Applied geostatistics
Exercise 5: Predicting from point samples (Part 2)

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

June 26, 2010

Contents

1 Kriging weights **1**
1.1 Answers . . . 3

2 Block kriging **4**
2.1 Answers . . . 8

3 Universal kriging **8**
3.1 Looking for a trend . . . 8
3.2 Fitting the residual variogram . . . 12
3.2.1 * Fixing the sill . . . 15
3.3 Predicting with UK . . . 16
3.4 * UK in a neighbourhood . . . 19
3.5 Answers . . . 20

4 * Insight into the UK system **20**
4.1 Answers . . . 25

5 Self-test **25**

References **28**

Index of R concepts **29**

Version 2.1. Copyright © 2006–2009 ITC, 2010 University of Twente 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).

玉不琢不成器 (Yù bù zhuó bù chéng qì)
"Jade that is not chiseled cannot become a gem"
– Chinese proverb

After completing this exercise you should be able to:

1. Understand how Ordinary Kriging weights are affected by point configuration and model
2. Predict block averages for attributes using block Ordinary Kriging
3. Predict attributes over a region using Universal Kriging

1 Kriging weights

In this section we take a short detour from R to investigate the Ordinary Kriging (OK) system in an interactive graphical program. This allows us to see the effect of variogram model, model parameters, sample point distribution, and prediction point location on the predictions and their variances.

For this section you must have access to an MS-Windows system. We will use the E{Z}-Kriging program[1] written by Dennis J. J. Walvoort, Wageningen University, Wageningen, The Netherlands.

> **Note:** If you don't have MS-Windows, or even if you do, we have also provided a **spreadsheet** named `OK_Explained.xls` in folder `datasets/xls` on the CD that can be investigated in Microsoft Excel or in the open-source office suite Open Office[2] calculator.

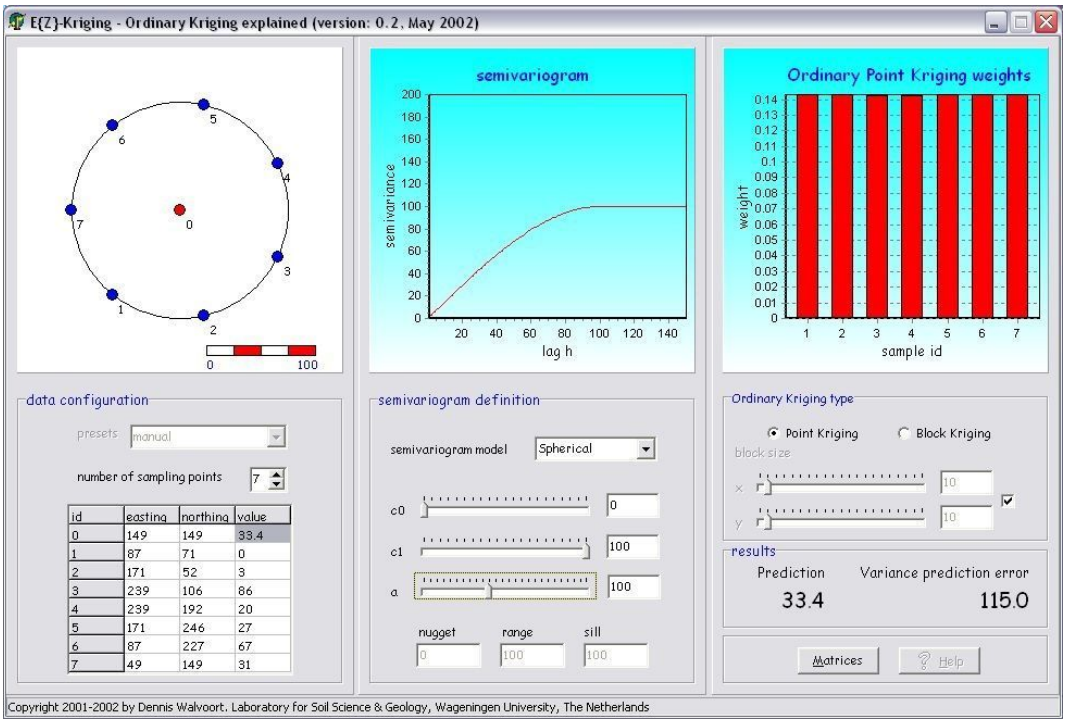
Task 1 : Install the program by unzipping file `E_Z_Kriging.zip` and view the user's manual `E_Z_Kriging.pdf`. •

Task 2 : Review sections 1 (Introduction) and 2 (Graphical User Interface) of the manual. This explains how to use the program. •

Task 3 : Open the program. •

The default configuration shows one prediction point in the middle of a circle of radius 100 with seven equally-spaced data points, a spherical semivariogram model with range 100, sill 100, and no nugget. The seven points have various data values.

[1] http://www.ai-geostats.org/index.php?id=114; also on the coure CD in directory `programs/freeware`
[2] http://www.openoffice.org/



Q1 : *What is the prediction and its variance?* *Jump to A1* •

Q2 : *What are the absolute and relative weights of the sample points?* *Jump to A2* •

Task 4 : Move point 2 next to point 1. •

Q3 : *What are now the weights? Explain.* *Jump to A3* •

Task 5 : Move point 2 half-way between point 1 and the prediction point, i.e. so that it 'masks' point 1. •

Q4 : *What are now the weights? Explain.* *Jump to A4* •

Task 6 : With the slider, reduce the range from 100 slowly towards 40. •

Q5 : *What happens to the weights as the range is reduced? Explain.* *Jump to A5* •

Task 7 : Move point 2 back to point 1, to form a two-point cluster. •

Q6 : *What are now the weights? Explain.* *Jump to A6* •

Task 8 : Restore the original configuration (equally-spaced points). With the slider increase and reduce the **range** parameter and observe the predictions and their variances. •

Q7 : *What happens to the predictions and their variances as the range is increased? reduced? Explain.* *Jump to A7* •

Task 9 : Restore the original configuration (equally-spaced points). With the slider increase the **nugget** and observe the predictions and their variances. •

Q8 : *What happens to the predictions and their variance as the nugget is increased? Explain.* *Jump to A8* •

This should give you some ideas on more experimentation.

1.1 Answers

A1 : *Prediction 33.4, variance 115 (remember, this is in squared units).* *Return to Q1* •

A2 : *All are equal, 1/7 = 0.142857...* *Return to Q2* •

A3 : *Points 3–7 have weights about 0.16, the weights of points 1 and 2 are reduced to about 0.9 (depending on exact configuration of the two points). Together the two points have only about 0.18 weight, this is the clustering effect.* *Return to Q3* •

A4 : *Point 2 now has about half the total weight (0.5); points 3–7 almost equal (about 0.1 each); point 1 now has a* ***negative*** *weight. The high weight for point 2 is explained by its proximity to the prediction point; the negative weight for point 1 because it is masked by point 2, i.e. in the same direction from the prediction point.* *Return to Q4* •

A5 : *They become more equal, until the range reaches the innermost point, at which time they all have equal weight. When all sample points are outside the*

range there is no longer any spatial dependence with the prediction point. *Return to Q5* •

A6 : *Points 1 and 2 have reduced weights, each about 0.1 (combined 0.2); the others have equal weights of about 0.15. Even though the range is short, there is still spatial dependence* ***between sample points*** *if they are closer than the range, so there is still a clustering effect in that case.* *Return to Q6* •

A7 : *Predictions do not change in this case because of the isotropic point distribution. Increased range decreases the kriging variance, and vice-versa. This is because the data values at sample points are more spatially-correlated with the prediction point at longer ranges.* *Return to Q7* •

A8 : *Predictions do not change, but the kriging variance is increased. There is spatial uncertainty at the sample points as well as at the prediction point.* *Return to Q8* •

2 Block kriging

This section introduces prediction on **blocks**, i.e. areas larger than the sample support. This is an easy modification of **punctual** OK prediction. The differences are:

1. the prediction is a **block average**, which smooths out local extremes;
2. the short-range variability (within a block) is removed, so the **kriging variances are lower** than for punctual OK.

