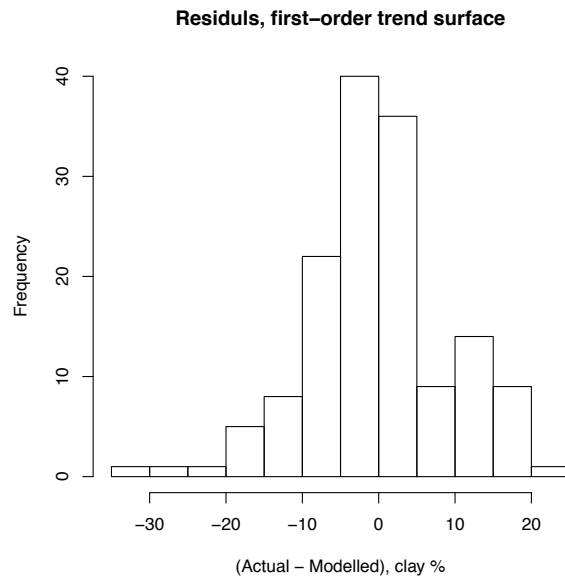
First we examine the performance of the model in **feature** (attribute) space, ignoring for the moment that these are geographic data.

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

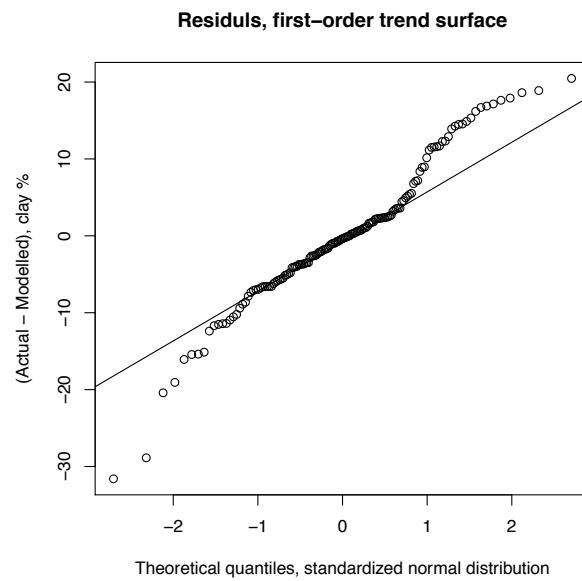
---

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

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



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

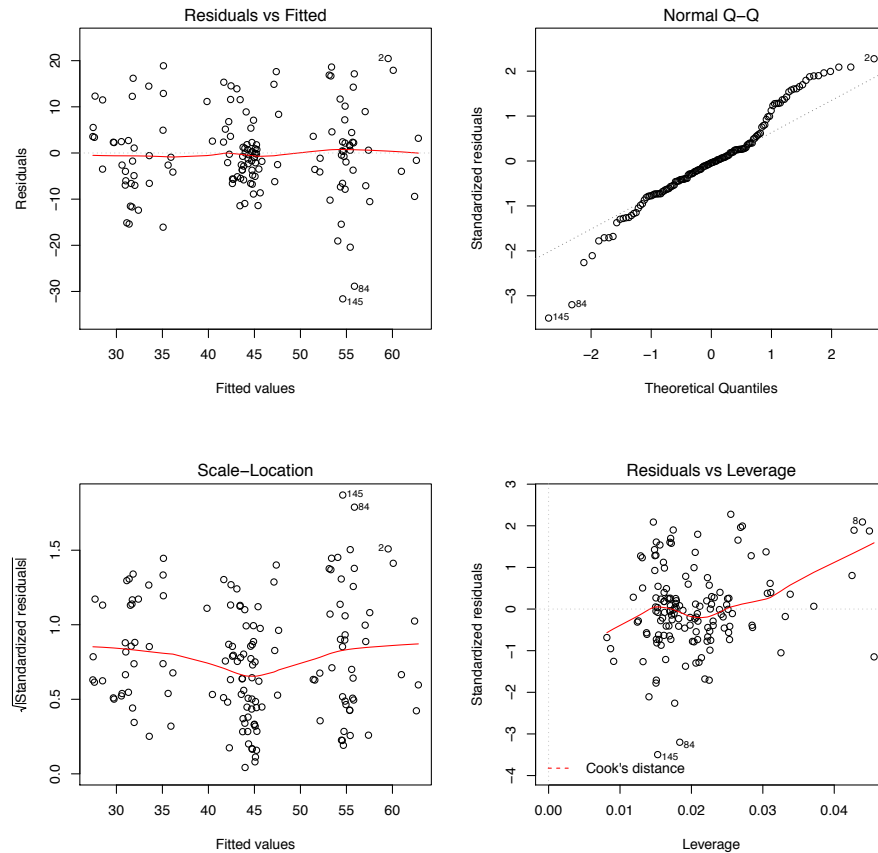



---

**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.

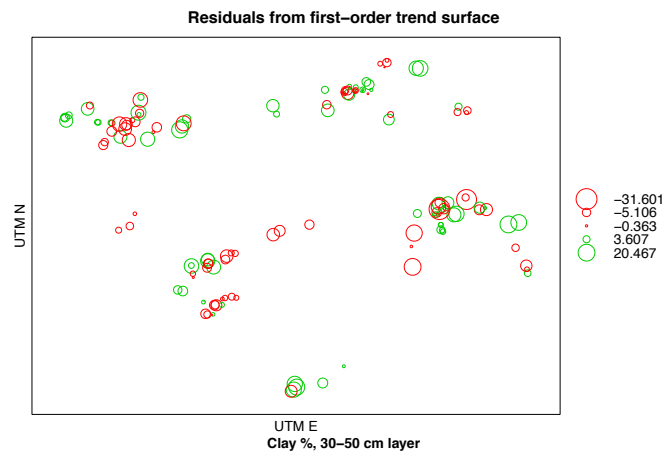
---

**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. This allows for an oriented bowl or dome shape, whereas the first surface only allows an oriented plane.

---

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

**Note:** The “full” surface includes an interaction term, which allows the principal axis of the second-order surface to be oriented obliquely to the coordinate axes. This is a special case of **polynomial regression** in two dimensions.

We fit the model and summarize it:

```
> 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 (the null hypothesis of no effect)?* *Jump to A8 •*

---

## 2.3 Simplifying the higher-order surface

The equation can likely be simplified without loss of fit. The interaction term and both E coordinate terms seem not to be significant. We try removing these and compare the fit. We are looking for the most **parsimonious** equation as measured by the Akaike Information Criterion (AIC); in general (but not always) this will also result in the highest **adjusted  $R^2$** .

---

**Task 8 :** Simplify the second-order trend surface by removing the interaction term, and compare the AIC and adjusted  $R^2$ . •

**Note:** Removing the interaction term says that any trend (first or second order) is aligned with the coordinate axes. A significant interaction term rejects this null hypothesis. In general a full second-order surface is fitted, but in this geographic setting it may be that the coordinate axes are aligned with the axes of the best-fit ellipsoid.

We use the **update** method, a shorthand for changing a model formula. Note the use of the dot (.) formula operator to represent the existing left-hand and right-hand sides of the formula being updated. Here we update the **ts2** model by removing a named term; the removal is symbolized by the minus (-) formula operator.

```

> ts2.step <- update(ts2, . ~ . - I(UTM_E * UTM_N))
> summary(ts2.step)

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

```

```

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(UTM_E^2)    8.41e-09   6.34e-09    1.33   0.1870
I(UTM_N^2)    3.57e-08   1.26e-08    2.83   0.0054 **
UTM_E        -1.08e-02   8.64e-03   -1.25   0.2138
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

> AIC(ts2, ts2.step)

      df      AIC
ts2      7 1068.9
ts2.step  6 1067.6

```

---

**Q9 :** *Is the model improved by removing the interaction term?* [Jump to A9](#) •

In the E direction, there are both linear and quadratic terms, both marginally significant. This indicates that the quadratic is not improving the linear in this coördinate direction. If we remove the quadratic, the linear may well be significant, and thus be included in the model.

**Note:** It is not advisable to remove the linear term and leave the quadratic in a polynomial regression model, even if the linear term is not statistically significant. Removal of the linear term corresponds to the hypothesis that the surface is centred on 0; in general this is clearly not true. The linear term effectively centres the surface in that direction (given the intercept); it allows for additive translation, e.g., adding a fixed amount to the coördinate (i.e., change of coördinate system) without affecting the significance of the linear term.

---

**Task 9 :** Simplify this equation by removing the quadratic term for E, i.e.,  $E^2$ ; compare the AIC and adjusted  $R^2$ . •

```

> ts2.step2 <- update(ts2.step, . ~ . - I(UTM_E^2))
> summary(ts2.step2)

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

```

Residuals:

Min	1Q	Median	3Q	Max
-29.86	-4.87	-1.03	3.88	22.02

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	2.78e+03	1.22e+03	2.28	0.024 *
I(UTM_N^2)	2.82e-08	1.13e-08	2.49	0.014 *
UTM_E	6.66e-04	5.89e-05	11.29	<2e-16 ***
UTM_N	-1.90e-02	7.44e-03	-2.55	0.012 *

---

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

Residual standard error: 8.95 on 143 degrees of freedom

Multiple R-squared: 0.526, Adjusted R-squared: 0.516

F-statistic: 53 on 3 and 143 DF, p-value: <2e-16

> AIC(ts2.step, ts2.step2)

	df	AIC
ts2.step	6	1067.6
ts2.step2	5	1067.4

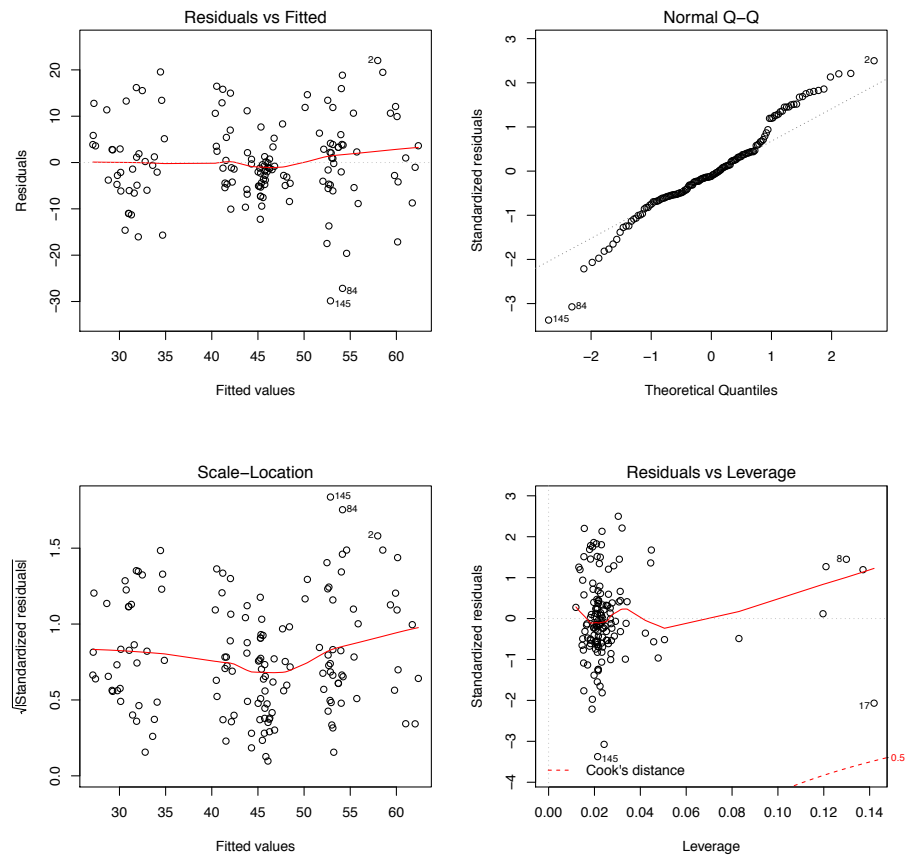
---

**Q10 :** *Is the model improved by removing the quadratic term for East? Is the linear term for E now significantly different from zero (i.e., no E-W trend)?* [Jump to A10](#) •

---

**Task 10 :** Plot the regression diagnostics for the fitted second-order surface, in both feature and geographic space. •

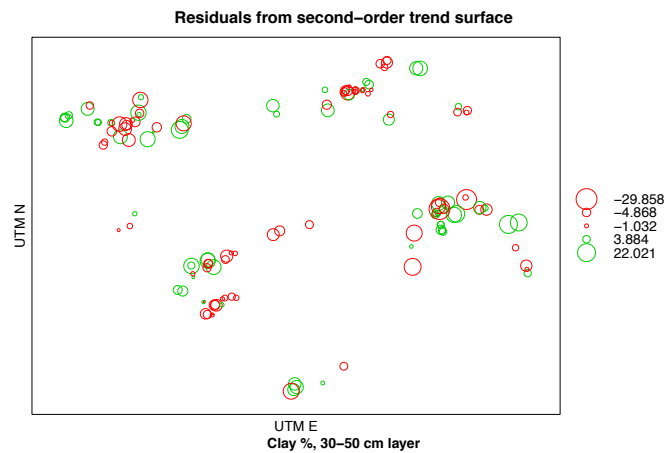
```
> par(mfrow = c(2, 2))
> plot(ts2.step2)
> par(mfrow = c(1, 1))
```