We continue with the Jura dataset from Exercise 4.

Task 10 : To set up this exercise:

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 exercises.
2. If the calibration dataset `jura.cal` is not loaded as a spatial object, do so. Note: it was saved as part of R data file `JuraEx4.RData` in Exercise 4; this can be loaded into the workspace with the `load` method.
3. If the fitted variogram model `vmf` from Exercise 4 §4.1 is no longer in the workspace, re-create it.
4. If the prediction grid `jura.grid` from Exercise 4 §4.4 is no longer in the workspace, re-create it.
5. If the kriging prediction `k.grid` from Exercise 4 §4.4 is no longer in the workspace, re-create it.

•

If you followed the instructions in Exercise 4, these should all have been saved in file `JuraEx4.RData`, so they can be restored with the `load` method:

```
> load("JuraEx4.RData")
```

> **Note:** If this file is not in the current directory, you either have to change the directory with the `setwd` method, or add the full path to the argument of the `load` method.

Block kriging with the `krige` method of `gstat` is exactly like OK, with one additional argument: `block`, which gives the dimensions of the block as a list. For the usual case of a 2D block (surface area), this is a list of two dimensions (which are usually, but not necessarily the same). The list is formed with the `c` method.

Task 11 : Use the fitted variogram model to predict at all the grid points by Ordinary Kriging (OK) with a block size of 50 x 50 m. •

Note that the coördinates are in km, so the desired block size is expressed as 0.05 km.

```
> bk.grid <- krige(Co ~ 1, loc = jura.cal, newdata = jura.grid,
+     model = vmf, block = c(0.05, 0.05))

[using ordinary kriging]

> summary(bk.grid)

Object of class SpatialPixelsDataFrame
Coordinates:
   min  max
s1 0.3 5.10
s2 0.4 5.75
Is projected: NA
proj4string : [NA]
Number of points: 10272
Grid attributes:
   cellcentre.offset cellsize cells.dim
s1             0.325     0.05        96
s2             0.425     0.05       107
Data attributes:
   var1.pred        var1.var
 Min.   : 2.92   Min.   : 0.346
 1st Qu.: 8.48   1st Qu.: 2.496
 Median : 9.73   Median : 3.857
 Mean   : 9.54   Mean   : 6.097
 3rd Qu.:10.98   3rd Qu.:10.459
 Max.   :15.69   Max.   :13.223
```

Q9 : *What are the minimum, mean and maximum predictions? How do these compare with the values for the OK predictions of Exercise 4 §4.4? Explain any differences.* *Jump to A9* •

Q10 : *What are the minimum, mean and maximum kriging variances? How do these compare with the for the OK kriging variances of Exercise 4 §4.4? Explain any differences.* *Jump to A10* •

We can see the effect of blocking visually by side-by-side plots:

Task 12 : Plot the BK predictions and their variances, next to and with the same scale as the OK predictions and variances. •

To specify a scale to `spplot`, we use the `at` optional argument. We want to use the same scale, so we first compute the extremes of both prediction grids together, once for the predictions and once for their variances, with the `range` method:

```
> range(k.grid$var1.pred, bk.grid$var1.pred)

[1]  2.9057 15.8122

> range(k.grid$var1.var, bk.grid$var1.var)

[1]  0.3455 15.0050
```

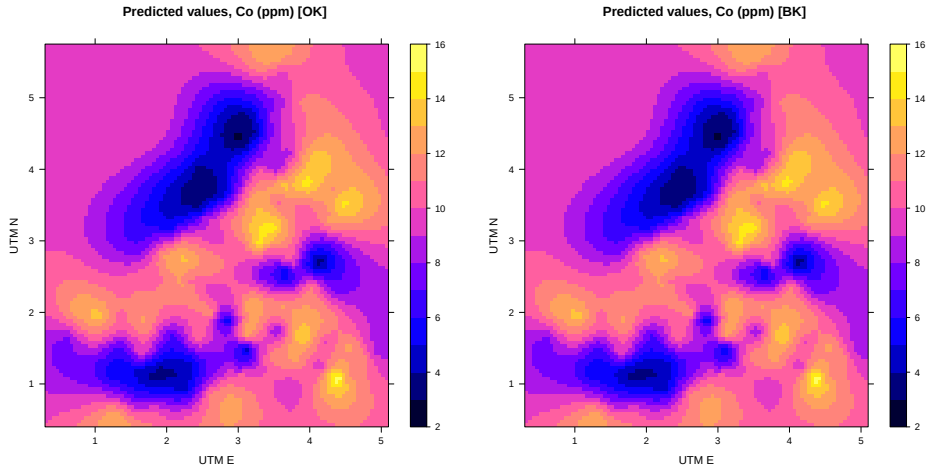
We round these a bit down and up, and break at each integer; we can use the `:` simple sequence operator for the integers, otherwise the `seq` method:

```
> (at.pred <- 2:16)

 [1]  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16

> (at.var <- seq(0, 15.5, by = 0.5))

 [1]  0.0  0.5  1.0  1.5  2.0  2.5  3.0  3.5  4.0  4.5  5.0  5.5  6.0
[14]  6.5  7.0  7.5  8.0  8.5  9.0  9.5 10.0 10.5 11.0 11.5 12.0 12.5
[27] 13.0 13.5 14.0 14.5 15.0 15.5
```

To plot several graphs together, we save each one as a Trellis graphics object, and then use the `split` and `more` optional arguments to the `print` method of the `lattice` package. (See [3, §5.4] for more on how to print several graphs in one window, or open multiple windows.)

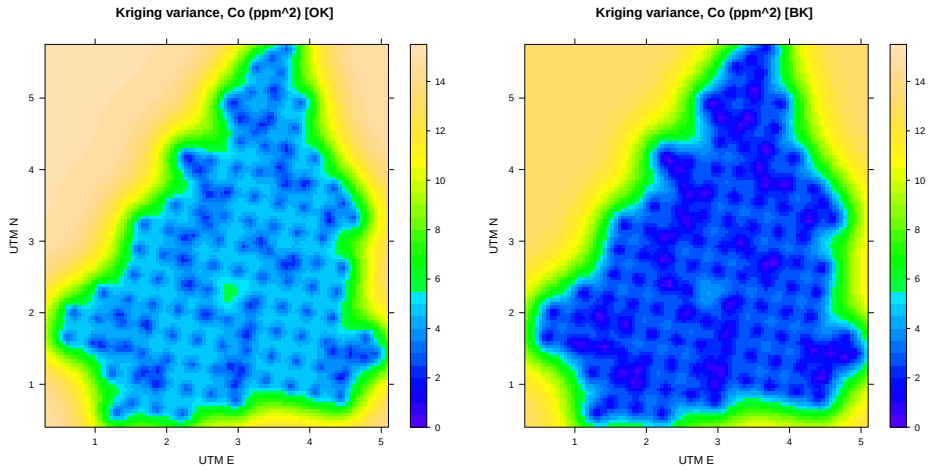
First the two predictions:

```
> plot.1 <- spplot(k.grid, zcol = "var1.pred", col.regions = bpy.colors(64),
+     main = "Predicted values, Co (ppm) [OK]", xlab = "UTM E",
+     ylab = "UTM N", scales = list(draw = T), at = at.pred)
> plot.2 <- spplot(bk.grid, zcol = "var1.pred", col.regions = bpy.colors(64),
+     main = "Predicted values, Co (ppm) [BK]", xlab = "UTM E",
+     ylab = "UTM N", scales = list(draw = T), at = at.pred)
> print(plot.1, split = c(1, 1, 2, 1), more = T)
> print(plot.2, split = c(2, 1, 2, 1), more = F)
```



Now the two kriging variances:

```
> plot.1 <- spplot(k.grid, zcol = "var1.var", col.regions = topo.colors(64),
+     main = "Kriging variance, Co (ppm^2) [OK]", xlab = "UTM E",
+     ylab = "UTM N", scales = list(draw = T), at = seq(0,
+         15.5, by = 0.5))
> plot.2 <- spplot(bk.grid, zcol = "var1.var", col.regions = topo.colors(64),
+     main = "Kriging variance, Co (ppm^2) [BK]", xlab = "UTM E",
+     ylab = "UTM N", scales = list(draw = T), at = seq(0,
+         15.5, by = 0.5))
> print(plot.1, split = c(1, 1, 2, 1), more = T)
> print(plot.2, split = c(2, 1, 2, 1), more = F)
```



Q11 : *Are there any differences in pattern or values with the OK predictions and variances from Exercise 4 §4.4?* *Jump to A11* •

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

We keep the interpolation grid `jura.grid`, the OK predictions `k.grid`, and the omnidirectional variogram model `vmf` for future use.

```
> rm(bk.grid, at.pred, at.var, plot.1, plot.2)
```

2.1 Answers

A9 : *Predicted values are almost identical.* *Return to Q9* •

A10 : *Kriging variances are substantially lower: the minimum is* 0.346 *for BOK,* 1.85 *for OK. The mean is* 6.097 *instead of* 7.86. *This reduction represents the variance within each block and the reduced variance between sample points and the prediction block.* *Return to Q10* •

A11 : *For the predicted values the two maps appear identical. For the variances the pattern is identical but the values are much lower with BK.* *Return to Q11* •

3 Universal kriging

Universal Kriging (UK) models both a **regional trend** and a **local spatial dependence** together. The trend is expressed as a function of the **coördinates**. The idea is that some of observed variation is due to a trend, and the variation in what is left over (the **residuals**) can be explained as a random field.

> **Note:** The term "Universal Kriging" is used by some authors also to include a trend in one or more **feature-space** predictors, i.e. co-variables. In this course we call this **Kriging with External Drift** (KED); the mathematics are the same but the predictors are not all in geographic space.

The steps in UK are:

1. Compute the regional **trend**;
2. Compute and model the variogram of the **residuals** from the regional trend;
3. Use this variogram model and the trend to **predict**.

In UK the trend is implicit in the kriging equations; this is useful if we want to restrict the kriging neighbourhood.

3.1 Looking for a trend

The first step, then, is to see if there is any regional trend. The R^2 of the trend gives some idea of the potential improvement in predictions.

For this section we continue with the Jura data set, specifically the calibration set `jura.cal`. We keep working with the target variable Cobalt concentration (attribute `Co`). This does not show a dramatic regional trend, so that UK does not differ too much from OK. Still, it illustrates the principles and procedures.

> **Note:** In this analysis we will use **ordinary least squares** (OLS) to compute the trend. This is only approximately correct; since we have evidence of spatial dependence between values, we should use **generalized least squares**

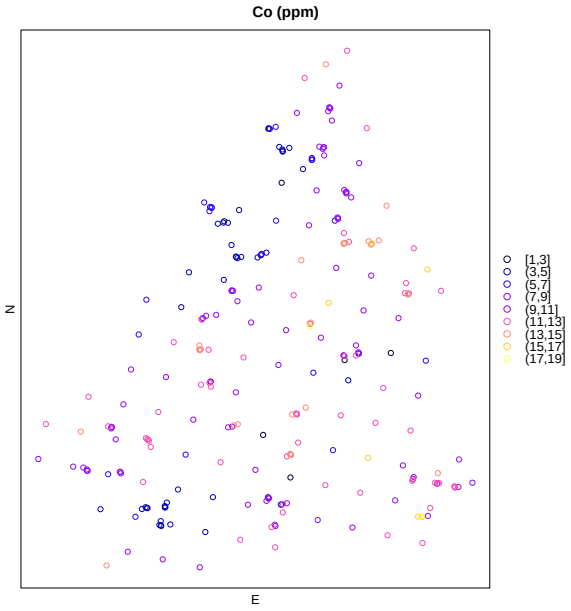
(GLS), weighting the points according to the **covariance structure**. But this leads to a "chicken-and-egg" problem: we don't know the covariance structure of the residuals until we compute them, and for that we need the trend, and for that we should know the covariance structure ...

In practice, OLS trend surfaces are used as a first approximation to the GLS surface, and the spatial structure of the residuals is modelled from this.

Task 14 : Visualize the regional trend in Co concentration. •

We use `spplot` with a colour ramp:

```
> print(spplot(jura.cal, zcol = "Co", col.regions = bpy.colors(8),
+     cuts = seq(1, 19, by = 2), key.space = "right", main = "Co (ppm)",
+     xlab = "E", ylab = "N", pch = 1))
```



Q12 : *Does there appear to be any regional trend in Co concentration? In what direction? How strong is it? Does it appear to be a plane or a higher-order surface?* *Jump to A12* •

Another way to visualize this is to compute a trend surface on the prediction grid using the `krige` of the `gstat` package without a model. However, before we can do this, we must ensure that the **names** of the coördinates within the must be the **same** in the prediction and sample datasets. Right now they are different, as we can see with the `dimnames` method:

```
> dimnames(coordinates(jura.cal))

[[1]]
NULL

[[2]]
[1] "X" "Y"
```