```

> tmp <- data.frame(e = tcp$UTM_E,
+                   n = tcp$UTM_N, residuals = residuals(ts2.step2))
> 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 11 :** Compare the diagnostics from the second- and first-order surfaces.

---

**Q11 :** *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 A11 •

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` with the `matrix` method:

```
> summary(residuals(ts1))

  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-31.600 -5.110  -0.363   0.000  3.610  20.500

> summary(residuals(ts2.step2))

  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -29.90  -4.87   -1.03    0.00   3.88   22.00

> diff(range(residuals(ts1)))

[1] 52.067

> diff(range(residuals(ts2.step2)))
```

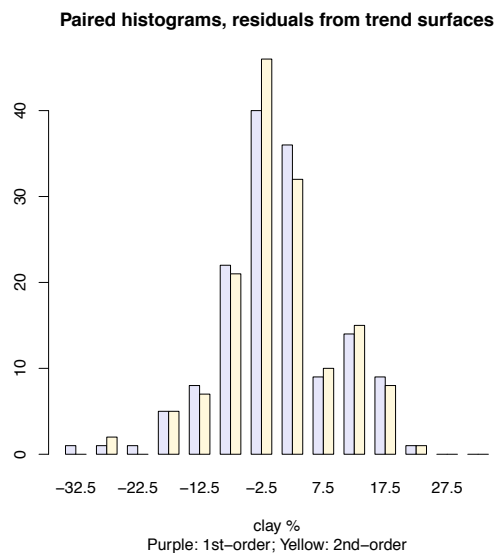
```
[1] 51.88
```

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

```
List of 6
```

```
$ 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 ...
$ 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.step2), 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 12 :** 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$resid2 <- residuals(ts2.step2)
> tmp$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] 19.47 22.02 3.62 -1.03 -8.72 ...
.. ..$ resid.diff: num [1:147] 1.559 1.555 0.435 0.569 0.684 ...
..@ coords.nrs : int [1:2] 1 2
..@ coords     : num [1:147, 1:2] 702638 701659 703488 703421 703358 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:147] "1" "2" "3" "4" ...
.. .. ..$ : 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

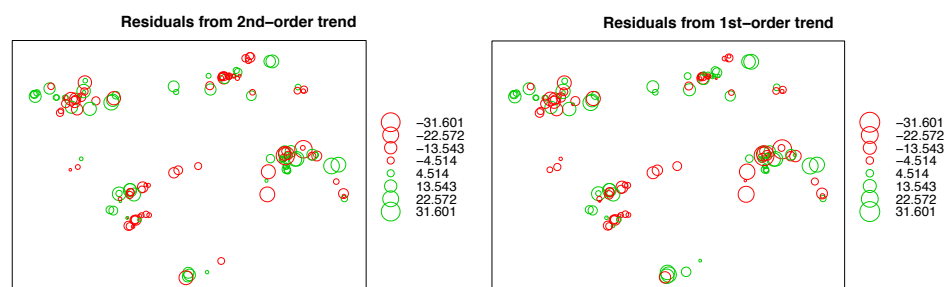
```

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

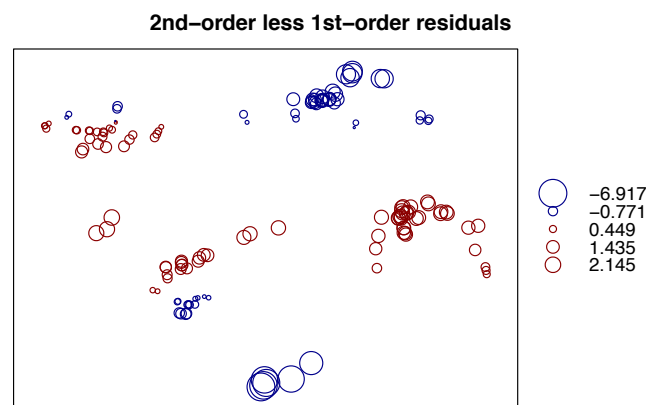


`trellis` which are produced by calls to `lattice` functions like `xyplot` or `bubble`; the `spplot` method of the `sp` package also uses lattice graphics and hence `print.trellis`. In interactive use it is called automatically when a `trellis` object is produced. However, here we call it explicitly 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")))
```




---

**Q12 :** 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 A12 •*

---

**Task 13 :** Clean up the workspace from this section. •

```
> rm(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 14 :** 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
.. .. ..$ : chr [1:147] "1" "2" "3" "4" ...
.. .. ..$ : 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
.. .. ..@ proj4args: chr NA
```

---

**Task 15 :** 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

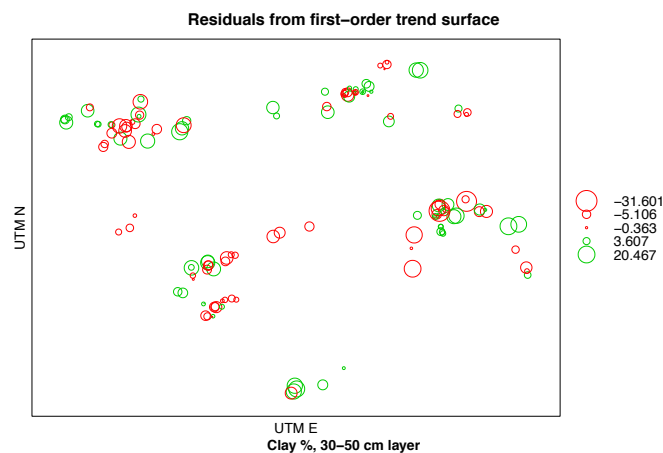
The results of this model are identical to the one where we specified the coördinates directly.

---

**Task 16 :** Display a poost-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 17 :** 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 trickier; 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.

## 2.6 Answers

---

**A1 :** Clay =  $-250.3 + 0.00065 \cdot \text{UTM}_E + -0.00045 \cdot \text{UTM}_N$  [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 and linear N terms and the intercept are significant at  $p < 0.1$ . Both quadratic and linear E terms are marginal. The interaction term  $I(UTM\_N * UTM\_E)$  is almost surely not significant. [Return to Q8](#) •

---

**A9** : Yes, it is slightly improved: the adjusted  $R^2$  is 0.001 higher and the AIC 1.4 lower. [Return to Q9](#) •

---

**A10** : This is a close call. The adjusted  $R^2$  is 0.003 lower (slightly worse), but the AIC is 0.2 lower, i.e., slightly better. The E term is now highly significant. [Return to Q10](#) •

---

**A11** : (1) Slightly better goodness-of-fit (adjusted  $R^2 = 51.6\%$  vs.  $49.9\%$ ) and lower (better) AIC; (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. In this study area the trend is not so strong, but does explain about half of the variability. [Return to Q11](#) •

---

**A12** : 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 Q12](#) •

---

### 3 Calibration and evaluation 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 **evaluate** the assumptions – are they justified?

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

Calibration : to model the spatial structure;

Evaluation : of the model's success.

We prefer the term **evaluation** to the more common term **validation**, because the latter term implies the model is completely valid.

A typical split is approximately 2/3 calibration, 1/3 evaluation. The Jura data set contains 359 observations; in previous studies with this same dataset the last 100 of these have been held out for evaluation, 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 **evaluation** set `jura.val`, made up of the remaining 100 observations from the full set. •

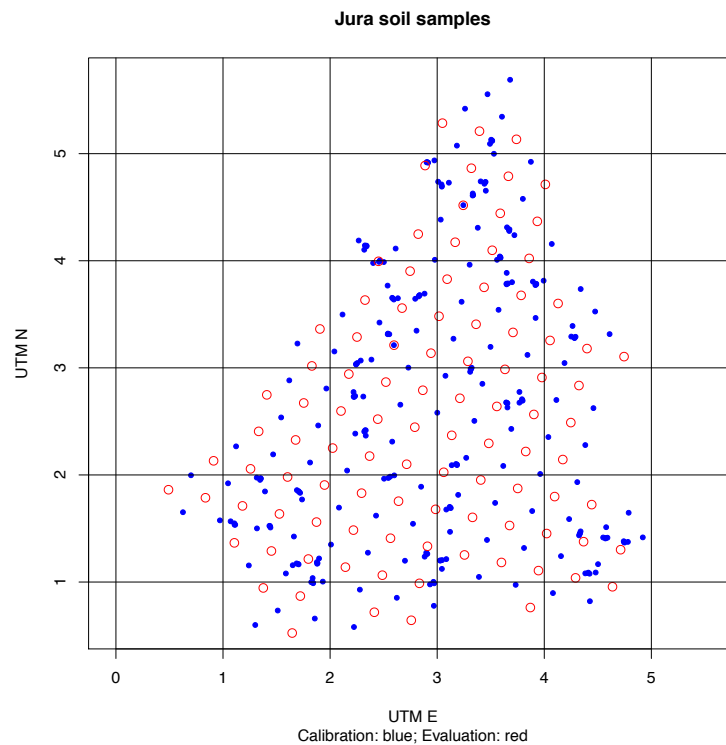
This is just row selection from a matrix using the `[]` operator, which also works with the `@data` slot `sp` object:

```
> 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; Evaluation: red")
> points(coordinates(jura.val), col = "red", pch = 1, cex = 1.2)
> grid(lty = 1, col = "black")
```




---

**Q13 :** What is the spatial distribution of the calibration and evaluation points? Do they equally represent the study area? Jump to A13 •

### 3.1 \* Random selection of evaluation points

This **optional** subsection discusses selection of evaluation 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 evaluation 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] 3 13 19 21 22 23 27 32 34 37 42 43 48 49 52 58
[17] 60 63 67 68 79 81 82 85 87 93 106 108 111 115 116 121
[33] 125 130 137 138 141 142 144 147 154 156 158 159 162 165 167 168
[49] 174 182 186 193 194 196 198 199 203 205 206 207 221 229 230 231
[65] 232 233 237 238 240 241 246 249 257 258 262 263 267 270 272 276
[81] 279 288 301 303 305 310 314 319 325 333 334 335 339 340 341 346
[97] 347 352 356 357

```

---

**Q14 :** *Your list of selected observations will be different; why?* [Jump to A14](#) •

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",...: 1 2 1 3 3 1 3 2 2 ...
 .. ..$ Land: Factor w/ 4 levels "Forest","Pasture",...: 3 1 3 2 3 3 3 2 3 4 ...
 .. ..$ Cd  : num [1:100] 0.24 2.566 4.495 0.685 1.12 ...
 .. ..$ Cu  : num [1:100] 20.72 7.88 10.24 10.92 20.72 ...
 .. ..$ Pb  : num [1:100] 22.4 55.2 32.9 30.8 41.2 ...
 .. ..$ Co  : num [1:100] 3.76 10.6 10.32 11.72 11.92 ...
 .. ..$ Cr  : num [1:100] 18.9 32.8 39.2 31.9 34.8 ...
 .. ..$ Ni  : num [1:100] 5.2 23.6 22.6 13.1 20 ...
 .. ..$ Zn  : num [1:100] 32.2 59.2 101.7 49.3 83.2 ...
 ..@ coords.nrs : int [1:2] 1 2
 ..@ coords     : num [1:100, 1:2] 2.5 4.13 2.9 3.87 1.71 ...
 .. ..- attr(*, "dimnames")=List of 2
 .. .. ..$ : NULL
 .. .. ..$ : chr [1:2] "X" "Y"
 ..@ bbox       : num [1:2, 1:2] 0.701 0.58 4.92 5.13
 .. ..- 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 evaluation set. We determine this with the `is.element` set operator<sup>1</sup>; we select the complement of the set with the `!` “not” logical operator.