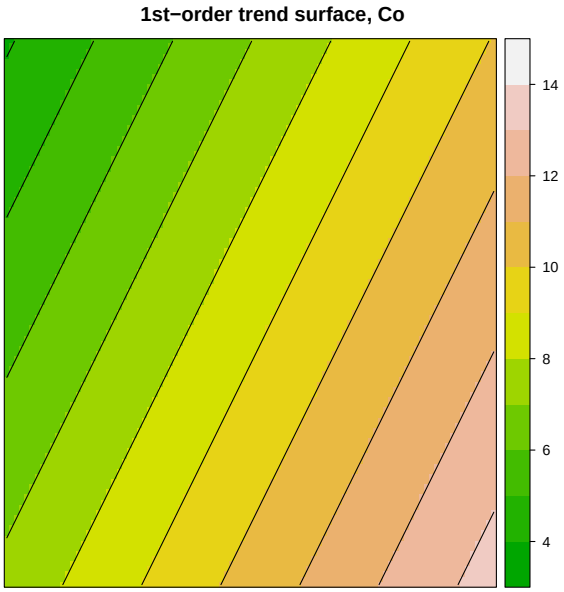
```
> dimnames(coordinates(jura.grid))

[[1]]
NULL

[[2]]
[1] "s1" "s2"
```

The dimensions of the grid were automatically named `s1` and `s2` when it was created by the `SpatialGrid` method in Exercise 4, and these were carried over to the spatial points. We can change these with the same the `dimnames` method; we have to acces the `@coords` and `@bbox` slots directly:

```
> dimnames(jura.grid@coords) <- list(NULL, c("X", "Y"))
> dimnames(jura.grid@coords)

[[1]]
NULL

[[2]]
[1] "X" "Y"

> dimnames(jura.grid@bbox) <- list(NULL, c("X", "Y"))
> dimnames(jura.grid@bbox)

[[1]]
NULL

[[2]]
[1] "X" "Y"
```

Now we can compute and display the trend surface:

```
> ts1 <- krige(Co ~ X + Y, loc = jura.cal, newdata = jura.grid,
+     model = NULL)

[ordinary or weighted least squares prediction]

> print(spplot(ts1, zcol = "var1.pred", contour = T, pretty = T,
+     col.regions = terrain.colors(64), main = "1st-order trend surface, Co"))
```



Q13 : *What is the approximate azimuth (direction) of this first-order trend surface?* *Jump to A13* •

Task 15 : Compute the regional trend. •

```
> (lm.co.xy <- lm(Co ~ coordinates(jura.cal), data = jura.cal))

Call:
lm(formula = Co ~ coordinates(jura.cal), data = jura.cal)

Coefficients:
           (Intercept)  coordinates(jura.cal)X
                 7.142                   1.297
coordinates(jura.cal)Y
                -0.639

> summary(lm.co.xy)

Call:
lm(formula = Co ~ coordinates(jura.cal), data = jura.cal)

Residuals:
   Min     1Q Median     3Q    Max
-9.200 -2.706  0.193  2.535  7.782

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)               7.142      0.732    9.75  < 2e-16 ***
coordinates(jura.cal)X    1.297      0.217    5.98  7.5e-09 ***
coordinates(jura.cal)Y   -0.639      0.163   -3.92  0.00011 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.32 on 256 degrees of freedom
```

```
Multiple R-squared: 0.145,        Adjusted R-squared: 0.139
F-statistic: 21.7 on 2 and 256 DF,  p-value: 1.9e-09
```

Q14 : *How much of the variation in Co concentration is explained by the trend?* *Jump to A14* •

Task 16 : Compute the direction of the trend. •

The two slope coefficients in E and N determine the **azimuth** (angle from N): the plane increases towards the angle determined by the ratio between them. The `atan2` method, in the form `atan2(y, x)`, returns the angle (in tadians) between the "x"-axis (as defined in the call) and the vector from the origin to "(x, y)" (also as defined in the call). For the azimuth, we want angle from N, so we call this in the form `atan2(E, N)` where E and N are extracted from the model fit with the `coefficients` method. This must be converted to degrees by the ratio $180/p$.

```
> coefficients(lm.co.xy)

           (Intercept) coordinates(jura.cal)X coordinates(jura.cal)Y
                7.1423                 1.2968                -0.6393

> atan2(coefficients(lm.co.xy)[2], coefficients(lm.co.xy)[3]) *
+     (180/pi)

coordinates(jura.cal)X
                116.24
```

Q15 : *In which direction does the plane increase?* *Jump to A15* •

3.2 Fitting the residual variogram

The next step is to compute the **residual variogram**. The `variogram` method of `gstat` computes the trend and, from that, the residuals, from the **model formula**. Recall: for OK we used the model formula `Co  1`, meaning to model `Co` from itself only (local spatial dependence). Now we use the formula `Co  coordinates(jura.cal)` to model the trend.

Task 17 : Compute the residual variogram of Co concentration, taking into account a first-order regional trend; compare it with the original variogram (no trend) both numerically and graphially. •

We will want to compare this variogram to that computed in Exercise 3, §3.3, so we re-compute that one; recall we used a cutoff of 1.6 km.

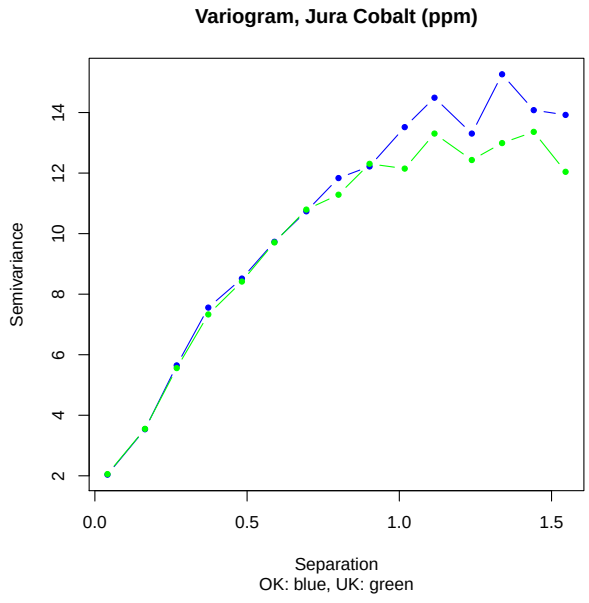
```
> v <- variogram(Co ~ 1, loc = jura.cal, cutoff = 1.6)
> vr <- variogram(Co ~ coordinates(jura.cal), loc = jura.cal,
+     cutoff = 1.6)
```

These have the same structure, the same distances, and the same numbers of point-pairs in each bin. So we can build one frame to compare them:

```
> (compare.v <- data.frame(np = v$np, separation = v$dist,
+     gamma.ok = v$gamma, gamma.uk = vr$gamma, gamma.dif = v$gamma -
+         vr$gamma))

     np separation gamma.ok gamma.uk gamma.dif
1   281   0.041378   2.0379   2.0569 -0.019034
2   209   0.164607   3.5373   3.5518 -0.014464
3   401   0.268811   5.6470   5.5585  0.088480
4   656   0.372567   7.5569   7.3306  0.226280
5   744   0.482890   8.5160   8.4170  0.098967
6   543   0.589803   9.7310   9.7087  0.022311
7   784   0.694563  10.7370  10.7946 -0.057611
8   966   0.800257  11.8357  11.2860  0.549710
9   749   0.902205  12.2191  12.3063 -0.087253
10 1115   1.017696  13.5181  12.1483  1.369825
11 1243   1.115531  14.4904  13.3057  1.184770
12 1170   1.238103  13.3056  12.4321  0.873472
13 1171   1.337529  15.2638  12.9918  2.272037
14 1266   1.441534  14.0793  13.3621  0.717197
15 1334   1.546090  13.9222  12.0425  1.879742
```

We can plot these on one graph, using the generic `plot` method to set up the plot and display the first variogram, and the `points` method to add the second variogram to the plot. Note the use of the `type="b"` optional argument to plot **both** the points and connecting lines:

```
> plot(compare.v$gamma.ok ~ compare.v$separation, pch = 20,
+     col = "blue", type = "b", xlab = "Separation", ylab = "Semivariance",
+     main = "Variogram, Jura Cobalt (ppm)", sub = "OK: blue, UK: green")
> points(compare.v$gamma.uk ~ compare.v$separation, pch = 20,
+     col = "green", type = "b")
```



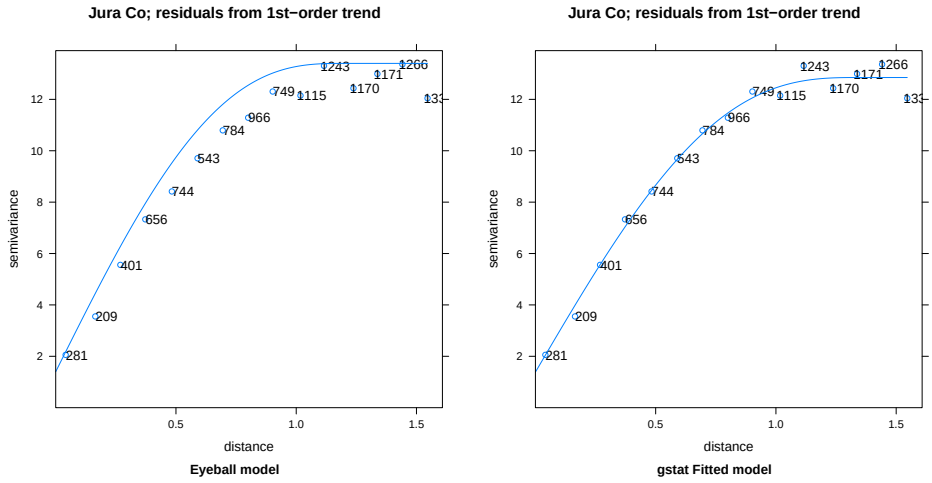
Q16 : *What are the major differences between the original and residual variograms? Why do they not differ very much?* *Jump to A16* •

The next step is to model the residual variogram.

Task 18 : Select a variogram model and fit parameters for the residual variogram. •

In the absence of any evidence to the contrary, we should use the same model form as for the original variogram, i.e. pentaspherical in this case. We first fit by eye and then automatically.We already know the fitted model for OK; it is a good starting point, with slightly reduced sill and range.

```
> vmf

  model   psill  range
1   Nug  1.3712 0.0000
2   Pen 12.9322 1.5239

> vrm <- vgm(12, "Pen", 1.2, 1.4)
> plot.1 <- plot(vr, pl = T, model = vrm, main = "Jura Co; residuals from 1st-order trend",
+     sub = "Eyeball model")
> (vrmf <- fit.variogram(vr, vrm))

  model   psill  range
1   Nug  1.3845 0.0000
2   Pen 11.4658 1.3469

> attributes(vrmf)$SSErr

[1] 3173.6

> plot.2 <- plot(vr, pl = T, model = vrmf, main = "Jura Co; residuals from 1st-order trend",
+     sub = "gstat Fitted model")
> print(plot.1, split = c(1, 1, 2, 1), more = T)
> print(plot.2, split = c(2, 1, 2, 1), more = F)
```



We compare the two parts of the model (nugget and structure); the first reported result is the difference in the nugget parameters and the second in

the structural model parameters. The range difference of the nugget is of course zero.

```
> vmf$range - vrmf$range
```

```
[1] 0.00000 0.17703
```

```
> vmf$psill - vrmf$psill
```

```
[1] -0.013331  1.466441
```

Q17 : *What are the differences between the original and residual variogram?* *Jump to A17* •

3.2.1 * Fixing the sill

This last difference reveals a problem: the nugget of the fitted residual variogram model is not the same as that for the original variogram model.

```
> vrmf$psill[1]
```

```
[1] 1.3845
```

```
> vmf$psill[1]
```

```
[1] 1.3712
```

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

```
[1] 0.009722
```

In this case the difference is quite small with respect to the total sill (about 1%), but still, in theory removing a long-range trend should have no effect on the nugget.

One way to handle this problem, well-justified by theory, is to fix the nugget from the original variogram surface and not allow it to change, specifying the optional `fit.sills` argument to the `fit.variogram` function, to fix the first sill (nugget) and allow the second (structural) to change. We take the nugget from the fitted model of original values, extracted from the fitted model object:

```
> (vrmf.2 <- fit.variogram(vr, vgm(12, "Pen", 1.2, vmf[1,
+     "psill"]), fit.sills = c(F, T)))
```

```
  model   psill  range
1   Nug  1.3712 0.0000
2   Pen 11.4670 1.3406
```

```
> attributes(vrmf.2)$SSErr
```

```
[1] 3193.6
```

Note that the nugget remains the same as specified from the original model. Comparing to the residual variogram fit without this restriction:

```
> print(vrmf)
```

```
  model   psill  range
1   Nug  1.3845 0.0000
2   Pen 11.4658 1.3469

> attributes(vrmf)$SSErr

[1] 3173.6

> rm(vrmf.2)
```

In this case, fixing the nugget slightly increases the structural sill and decreases the range; these are minor adjustments which would have little practical consequences. However, this technique is useful if the automatically-fitted nugget of a residual variogram is unrealistically low (see the self-test for an example).

3.3 Predicting with UK

Now we are in a position to predict with UK.

Task 19 : Predict the Co concentration over the region using UK. •

The `krige` command is almost the same as for OK; we just need to specify the trend and the changed variogram model.

> **Note:** The coördinate names of the prediction grid and the data frame with the sample points must be the same; we did that at the beginning of §3.

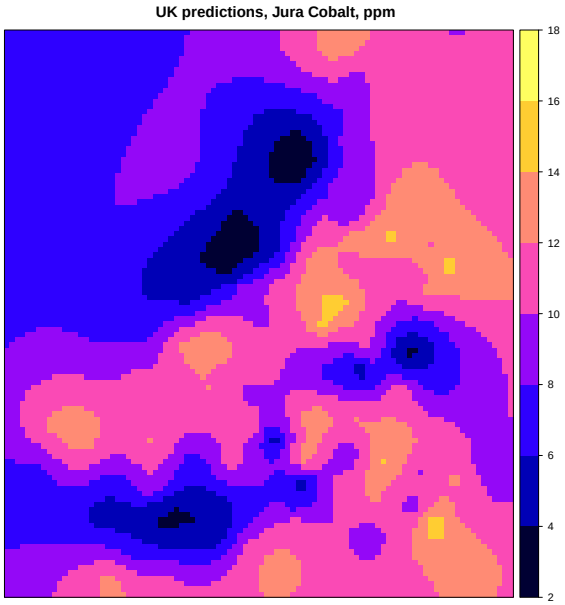
Finally we can compute the UK predictions; note the use of the formula `X + Y` to specify the trend; these variables refer both to the samples `jura.cal` and the prediction points `jura.grid`.

```
> uk.grid <- krige(Co ~ X + Y, loc = jura.cal, newdata = jura.grid,
+     model = vrmf)

[using universal kriging]

> summary(uk.grid)

Object of class SpatialPixelsDataFrame
Coordinates:
  min  max
X 0.3 5.10
Y 0.4 5.75
Is projected: NA
proj4string : [NA]
Number of points: 10272
Grid attributes:
  cellcentre.offset cellsize cells.dim
X             0.325     0.05        96
Y             0.425     0.05       107
Data attributes:
   var1.pred        var1.var
 Min.   : 2.93   Min.   : 1.86
 1st Qu.: 7.55   1st Qu.: 4.30
 Median : 9.38   Median : 5.70
 Mean   : 9.20   Mean   : 8.28
```

```
 3rd Qu.:11.01   3rd Qu.:12.74
 Max.   :15.82   Max.   :19.85
```