---

<sup>1</sup> The other set operators are `union`, `intersection`, `setdiff` and `setequal`



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

```
Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
..@ data      : 'data.frame':      259 obs. of  9 variables:
.. ..$ Rock: Factor w/ 5 levels "Argovian","Kimmeridgian",...: 3 2 2 5 5 5 1 1 3
.. ..$ Land: Factor w/ 4 levels "Forest","Pasture",...: 3 2 3 3 3 3 3 3 3 ...
.. ..$ Cd  : num [1:259] 1.74 1.33 2.15 1.56 1.15 ...
.. ..$ Cu  : num [1:259] 25.7 24.8 22.7 34.3 31.3 ...
.. ..$ Pb  : num [1:259] 77.4 77.9 56.4 66.4 72.4 ...
.. ..$ Co  : num [1:259] 9.32 10 11.92 16.32 3.5 ...
.. ..$ Cr  : num [1:259] 38.3 40.2 43.5 38.5 40.4 ...
.. ..$ Ni  : num [1:259] 21.3 29.7 29.7 26.2 22 ...
.. ..$ Zn  : num [1:259] 92.6 73.6 90 88.4 75.2 ...
..@ coords.nrs : int [1:2] 1 2
..@ coords      : num [1:259, 1:2] 2.39 2.54 4.31 4.38 3.24 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:2] "X" "Y"
..@ bbox        : num [1:2, 1:2] 0.491 0.524 4.788 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
```

---

**Q15 :** *Do the two sets cover the same area?*

*Jump to A15 •*

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 0.701 4.92
Y 0.580 5.13

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

```

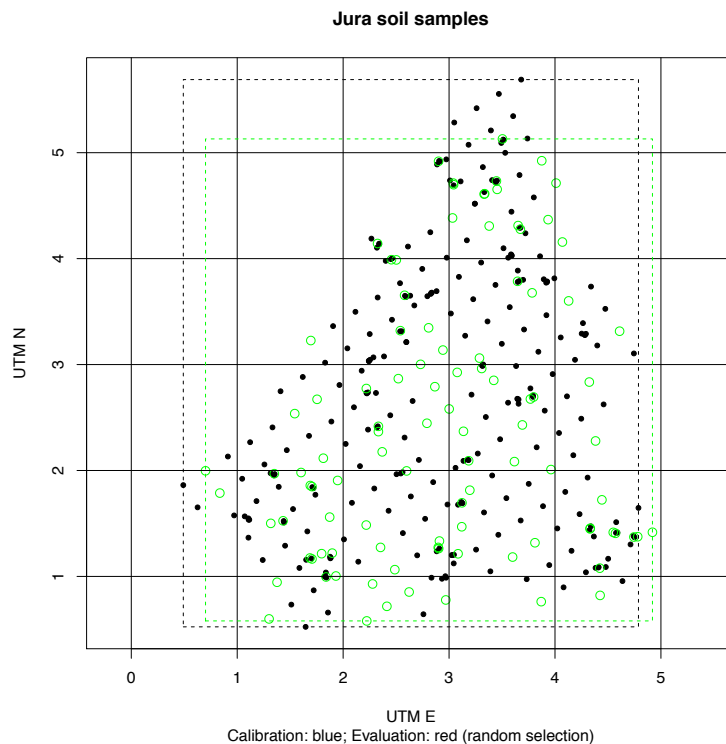
      min    max
X 0.491 4.788
Y 0.524 5.690

```

```

> plot(coordinates(jura.cal.rand), col = "black", asp = 1,
+       pch = 20, xlab = "UTM E", ylab = "UTM N", main = "Jura soil samples",
+       sub = "Calibration: blue; Evaluation: 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 evaluation 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

---

**A13 :** The evaluation 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 Q13* •

---

**A14 :** *It is a random sample.*

*Return to Q14 •*

---

**A15 :** *This depends on the random sample, but in general the evaluation set will be contained in a somewhat smaller bounding box. They both cover the area; because of the random selection of the smaller evaluation set it may be some spatial clusters and holes.*

*Return to Q15 •*

## 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).

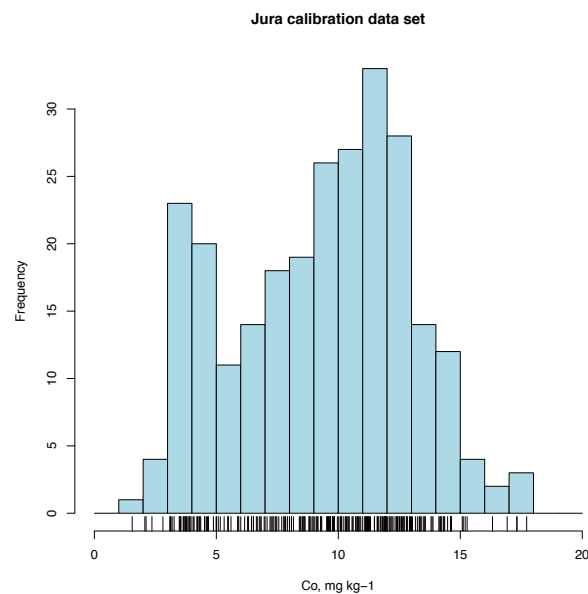
---

**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.

---

**Q16 :** *Describe the feature-space distribution of Co in the calibration dataset. Would a transformation make this more symmetrically-distributed?*  
*Jump to A16 •*

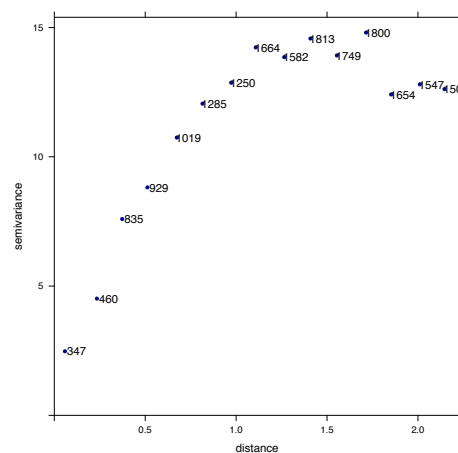
So we compute the empirical variogram of the untransformed variable Co.

---

**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 coordinates 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**.

---

**Q17 :** *Does this variogram approach or reach a sill? Approximately what value? At approximately what range?*  
*Jump to A17 •*

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.

---

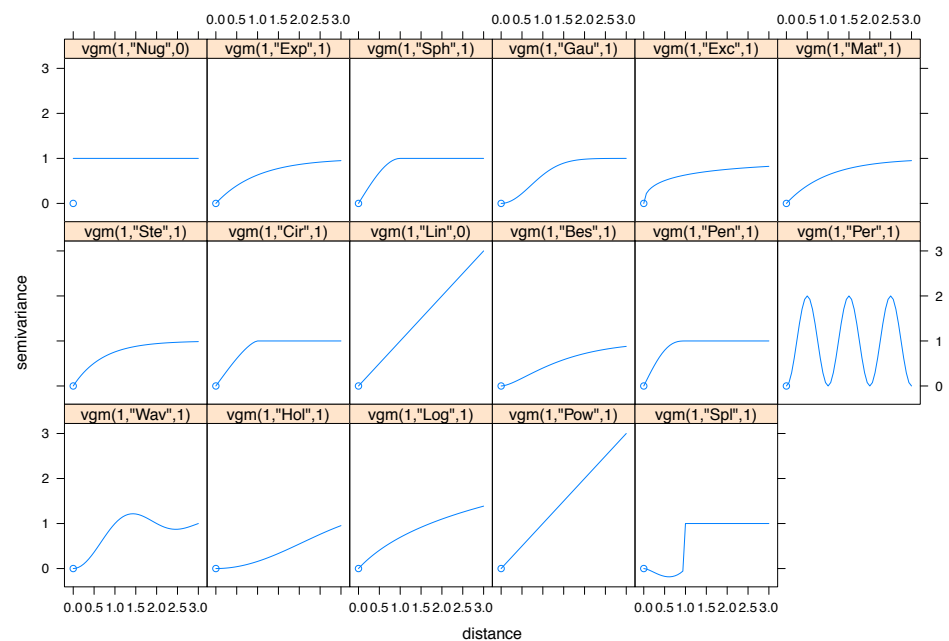
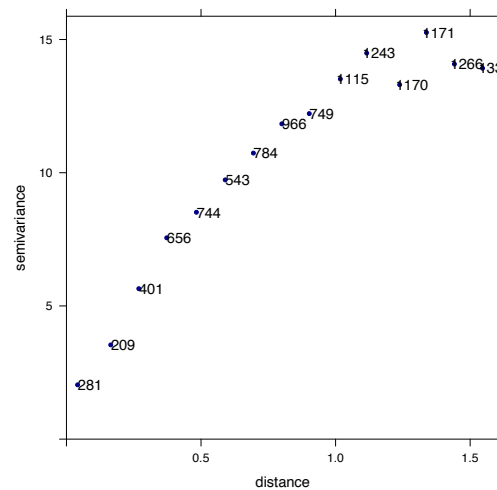
**Q18 :** Does this variogram show point-pairs with more separation than the local spatial dependence? I.e. is the range shorter than the maximum separation shown in the variogram? What would be an appropriate maximum?

*Jump to A18 •*

**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")
```



The **model name** as used by **gstat** is shown above each panel; we can 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  Wav                                Wav (wave)
14  Hol                                Hol (hole)
15  Log                                Log (logarithmic)
16  Pow                                Pow (power)
17  Spl                                Spl (spline)
18  Leg                                Leg (Legendre)
19  Err                                Err (Measurement error)
20  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.

---

**Q19 :** Which **model forms** match this empirical variogram? [Jump to A19](#) •

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.