We then plot it:

```
> print(spplot(uk.grid, zcol = "var1.pred", col.regions = bpy.colors(64),
+     pretty = T, cuts = 8, main = "UK predictions, Jura Cobalt, ppm",
+     key.space = "right"))
```



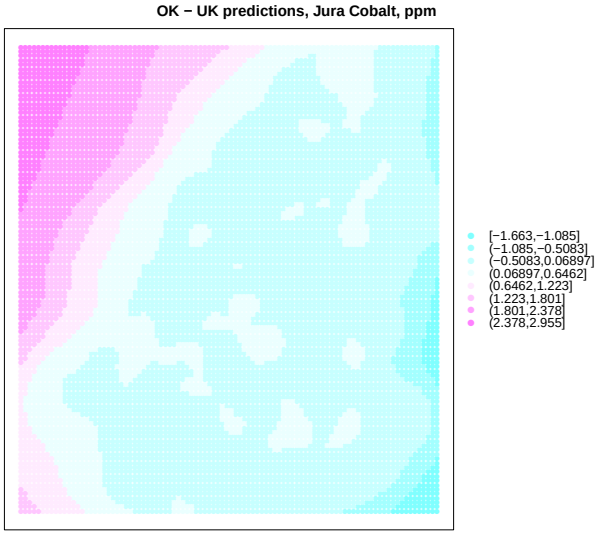
Task 20 : Compute and plot a difference map between UK and OK predictions and variances. •

First the computation: we make a new data frame, with two fields, which we explicitly name `pred` and `var`:

```
> diff.ok.uk <- data.frame(pred = k.grid$var1.pred - uk.grid$var1.pred,
+     var = k.grid$var1.var - uk.grid$var1.var)
> str(diff.ok.uk)

'data.frame':        10272 obs. of  2 variables:
 $ pred: num  2.96 2.92 2.88 2.85 2.81 ...
 $ var : num  -4.84 -4.72 -4.59 -4.47 -4.35 ...

> coordinates(diff.ok.uk) <- coordinates(uk.grid)
```

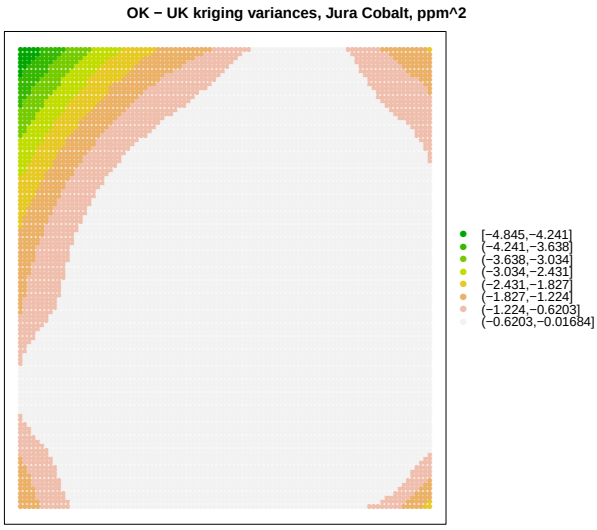
Then we plot these differences; first the predictions:

```
> print(spplot(diff.ok.uk, zcol = "pred", pretty = T, cuts = 8,
+     col.regions = cm.colors(64), main = "OK - UK predictions, Jura Cobalt, ppm",
+     key.space = "right"))
```



Then the variances:

```
> print(spplot(diff.ok.uk, zcol = "var", pretty = T, cuts = 8,
+     col.regions = terrain.colors(64), main = "OK - UK kriging variances, Jura Cobalt, ppm^
+     key.space = "right"))
```



Q18 : *What are the major differences? Explain.* *Jump to A18* •

Task 21 : Clean up from this section. •

```
> rm(diff.ok.uk)
```

3.4 * UK in a neighbourhood

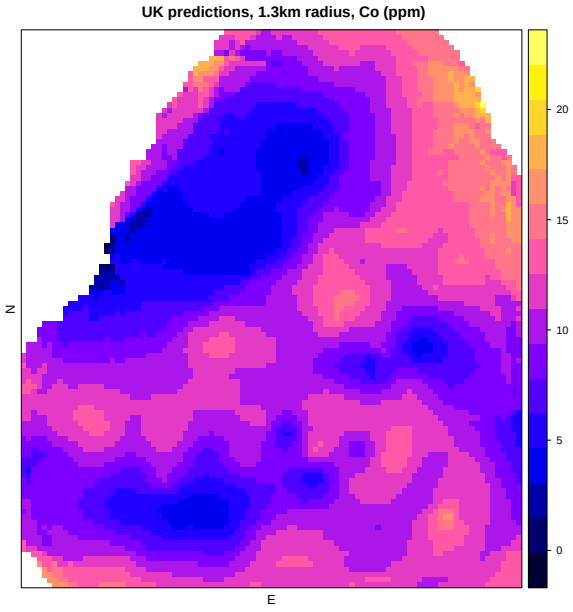
In this **optional** section we see how to restrict the UK system to a local **neighbourhood**. This allows local fitting of the trend surface, i.e. a non-stationary trend. There is no theory for the selection of the neighbourhood radius, but it often works well in practice if the radius is the **variogram range**, in this case about 1.3 km.