---

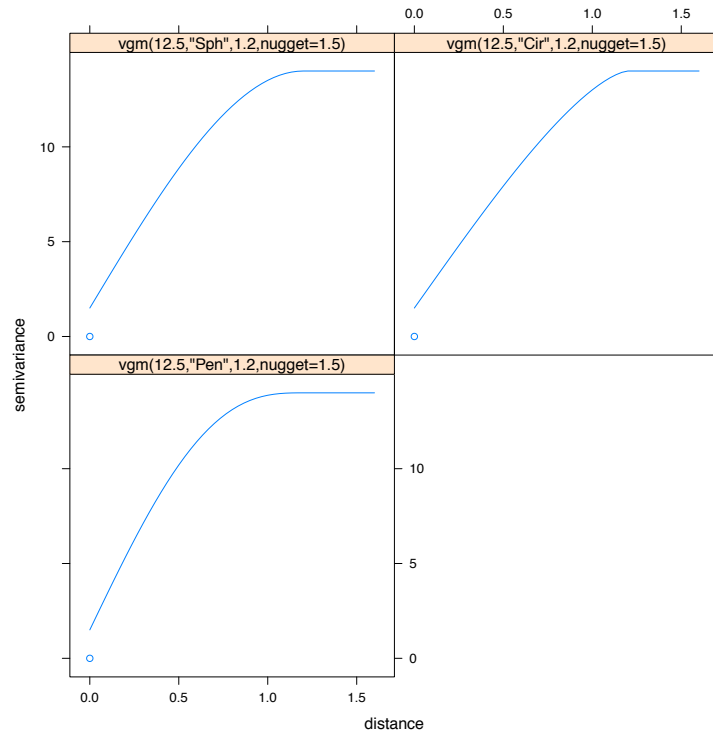
**Q20 :** What are approximate values for range, total sill and nugget? [Jump to A20](#) •

**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 26 :** 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))
```




---

**Q21 :** Which of these three best matches the empirical variogram? [Jump to A21](#) •

## 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 27 :** 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
```

```

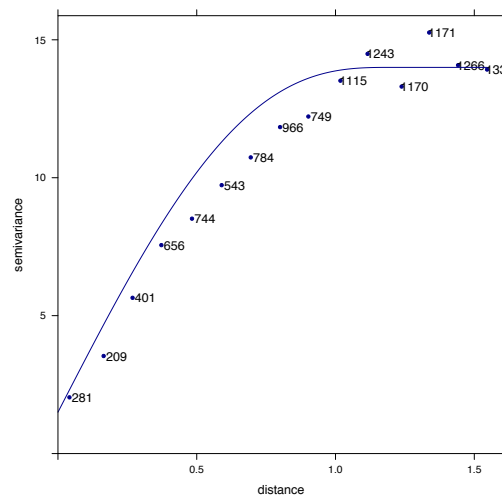
2   Pen  12.5   1.2

> class(v)

[1] "variogramModel" "data.frame"

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

```




---

**Q22 :** *How well does this model, estimated by eye, fit the empirical variogram?*

*[Jump to A22](#)*

•

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 28 :** 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(vmf)

```



```

      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 successful in practice.

---

**Q23 :** *How much did the automatic fit change the estimated fit?* [Jump to A23](#) •

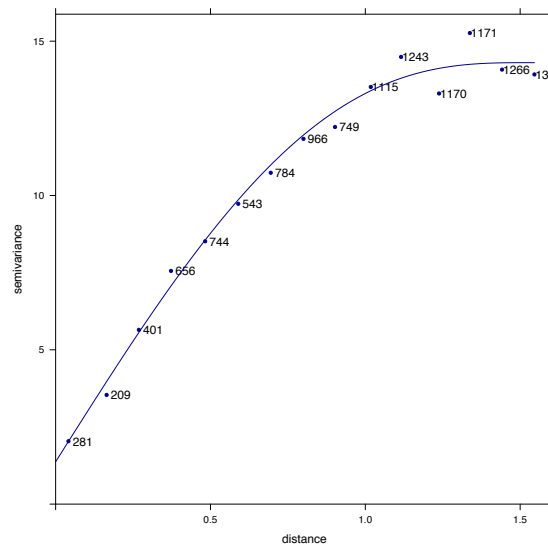
---

**Task 29 :** Plot the empirical variogram with the fitted model superimposed. •

```

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

```




---

**Q24 :** *How well does the fitted model match the empirical variogram?*  
[Jump to A24](#) •

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.

---

**Q25 :** *What proportion of the total variance in Co is explained by the fitted variogram model?*  
[Jump to A25](#) •

```
> 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.

---

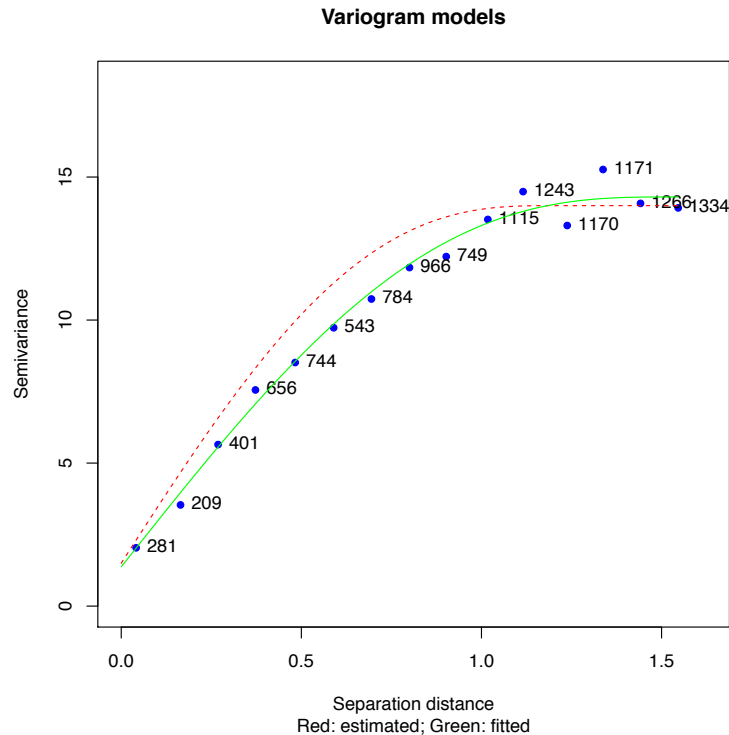
**Task 30 :** Plot empirical variogram with both the estimated and fitted variogram models superimposed, to visualise the effect of the automatic fit. •

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 continuous plots of 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 31 :** Plot empirical 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.

```

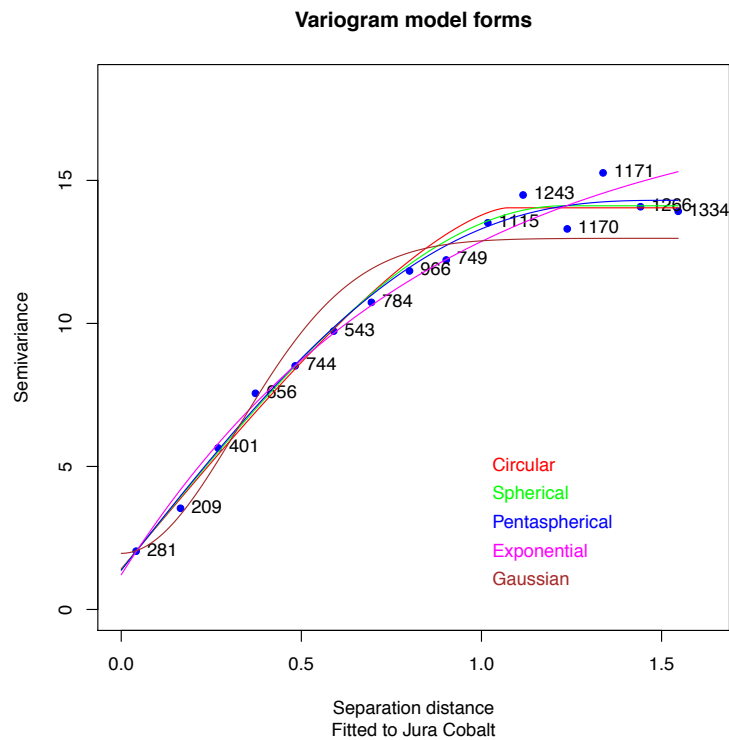
> plot(v$gamma ~ v$dist, xlim = c(0, max(v$dist) * 1.05),
+      ylim = c(0, max(v$gamma) * 1.2), pch = 20, col = "blue",
+      cex = 1.2, xlab = "Separation distance", ylab = "Semivariance",
+      main = "Variogram model forms", sub = "Fitted to Jura Cobalt")
> text(v$dist, v$gamma, v$np, pos = 4)
> 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.

---

**Q26 :** What are the similarities and differences in these model forms, for the models fitted to this empirical variogram? *Jump to A26 •*

#### 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 empirical variogram; this is called **internal goodness-of-fit**.

---

**Task 32 :** 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 instead 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

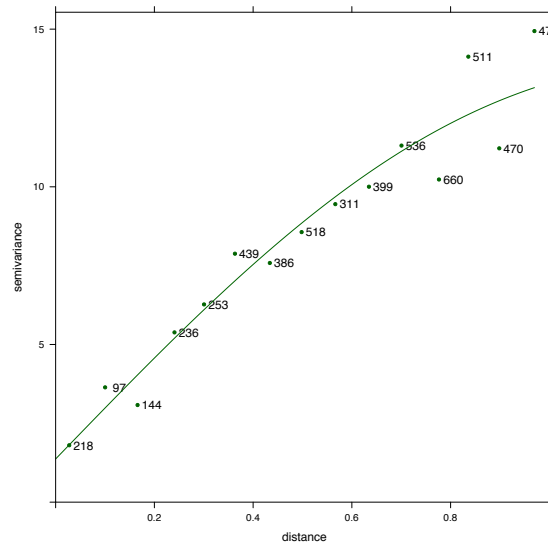
> attributes(vmf)$SSerr

[1] 4803.9

> attributes(vmf2)$SSerr

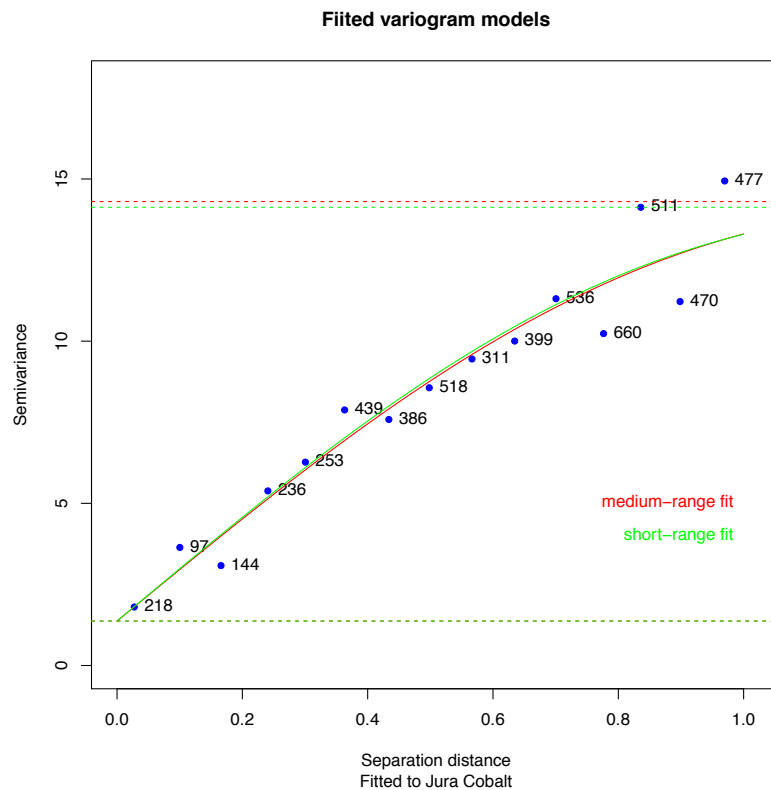
[1] 20569
```

```
> 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 = "Fitted 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)
```




---

**Q27 :** 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 A27*

•

---

**Q28 :** How did the fitting error change? Explain why. Does a larger error in this case mean that the model is not suitable? *Jump to A28* •

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
```

---

**Q29 :** Which of these models has the closest fit? Are there major differences? *Jump to A29*

•

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

---

**Task 33 :** 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.

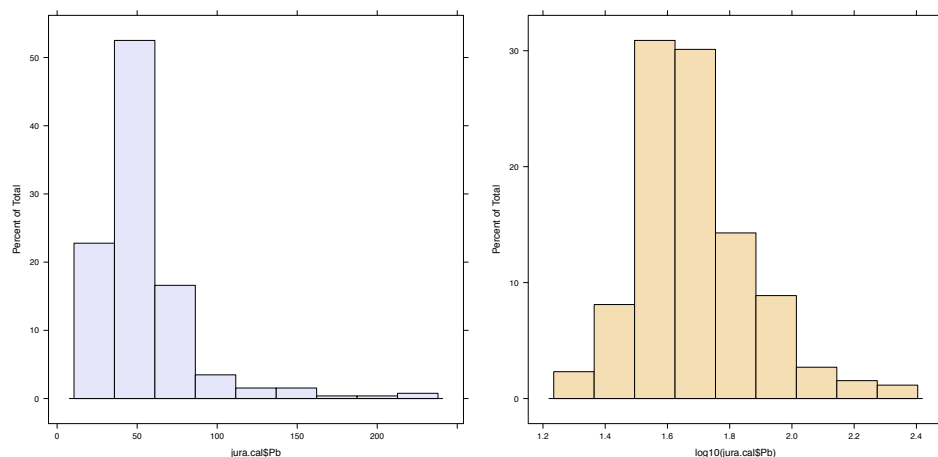
In the Jura dataset, several of the metals exhibit an empirical variogram that suggest multiple structures. One of these is lead (Pb). Since it is strongly right-skewed, we log-transform it to an approximately symmetric form before computing the variogram.

---

**Task 34 :** 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)
```





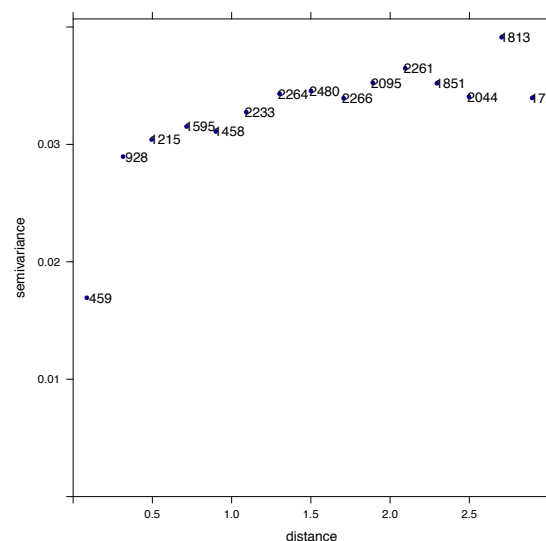

---

**Q30 :** Describe the shapes of these two histograms. Which one should be used to compute semi-variances? *Jump to A30* •

---

**Task 35 :** Compute the empirical variogram of the  $\log_{10}(\text{Pb})$  values and plot it. •

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




---

**Q31 :** What is the evidence for two structures in this empirical variogram? What are their approximate ranges? *Jump to A31* •

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

**Task 36 :** 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))

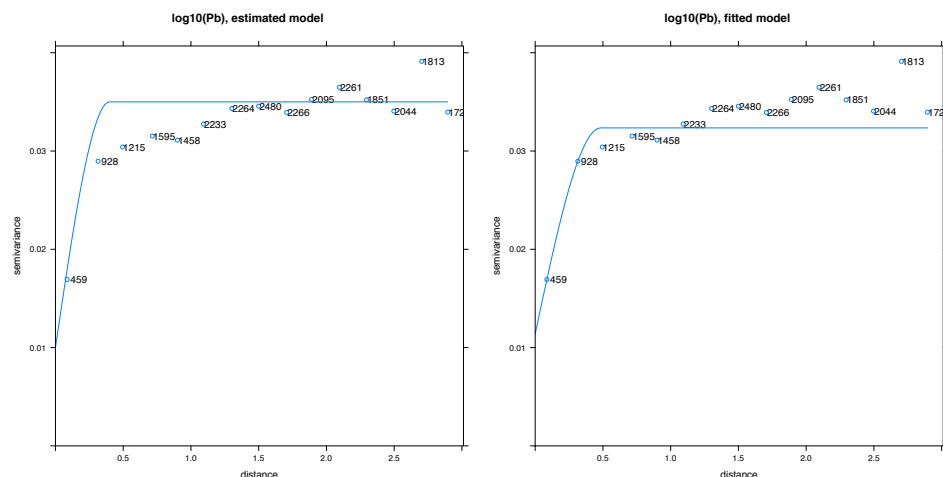
model psill range
1  Nug 0.010  0.0
2  Sph 0.025  0.4

> (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)
```




---

**Q32 :** Describe this fit. What is the range of the model? Does this match the range of the variogram? [Jump to A32](#) •

---

**Q33 :** 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 A33](#) •

---

**Task 37 :** 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))

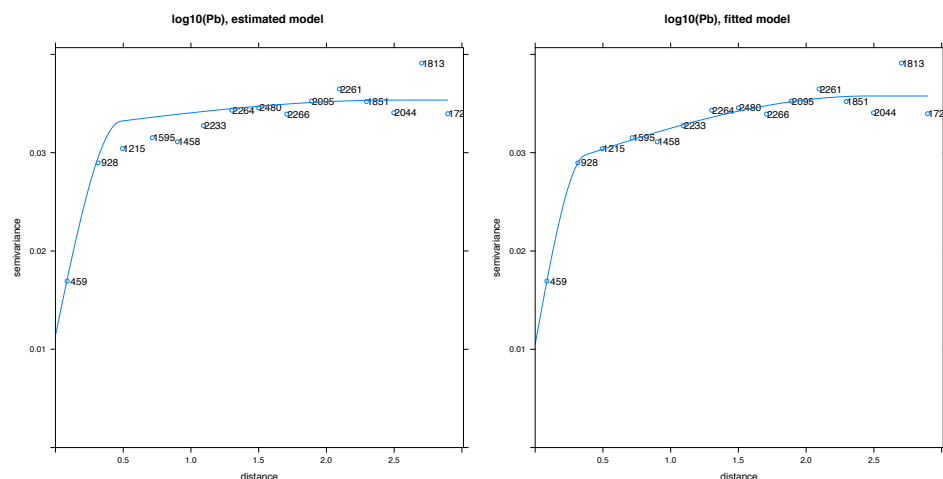
      model    psill  range
1   Nug 0.011386 0.00000
2   Sph 0.020966 0.48408
3   Sph 0.003000 2.50000

> (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)
```




---

**Q34 :** Describe this fit. What are the two ranges? What does this fitted model imply about the spatial processes? *Jump to A34 •*

---

**Q35 :** What are the relative proportions of spatial variability accounted for by the three model components (two structures and nugget)? [Jump to A35](#) •

```
> 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 38 :** 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 ellipse** 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.

---

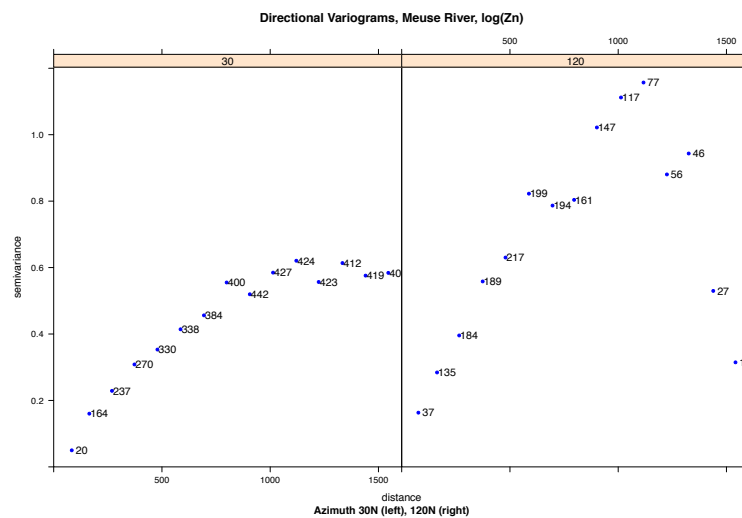
**Task 39 :** Load the `meuse` sample dataset from the `sp` package and convert it from a dataframe to a spatial object by specifying its coordinates. •

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

---

**Task 40 :** 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 ellipse. •

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




---

**Q36 :** What variogram model form do you suggest to fit the variogram of the suspected major axis (30 ° N)? *Jump to A36* •

---

**Q37 :** Does it appear that the sills are the same for the major and minor axes? *Jump to A37* •

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 ellipse flattening).

---

**Q38 :** *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 A38 •

---

**Task 41 :** Fit a spherical variogram model, with geometric anisotropy, to the directional variograms and display the fit. •

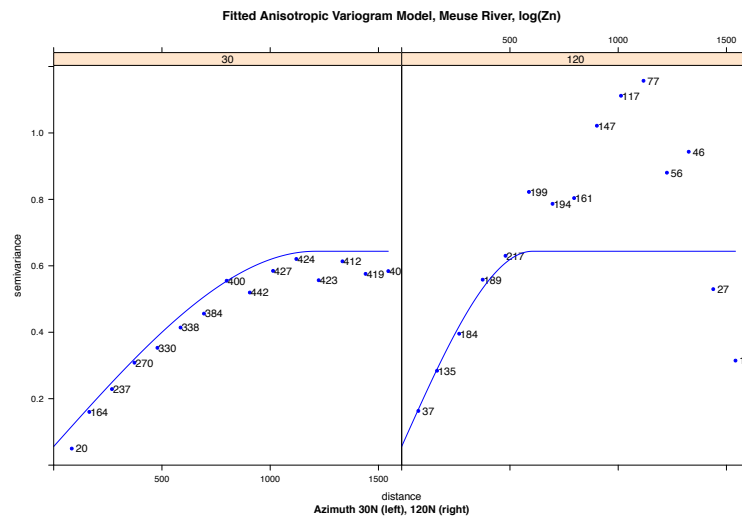
```
> (vmf.a <-
+   fit.variogram(v.a,
+                 vgm(0.55, "Sph", 1100, 0.05, anis=c(30, 0.5))))

      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(Zn)",
+   model=vmf.a,
+   sub="Azimuth 30N (left), 120N (right)",
+   pl=T, pch=20, col="blue"))
```




---

**Q39 :** What variogram parameters did method `fit.variogram` alter?  
[Jump to A39](#) •

---

**Task 42 :** Compute the best omnidirectional spherical model for the same variable and display the fitted model on the empirical variogram. •

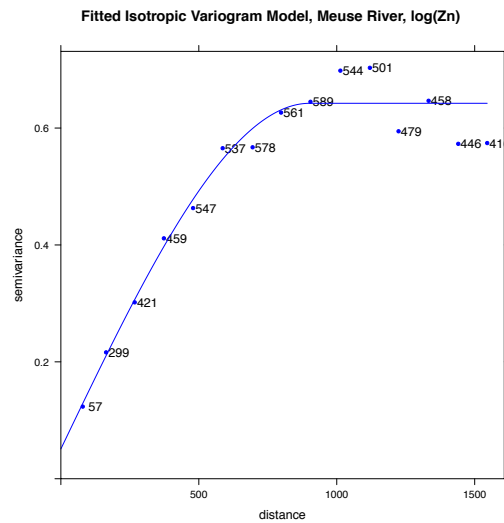
```
> v <- variogram(log(zinc)~1, meuse, cutoff=1600)
> (vmf <- fit.variogram(v, vgm(0.55, "Sph", 1100, 0.05)))

  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(Zn)",
+   model=vmf, pl=T, pch=20, col="blue"))
```




---

**Q40 :** Compare the two models: parameters and fit (visual and numerical).

[Jump to A40](#) •

---

**Q41 :** Would you use the isotropic or anisotropic variogram models for geostatistical prediction (kriging) in this study area?