Task 22 : Compute UK predictions with a trend neighbourhood of 1.3 km; compare with the UK predictions with a regional trend. •

The only modifications to the `krige` method call are the addition of two optional arguments:

1. `maxdist`: the maximum distance from a prediction point to fit the trend surface; this restricts the neighbourhood;
2. `nmin`: the **minimum number of observations** to use to make a prediction; this ensures that a trend surface is not fitted with too few points.

```
> uk13.grid <- krige(Co ~ X + Y, loc = jura.cal, newdata = jura.grid,
+     model = vrmf, maxdist = 1.3, nmin = 12)

[using universal kriging]

> print(spplot(uk13.grid, zcol = "var1.pred", col.regions = bpy.colors(64),
+     main = "UK predictions, 1.3km radius, Co (ppm)",
+     xlab = "E", ylab = "N"))
```



Q19 : *What are the major differences between UK over the whole region and with a restricted neighbourhood? Explain.* *Jump to A19* •

3.5 Answers

A12 : *Yes; the lower values appear to be concentrated on the NW side and the higher on the SE. The trend appears to run to the ENE (increasing); a plane seems sufficient to model it.* *Return to Q12* •

A13 : *The increase is towards the ESE, approximately 120˚ from North.* *Return to Q13* •

A14 : 13.9% *is explained (see the adjusted* R^2*).* *Return to Q14* •

A15 : *Azimuth 116˚.* *Return to Q15* •

A16 : *They are quite similar up till about 1 km separation; then the residual variogram is significantly lower. This means that the trend has taken out some of the long-range structure. The short-range dependence is almost the same. The range of the residual variogram also appears to be a bit shorter. The small difference reflects the low* R^2 *of the trend surface.* *Return to Q16* •

A17 : *The residual variogram model has somewhat lower sill (about* 1.5 *lower) and shorter range (about* 0.18 *km shorter). This reflects the variability taken out by the trend. The nuggets are almost equal.* *Return to Q17* •

A18 : *The OK predictions are higher in the NW corner and lower in the SE corner than the UK predictions This is because the global trend increases from NW to SE.*

The OK kriging variances are much lower than for UK in the corners away from sample points. This is because there is no error from the trend surface, which error increases out from the centroid (centre of mass) of the sample points from which the surface was calculated. *Return to Q18* •

A19 : *The UK with restricted neighbourhood is patchy; the regional trend is not obvious. The blank areas are those without at least 12 observations within 1.3 km.* *Return to Q19* •

4 * Insight into the UK system

In this **optional** section we look at how `gstat` solves the universal kriging system. This is an extension of what we saw for the OK system in the previous exercise.

Because the size of the kriging matrix depends on the number of sample points, we first reduce the size of the problem.

Task 23 : Select the first six sample points of the prediction sample,

and their Co concentration. Define a one-point `SpatialPoints` object at $(3.0, 2.5)$ to be predicted.

Predict at this point with UK with a first-order trend, showing the kriging system, by using the optional `debug.level` argument to the `krige` method.

•

Note that the **residual** variogram model `vrmf` must be used for Universal Kriging.

The `debug.level` argument is passed by `krige` to the more general `predict.gstat` method; `?predict.gstat` shows various useful values; `debug.level=32` prints the covariance matrices, design matrices, solutions, and kriging weights.

```
> (jura.cal.6 <- jura.cal[1:6, "Co"])

     coordinates    Co
1 (2.386, 3.077)  9.32
2 (2.544, 1.972) 10.00
3 (2.807, 3.347) 10.60
4 (4.308, 1.933) 11.92
5 (4.383, 1.081) 16.32
6 (3.244, 4.519)  3.50

> (jura.pt <- SpatialPoints(data.frame(X = 3, Y = 2.5)))

SpatialPoints:
     X   Y
[1,] 3 2.5
Coordinate Reference System (CRS) arguments: NA

> k.pt <- krige(Co ~ X + Y, locations = jura.cal.6, newdata = jura.pt,
+     model = vrmf, debug.level = 32)

[using universal kriging]
we're at location X: 3 Y: 2.5 Z: 0
zero block size
we're at point X: 3 Y: 2.5 Z: 0

# X:
Matrix: 6 by 3
row 0:              1          2.386          3.077
row 1:              1          2.544          1.972
row 2:              1          2.807          3.347
row 3:              1          4.308          1.933
row 4:              1          4.383          1.081
row 5:              1          3.244          4.519
[using generalized covariances: max_val - semivariance()]
# Covariances (x_i, x_j) matrix C (lower triangle only):
Matrix: 6 by 6
row 0:     12.8502702              0              0              0
             0              0
row 1:    0.126071034     12.8502702              0              0
             0              0
row 2:     4.18617845              0     12.8502702              0
             0              0
row 3:              0              0              0     12.8502702
             0              0
row 4:              0              0              0     1.03997138
    12.8502702              0
```

```
row 5:              0              0  0.00984774609              0
            0     12.8502702

# X'C-1 X:
Matrix: 3 by 3
row 0:    0.415790555     1.37795277     1.09602191
row 1:     1.37795277     4.83623914      3.4492001
row 2:     1.09602191      3.4492001     3.45686012

# beta:
Vector: dim: 3
     15.894892    0.599556878    -2.91373745
# Cov(beta), (X'C-1 X)-1:
Matrix: 3 by 3
row 0:     96.9795145    -19.7729994    -11.0188553
row 1:    -19.7729994     4.74849477     1.53119647
row 2:    -11.0188553     1.53119647     2.25508225

# Corr(beta):
Matrix: 3 by 3
row 0:              1   -0.921413494   -0.745101238
row 1:   -0.921413494              1    0.467920145
row 2:   -0.745101238    0.467920145              1

# X0 (X values at prediction location x0):
Matrix: 3 by 1
row 0:              1
row 1:              3
row 2:            2.5

# BLUE(mu), E(y(x0)) = X0'beta:
Vector: dim: 1
    10.4092191
# Covariances (x_i, x_0), C0:
Matrix: 6 by 1
row 0:     1.11375274
row 1:     2.16156596
row 2:    0.965328826
row 3:              0
row 4:              0
row 5:              0

# C-1 C0:
Matrix: 6 by 1
row 0:   0.0677451658
row 1:    0.167547081
row 2:     0.05305226
row 3:              0
row 4:              0
row 5: -4.06563582e-05

# [a] Cov_ij(B,B) or Cov_ij(0,0):
Matrix: 1 by 1
row 0:     12.8502702

# [c] (x0-X'C-1 c0)'(X'C-1 X)-1(x0-X'C-1 c0):
Matrix: 1 by 1
row 0:     1.30739838

# [b] c0'C-1 c0:
```

```
Matrix: 1 by 1
row 0:    0.488828306

# Best Linear Unbiased Predictor:
Vector: dim: 1
    10.3409813
# MSPE ([a]-[b]+[c]):
Matrix: 1 by 1
row 0:     13.6688402

# kriging weights:
Matrix: 6 by 1
row 0:    0.199175674
row 1:    0.353321362
row 2:    0.147214249
row 3:   0.0970322758
row 4:    0.118795406
row 5:   0.0844610338
```

Working through this output, we see:

1. The **prediction point** and **block size** (here, zero);
2. The **design matrix** X with three columns: (1) a column of 1's to predict the spatial mean; (2) a column of the X coordinates of the sample points; (3) a column of the Y coordinates of the sample points;
3. The **covariance matrix** C between sample points; this depends of course on the **variogram model**;
4. The **quadratic form** $X^T C^{-1} X$ used in many of the subsequent calculations; this accounts for the covariance between sample points;
5. The **trend coefficients** β, estimated by GLS using the covariance matrix; note that this replaces the **spatial mean** from the OK system; there is a coefficient for the intercept and one each for each coordinate;
6. The **covariance** of the trend coefficients;
7. The **correlation matrix** of the trend coefficients;
8. The values of the **basis functions** at the prediction points, i.e. 1 and the actual coordinates of the point;
9. The **BLUE** of the trend at the prediction point;
10. The **covariance vector** C_o between the prediction point and each sample point;
11. The product $C^{-1} C_0$;
12. The **within-block** or **at-point** covariance, here just the total variogram sill (estimating the overall variance);
13. A matrix product used in the BLUP;
14. A matrix product used in the BLUP;
15. The **BLUP** at the prediction point; this is the **kriging prediction**;

16. The **kriging prediction variance** at the prediction point;
17. The **kriging weights**, i.e. the weights given to each sample point when their values are summed into the BLUP.