[Jump to A41](#) •

---

**Task 43 :** 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 44 :** 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 45 :** 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 46 :** Display the four empirical variograms on the same graph. •

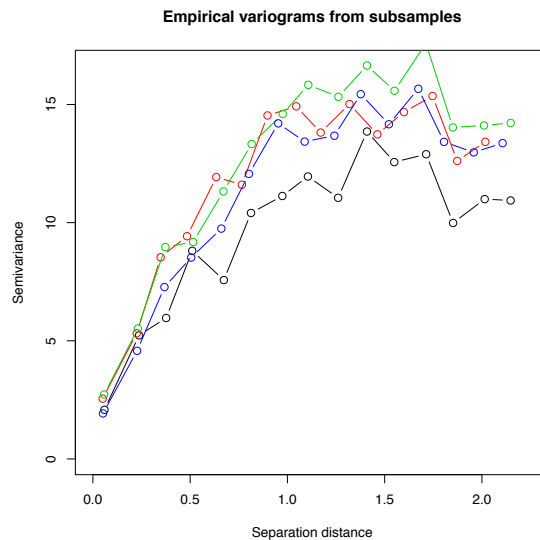
We use the `plot` method to display the first scatterplot of semivariance against separation, thereby setting up the plot (axes, labels, title etc.). We then use the `points` method to add point sets to the scatterplot; this is called once for each remaining sub-sample, inside a `for` loop.

**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")
+ }

```



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

---

**Q42 :** *How different are the four empirical variograms from each other?*

*Jump to A42 •*

---

**Task 47 :** 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  1.2767 0.000
2  Pen 10.6702 1.425

[[2]]
  model  psill range

```

```

1  Nug  1.6135  0.0000
2  Pen 12.7427  1.2817

```

```

[[3]]
      model  psill  range
1  Nug  1.7286  0.0000
2  Pen 13.9969  1.5069

```

```

[[4]]
      model  psill  range
1  Nug  1.0813  0.0000
2  Pen 13.2271  1.5232

```

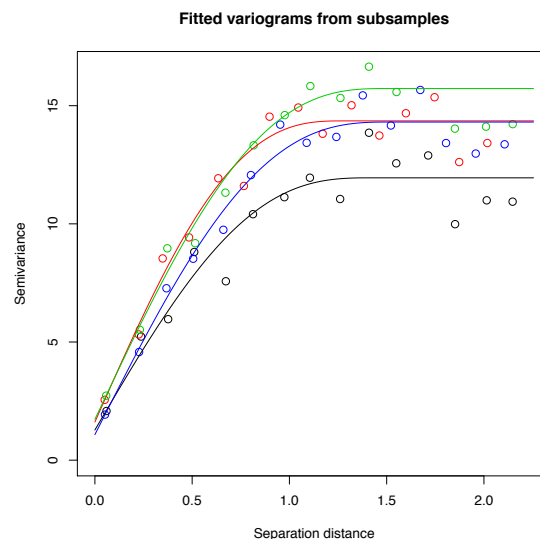
---

**Task 48 :** 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)

```




---

**Q43 :** How different are the four fitted variograms? Where do they differ most? Jump to A43 •

We can answer this question visually (looking at the figure) but also numer-

ically. 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)<sup>2</sup>.

```
> 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] 1.0813 1.2767 1.6135 1.7286

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

[1] 0.44795

> print(sort(psills))

[1] 10.670 12.743 13.227 13.997

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

[1] 0.25619

> print(sort(ranges))

[1] 1.2817 1.4250 1.5069 1.5232

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

[1] 0.16475

> 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?

---

**Q44 :** Which of these fits best models the spatial process? [Jump to A44](#) •

---

**Task 49 :** Remove the temporary objects created in this section from the workspace. •

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

---

<sup>2</sup> 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

---

**A16 :** The distribution is more-or-less normal with a mode near  $12 \text{ mg kg}^{-1}$ , but with a second mode near  $3 \text{ mg kg}^{-1}$ . 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 Q16](#) •

---

**A17 :** Yes, it does not keep increasing; it reaches a value of about  $14 \text{ mg kg}^{-12}$  near 1 km and then is more-or-less constant past this range. [Return to Q17](#) •

---

**A18 :** 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 Q18](#) •

---

**A19 :** Spherical, circular, pentaspherical. [Return to Q19](#) •

---

**A20 :** The total sill is about  $14 (\text{mg kg}^{-1})^2$ , of which the Nugget is about  $1.5 (\text{mg kg}^{-1})^2$ , so the structural sill is about  $14 - 1.5 = 12.5 (\text{mg kg}^{-1})^2$ ; Range 1.2 km. [Return to Q20](#) •

---

**A21 :** Because of the gradual transition “shoulder”, the pentaspherical model appears best. [Return to Q21](#) •

---

**A22 :** 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 Q22](#) •

---

**A23 :** 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 Q23](#) •

---

**A24 :** Very well. [Return to Q24](#) •

---

**A25 :** 90.4% [Return to Q25](#) •

---

**A26 :** 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 Q26](#) •

---

**A27 :** The fit has changed somewhat: the nugget is almost the same but the partial sill and range somewhat lower. [Return to Q27](#) •

---

**A28 :** 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 Q28](#) •

---

**A29 :** 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 Q29](#) •

---

**A30 :** Untransformed: strongly right-skewed; log-transformed: nearly symmetric. The log-transformed variable should be used, to avoid an erratic variogram. [Return to Q30](#) •

---

**A31 :** 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 Q31](#) •

---

**A32 :** 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 Q32](#) •

---

**A33 :** 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 Q33](#) •

---

**A34 :** 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 processes 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 Q34](#) •

---

**A35 :** 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 Q35](#) •

---

**A36 :** 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 Q36](#) •

---

**A37 :** No it does not! There seems to be more variability on the minor axis, i.e.

at azimuth 120°N.

[Return to Q37](#) •

---

**A38** : 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 Q38](#) •

---

**A39** : 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 Q39](#) •

---

**A40** : 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 Q40](#) •

---

**A41** : 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 Q41](#) •

---

**A42** : This varies each time the four sub-samples are drawn. In general, the close-range structure is more consistent.

[Return to Q42](#) •

---

**A43** : 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 Q43](#) •

---

**A44** : They are all equally-good estimates, there is no way to know which sub-sample gives (by chance) the closest match to the presumed “true” spatial process.

[Return to Q44](#) •

## 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`<sup>3</sup>; 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 50 :** Load the `spatial` and `MASS` packages, and the `topo` dataset; examine the dataset structure. •

```
> require(spatial)
> require(MASS)
> help(topo)
> str(topo)
> summary(topo)
```

---

**Q45 :** *What do the three fields represent?* Jump to A45 •

## 5.3 Step 1: OLS trend surface

We begin by fitting an OLS trend surface.

---

**Task 51 :** Make a postplot of the elevations. •

We use the `with` function to save some typing. This useful exposes the names inside its first argument (here, data frame `topo`) for the second argument (here, a compound command with `plot` and `text` functions). So we can write `y ~ x` instead of `topo$y ~ topo$x`, etc.

```
> with(topo, {
+   plot(y ~ x, cex = 3 * z/max(z), pch = 1, col = "blue",
+       asp = 1, main = "Elevation (feet)", xlab = "E",
```

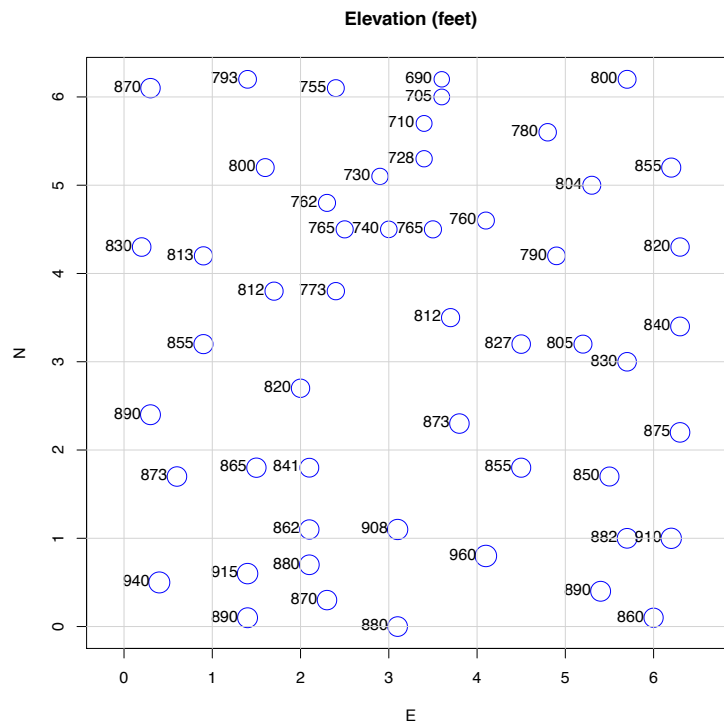
---

<sup>3</sup> Also available from the author’s web site, <http://www.kgs.ku.edu/Mathgeo/Books/Stat/index.html>

```

+       ylab = "N")
+       text(x, y, z, pos = 2)
+   })
> grid(lty = 1)

```




---

**Q46 :** What seems to be the regional trend?

*Jump to A46 •*

The form suggests a second-order surface.

---

**Task 52 :** 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, topo$x, topo$y, topo$z)
> class(ts2)

```

```
[1] "trls"
```

```
> summary(ts2)
```

Analysis of Variance Table

```

Model: surf.ls(np = 2, x = topo$x, y = topo$y, z = topo$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 53 :** 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 ...