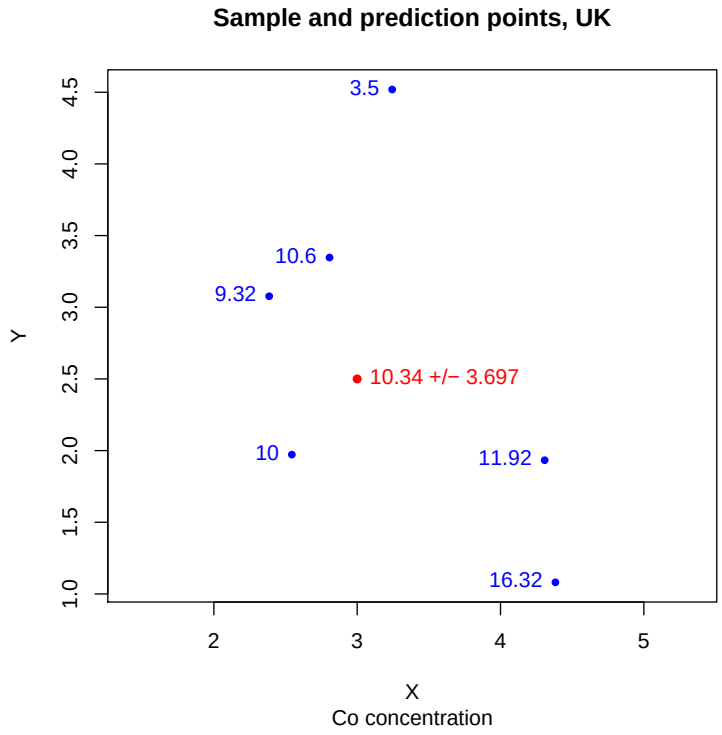
Q20 : *In which direction is the trend?* *Jump to A20* •

The final result is in the kriging object:

```
> print(k.pt)

  coordinates var1.pred var1.var
1    (3, 2.5)    10.341   13.669
```

Task 24 : Plot the sample points and their Co concentration, with the prediction point and its predicted concentration and standard error of the prediction. •

```
> plot(coordinates(jura.cal.6), col = "blue", pch = 20,
+     asp = 1, main = "Sample and prediction points, UK",
+     sub = "Co concentration")
> text(coordinates(jura.cal.6)[, "X"], coordinates(jura.cal.6)[,
+     "Y"], col = "blue", pos = 2, jura.cal.6$Co)
> points(coordinates(jura.pt), col = "red", cex = 1.2,
+     pch = 20)
> text(coordinates(jura.pt), col = "red", pos = 4, paste(round(k.pt$var1.pred,
+     2), "+/-", round(sqrt(k.pt$var1.var), 3)))
```



Note that the OK prediction (previous exercise) with this same system was

10.08 ± 3.764. The extra information on the trend has reduced the prediction variance somewhat, and the trend itself has changd the prediction; it is slightly higher because the point is a bit towards the SSE from the centroid.

Task 25 : Clean up from this section. •

```
> rm(jura.cal.6, jura.pt, k.pt)
```

4.1 Answers

A20 : *The trend is the plane defined by the six sample points, taking into account the spatial correlation. It is given by the β coefficients: $z = 15.8949 + 0.5996X - 2.9137Y$; the trend is increasing to the South-southeast.* *Return to Q20* •

5 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 exercise we use a small dataset of soil samples from Oxfordshire, UK, supplied as the example dataset `oxford` in the `gstat` package; this was originally collected by Burrough et al. [2] and is also used as an example in the text of Burrough and McDonnell [1].

Task 1 : Load the Oxford dataset into the workspace, examine its structure, and convert it to a spatial object by specifying the coördinates. •

Task 2 : Convert the `oxford` object from `SpatialPointsDataFrame` to `SpatialPixelsDataFrame` by specifying that the points are on a grid, with the `gridded` spatial method. •

Task 3 : Display a post-plot of attribute `CEC1` (cation exchange capacity of the topsoil). •

Q1 : *Does there appear to be a trend in the CEC across the study area? If so, in which direction?* •

Task 4 : **Optional**: Compute first- and second-order trend surfaces as linear models and summarize them. •

Q2 : ***Optional****: How much of the variability is explained by the surfaces?*

•

Q3 : ***Optional****: What is the azimuth of the direction of maximum increase for the first-order surface?*

•

Task 5 : **Optional**: Display these as contour plots.

•

Task 6 : Compute the ordinary variogram and the residual variogram from the suspected trends, and plot them on the same variogram graph.

•

Q4 : *What are the differences between the ordinary and residual variograms?*

•

Task 7 : Model the ordinary and residual variograms with the **same model form** but different fitted parameters.

•

Q5 : *What are the parameters of the fitted ordinary and residual variograms? By what proportion does the trend surface reduce the range and sill?*

•

Task 8 : Predict the CEC by Block Ordinary and Block Universal Kriging, on 100 x 100 m blocks centred at the sample points. Hint: you can use the same object as both `location` and `newdata`.

•

Task 9 : Display the prediction maps with the same colour scale.

•

Q6 : *Describe and explain the major spatial differences in the predictions.*

•

Task 10 : **Optional**: Display the prediction variances with the same colour scale.

•

Q7 : ***Optional****: Describe and explain the major spatial differences in the prediction variances.*

•

Task 11 : Clean up the workspace. •

References

[1] P A Burrough and R A McDonnell. *Principles of geographical information systems*. Oxford University Press, Oxford, 1998. 25

[2] P.A. Burrough, P.H.T. Beckett, and M.G. Jarvis. The relation between cost & utility in soil survey (I-III). *Journal of Soil Science*, 22(3):359–394, 1971. 25

[3] D G Rossiter. *Introduction to the R Project for Statistical Computing for use at ITC*. International Institute for Geo-information Science & Earth Observation (ITC), Enschede (NL), 2.95 edition, 2007. URL http://www.itc.nl/personal/rossiter/teach/R/RIntro_ITC.pdf. 6

Index of R Concepts

`:` operator, 6

`atan2`, 12

`c`, 5
`coefficients`, 12

`debug.level` method argument, 21
`dimnames`, 9, 10

`fit.variogram (package:gstat)`, 15

`gstat` package, 4, 5, 9, 12, 20, 25

`krige (package:gstat)`, 5, 9, 16, 19, 21

`lattice` package, 6
`load`, 4, 5

`oxford` dataset, 25

`plot`, 13
`points`, 13
`predict.gstat (package:gstat)`, 21
`print (package:lattice)`, 6

`range`, 6

`seq`, 6
`setwd`, 5
`sp` package, 4
`SpatialGrid (package:sp)`, 10
`SpatialPoints (package:sp)`, 21
`spplot (package:sp)`, 6, 9

`variogram (package:gstat)`, 12