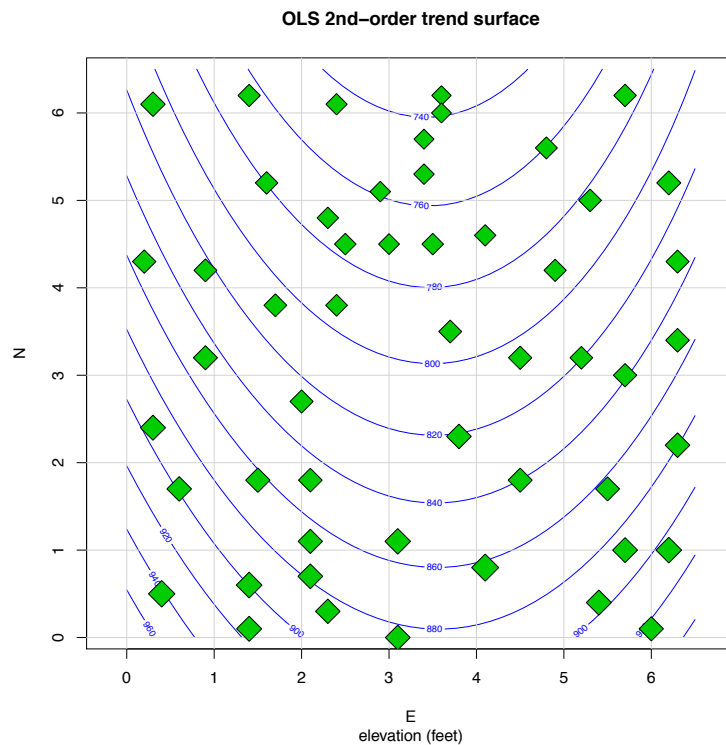
```

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(topo$x, topo$y, cex = 3 * (topo$z/max(topo$z)),
+        pch = 23, bg = 3)

```




---

**Q47 :** Describe the visual fit of this trend surface to the sample points.  
[Jump to A47](#) •

## 5.4 Step 2: OLS residuals

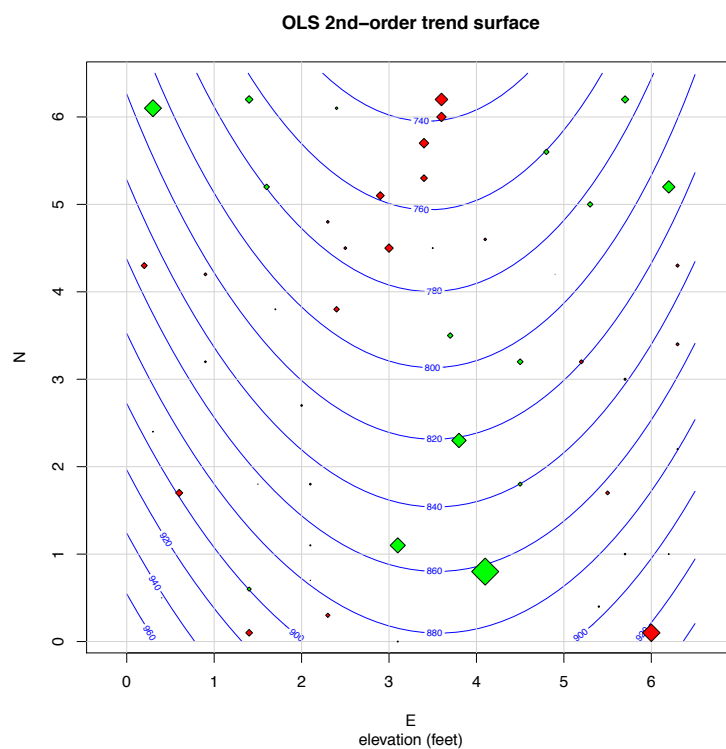
We now visualize the goodness-of-fit with the residuals.

---

**Task 54 :** 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(topo$x, topo$y, cex = 3 * abs(r)/max(abs(r)),
+       pch = 23, bg = ifelse(r < 0, "red", "green"))
> rm(r)
```

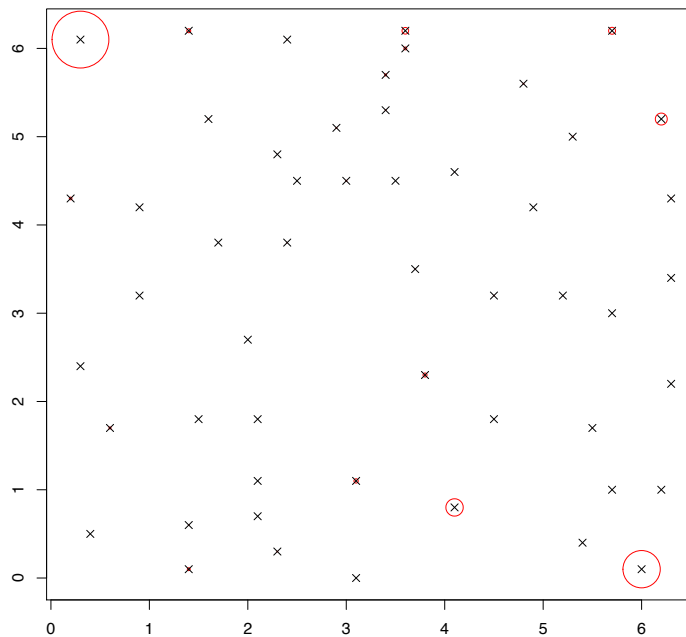



---

**Q48 :** *Do the residuals appear to be spatially correlated?* [Jump to A48](#) •

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 0.19 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  $> 1.5$ :

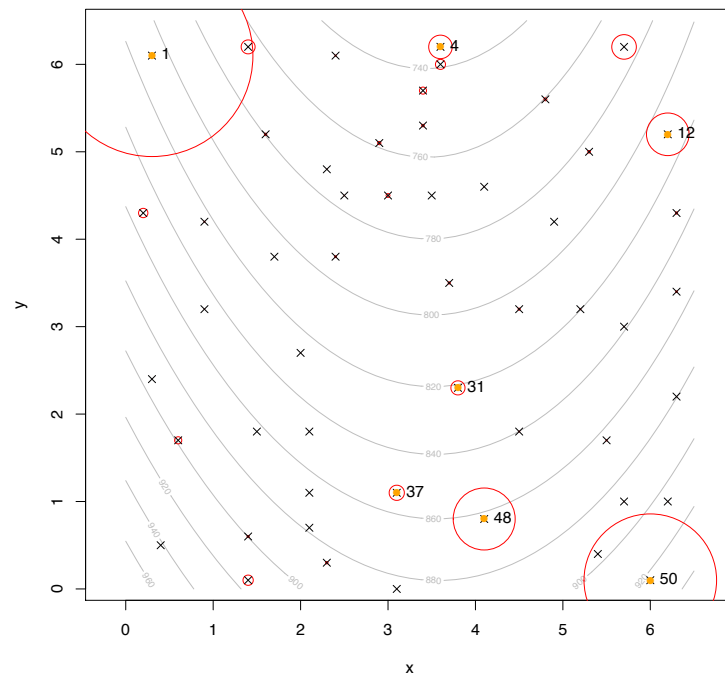
```
> (cand <- as.data.frame(infl.ts2)[abs(infl.ts2$stresid) >
+   1.5, ])
```

	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)
```

---

**Q49 :** Which points have the highest standardized residuals? Is there any spatial pattern to these? *Jump to A49 •*

### 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 **correlogram**, which is a plot of the correlation coefficient versus separation between the residuals at point-pairs.

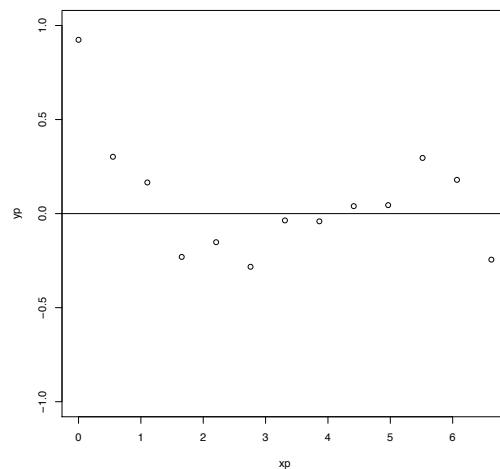
---

**Task 55 :** 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)
```

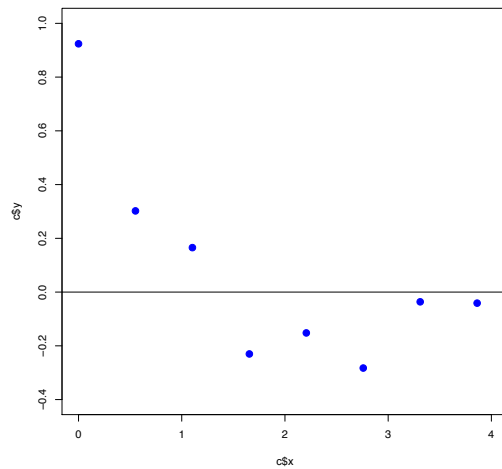
	x	y	cnt
1	0.00000	0.923919	65
2	0.55172	0.302105	73
3	1.10345	0.165462	107
4	1.65517	-0.230165	137
5	2.20690	-0.151937	137
6	2.75862	-0.282654	145
7	3.31035	-0.036221	155
8	3.86207	-0.041093	148
9	4.41380	0.039679	139
10	4.96552	0.045055	108
11	5.51725	0.295976	85
12	6.06897	0.179504	52
13	6.62069	-0.244652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



The default plot shows spatial correlation on a  $[-1 \dots +1]$  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)
```






---

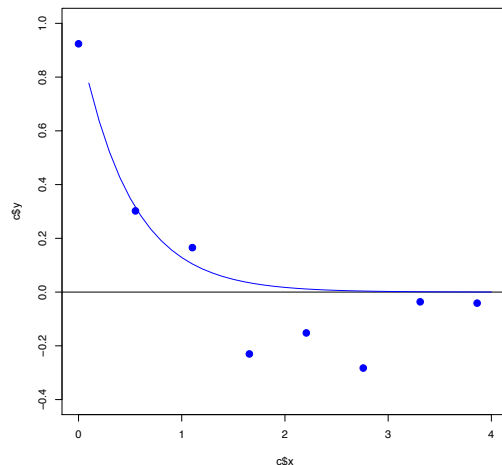
**Q50 :** Describe the spatial correlation (strong, moderate, weak?). What is the approximate range? What is the nugget effect as a proportion of the maximum correlation (1)? Are there enough point-pairs to make a reliable model? Jump to A50 •

---

**Task 56 :** 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 57 :** 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, topo$x,
+   topo$y, topo$z)
```

---

**Q51 :** How much did the internal goodness-of-fit ( $R^2$ ) change from the OLS to the GLS surface? *Jump to A51* •

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

Analysis of Variance Table

Model: `surf.gls(np = 2, covmod = expcov, x = topo$x, y = topo$y, z = topo$z,`  
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 = topo$x, y = topo$y, z = topo$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

```

---

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

```

> 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

```

---

**Q53 :** *How much did the residuals change from the OLS to the GLS surface?* Jump to A53 •

```

> 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.
-63.30  -14.40   -4.87    0.00   15.00   97.80

```

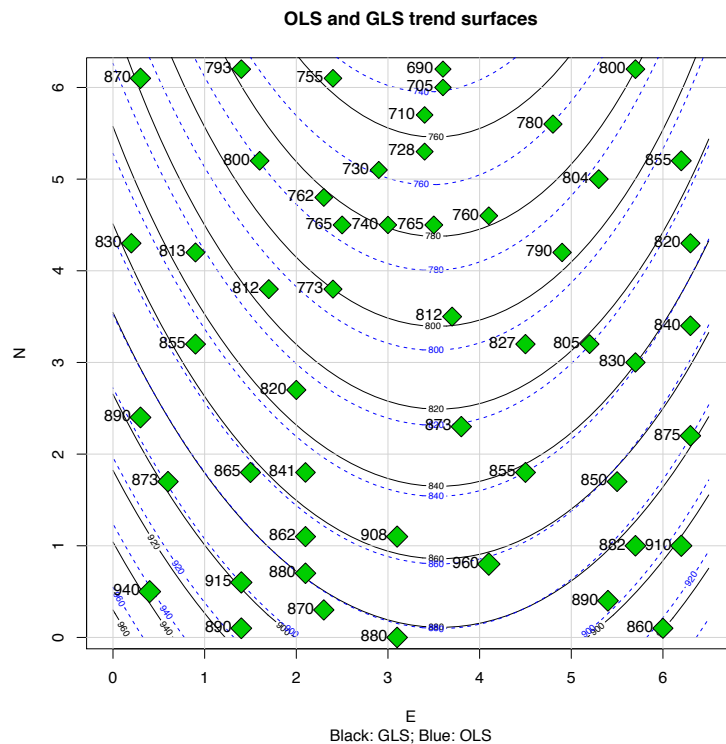
---

**Task 58 :** 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(topo$x, topo$y, cex = topo$z * 2.5/max(topo$z),
+   pch = 23, bg = 3)
> text(topo$x, topo$y, topo$z, pos = 2)

```




---

**Q54 :** Describe the differences between the OLS and GLS surfaces. [Jump to A54](#) •

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

---

**Task 59 :** Plot the 2nd-order GLS trend surfaces, with the residuals superimposed. •

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

```

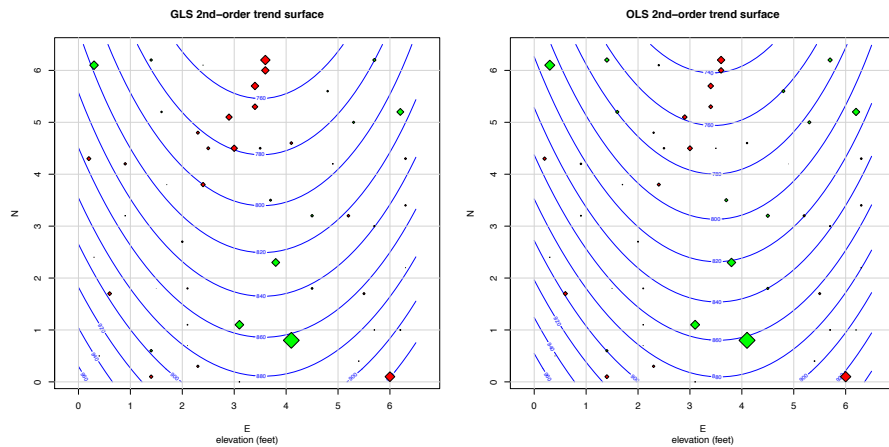
> par(mfrow = c(1, 2))
> r <- residuals(ts2)
> r.g <- residuals(ts2.g)
> r.max <- max(abs(r), abs(r.g))
> eqscplot(tmat2.g, type = "n", main = "GLS 2nd-order trend surface",
+   sub = "elevation (feet)", xlab = "E", ylab = "N")
> contour(tmat2.g, add = T, lty = 1, col = "blue")

```

```

> grid(lty = 1)
> points(topo$x, topo$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(topo$x, topo$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))

```




---

**Q55 :** *How is the spatial distribution of the residuals affected by a GLS fit, rather than an OLS fit?* Jump to A55 •

## 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 iteratively re-fit the GLS surface with the revised correlation structure.

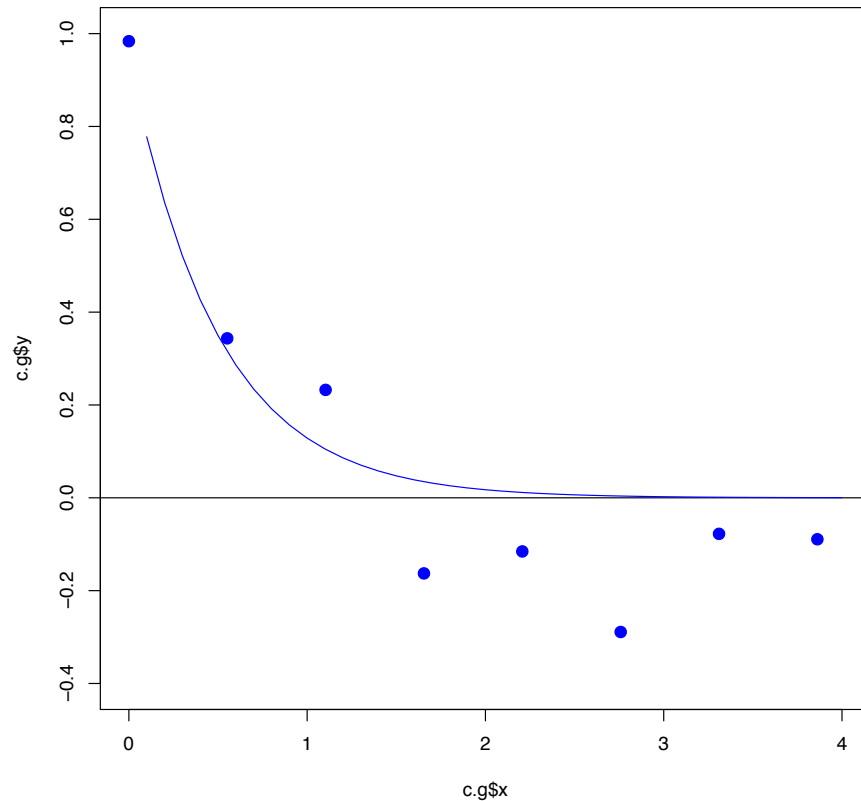
---

**Task 60 :** 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)

```



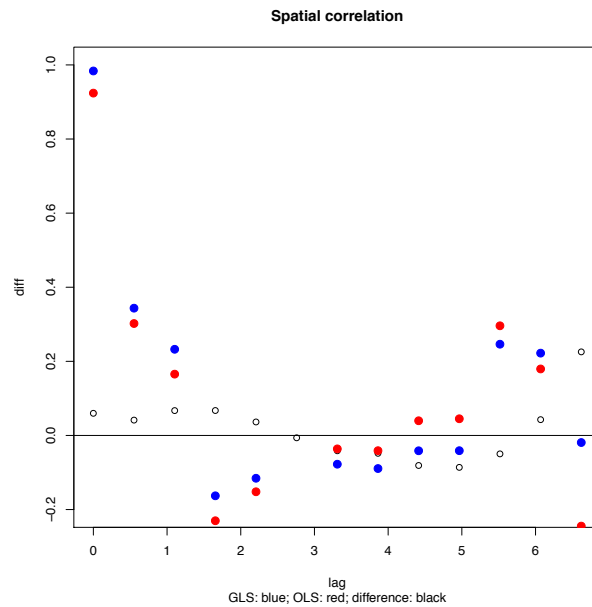

---

**Q56 :** Does the model from the OLS residuals appear to fit the GLS residuals? Jump to A56 •

---

**Task 61 :** 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$y))
> plot(diff ~ lag, data=tmp, ylim=c(-.2, 1), pch=1,
+       main="Spatial correlation",
+       sub="GLS: blue; OLS: red; difference: black")
> points(tmp$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)
```




---

**Q57 :** *Is there much difference between the two correlograms?* [Jump to A57](#) •

---

**Q58 :** *Is it necessary to iteratively fit the GLS surface?* [Jump to A58](#) •

---

**Task 62 :** Clean up the workspace from the previous three subsections. •

We also need to remove the `spatial` package from the search path, to avoid naming conflicts with the `gstat` package.

Both the `spatial` and `gstat` packages contains a `variogram` method. This is one of the disadvantages of open-source software – each package author picks their own function names, and there can be conflicts. If several packages are in memory, the last-loaded (here, `spatial`) takes precedence. We remove it with the `detach` function, specifying that the object to be removed is a package.

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

## 5.8 Answers

---

**A45 :** *x and y are local survey coördinates in units of 50 feet<sup>4</sup>; z is elevation above an unspecified base in feet.* [Return to Q45](#) •

---

<sup>4</sup> 1' = 0.3048 m

**A46 :** 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 Q46](#) •

---

**A47 :** The fit appears quite good. The bowl shape adjusts well to the sample points. [Return to Q47](#) •

---

**A48 :** 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 Q48](#) •

---

**A49 :** The largest standardized residuals (points 1, 50, 48, 12) are towards the outside of the bowl. [Return to Q49](#) •

---

**A50 :** 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 Q50](#) •

---

**A51 :** 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 Q51](#) •

---

**A52 :** The coefficients changed significantly, mostly by about 8% relative. [Return to Q52](#) •

---

**A53 :** The GLS residuals have a somewhat narrower range (smaller extremes); the mean residual is no longer  $\approx 0$ . [Return to Q53](#) •

---

**A54 :** 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 Q54](#) •

---

**A55 :** 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 Q55](#) •

---

**A56 :** 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 Q56](#) •

---

**A57 :** There is very little difference in the correlograms. The GLS residuals have slightly higher spatial correlation to 2 distance units, mainly due to a reduced



nugget (short-range correlation closer to 1).

[Return to Q57](#) •

---

**A58 :** *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 Q58](#) •

## 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.* •

Local  
spatial structure

---

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

---

**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. 54
- [2] J. C. Davis. *Statistics and data analysis in geology*. John Wiley & Sons, New York, 3rd edition, 2002. 55
- [3] P Goovaerts. *Geostatistics for natural resources evaluation*. Applied Geostatistics. Oxford University Press, New York; Oxford, 1997. 20
- [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. 28
- [6] B D Ripley. *Spatial statistics*. John Wiley and Sons, New York, 1981. 53, 55
- [7] W N Venables and B D Ripley. *Modern applied statistics with S*. Springer-Verlag, New York, fourth edition, 2002. 53, 55

## Index of R Concepts

- ! operator, 22
- formula operator, 8
- . formula operator, 8
- : operator, 26
- [[] operator, 47
- [] operator, 20
  
- abs, 14, 58
- add.to argument (vgm function), 41
- anis argument (vgm function), 44
- attributes, 35
  
- barplot, 12
- bbox (package:sp), 23
- bubble (package:lattice), 14, 15
- bubble (package:sp), 6, 17, 71
  
- c, 44
- cex graphics argument, 58
- contour, 57
- coordinates (package:sp), 16, 17, 71
- correlogram (package:spatial), 62
- cutoff argument (variogram function), 27
  
- detach, 69
- diff, 12
- dim, 21
  
- eqscplot (package:MASS), 57
- expcov (package:spatial), 63
  
- fit.method argument (fit.variogram function), 31
- fit.variogram (package:gstat), 30, 31, 33, 34, 45
- fitted, 3
- for operator, 47
- function, 23
  
- gaucov (package:spatial), 63
- gls, 55
- gstat package, 1, 2, 26, 28, 30, 32, 44, 64, 69
  
- hist, 12, 25
  
- I, 17
- ifelse, 58
- intersection, 22
- is.element, 22
  
- krige (package:gstat), 2
  
- lattice package, 14, 15
- lines, 23, 32
- lm, 2, 16, 54, 71
- loc argument (variogram function), 26
- log10, 72
  
- MASS package, 53, 55, 57
- matrix, 12
- max, 14
- meuse dataset, 43
- model argument (plot.gstatVariogram function), 29, 32
- more lattice graphics argument, 15
  
- n function argument, 57
- nlme package, 55
  
- pch graphics argument, 6
- plot, 29, 32, 47, 59
- plot graphics argument, 12
- plot.gstatVariogram (package:gstat), 29, 32
- plot.trls (package:spatial), 59
- points, 47
- pos argument (plot function), 33
- print (package:lattice), 14
- print, 14
- print.trellis (package:lattice), 14, 15
  
- range, 14
- residuals, 3
  
- sample, 21, 47
- seq, 26
- setdiff, 22
- setequal, 22
- sp package, 1, 2, 6, 15–17, 43
- spatial package, 53, 55, 57, 59, 62–64, 69
- sphercov (package:spatial), 63
- split lattice graphics argument, 15
- spplot (package:sp), 15
- summary, 12
- surf.gls (package:spatial), 61, 64
- surf.ls (package:spatial), 56, 61
  
- text, 33
- topo dataset, 55

trellis class, 14, 15  
trls.influence (package:spatial), 60  
trmat (package:spatial), 57  
type graphics argument, 47  
  
union, 22  
update, 8  
  
variogram (package:gstat), 26, 27  
variogramLine (package:gstat), 32  
vector, 47  
vgm (package:gstat), 28, 29, 41, 44  
  
with, 55  
  
xyplot (package:lattice), 15